

100%

ONE HUNDRED PERCENT

COMPREHENSIVE
AUTHORITATIVE
WHAT YOU NEED

ONE HUNDRED PERCENT

Master the material
for the **Oracle8i DBA**
Exam 1Z0-023

Test your knowledge
with assessment
questions, scenarios,
and lab exercises

Practice on
state-of-the-art
test-preparation
software

“Damir Bersinic delivers a clearly written, comprehensive guide — with hundreds of targeted exam-prep questions — that is sure to help Oracle professionals prepare for and pass the second Oracle certification exam.”

— *Harry Liebschutz, Senior Oracle Consultant, HDL Consulting, Inc., and author of The Oracle Cookbook*



Oracle⁸ⁱ™ DBA: Architecture and Administration

Certification Bible

**Damir Bersinic, Oracle Trainer, MCDBA, MCSE+I, MCT,
Todd Ross, and Yury Sabinin**

Hungry Minds Test
Engine powered by



**Oracle8i™ DBA:
Architecture and
Administration
Certification Bible**

Oracle8i™ DBA: Architecture and Administration Certification Bible

Damir Bersinic, Todd Ross, Yury Sabinin



Hungry Minds™

Best-Selling Books • Digital Downloads • e-Books • Answer Networks • e-Newsletters • Branded Web Sites • e-Learning

New York, NY ♦ Cleveland, OH ♦ Indianapolis, IN

**Oracle8i™ DBA: Architecture and Administration
Certification Bible**

Published by

Hungry Minds, Inc.

909 Third Avenue

New York, NY 10022

www.hungryminds.com

Copyright © 2002 Hungry Minds, Inc. All rights reserved. No part of this book, including interior design, cover design, and icons, may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording, or otherwise) without the prior written permission of the publisher.

Library of Congress Control Number: 2001092906

ISBN: 0-7645-4817-4

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

1P/TQ/RR/QR/IN

Distributed in the United States by
Hungry Minds, Inc.

Distributed by CDG Books Canada Inc. for Canada; by Transworld Publishers Limited in the United Kingdom; by IDG Norge Books for Norway; by IDG Sweden Books for Sweden; by IDG Books Australia Publishing Corporation Pty. Ltd. for Australia and New Zealand; by TransQuest Publishers Pte Ltd. for Singapore, Malaysia, Thailand, Indonesia, and Hong Kong; by Gotop Information Inc. for Taiwan; by ICG Muse, Inc. for Japan; by Intersoft for South Africa; by Eyrolles for France; by International Thomson Publishing for Germany, Austria, and Switzerland; by Distribuidora Cuspide for Argentina; by LR International for Brazil; by Galileo Libros for Chile; by Ediciones ZETA S.C.R. Ltda. for Peru;

by WS Computer Publishing Corporation, Inc., for the Philippines; by Contemporanea de Ediciones for Venezuela; by Express Computer Distributors for the Caribbean and West Indies; by Micronesia Media Distributor, Inc. for Micronesia; by Chips Computadoras S.A. de C.V. for Mexico; by Editorial Norma de Panama S.A. for Panama; by American Bookshops for Finland.

For general information on Hungry Minds' products and services please contact our Customer Care department within the U.S. at 800-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

For sales inquiries and reseller information, including discounts, premium and bulk quantity sales, and foreign-language translations, please contact our Customer Care department at 800-434-3422, fax 317-572-4002 or write to Hungry Minds, Inc., Attn: Customer Care Department, 10475 Crosspoint Boulevard, Indianapolis, IN 46256.

For information on licensing foreign or domestic rights, please contact our Sub-Rights Customer Care department at 212-884-5000.

For information on using Hungry Minds' products and services in the classroom or for ordering examination copies, please contact our Educational Sales department at 800-434-2086 or fax 317-572-4005.

For press review copies, author interviews, or other publicity information, please contact our Public Relations department at 317-572-3168 or fax 317-572-4168.

For authorization to photocopy items for corporate, personal, or educational use, please contact Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, or fax 978-750-4470.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND AUTHOR HAVE USED THEIR BEST EFFORTS IN PREPARING THIS BOOK. THE PUBLISHER AND AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS BOOK AND SPECIFICALLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THERE ARE NO WARRANTIES WHICH EXTEND BEYOND THE DESCRIPTIONS CONTAINED IN THIS PARAGRAPH. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES OR WRITTEN SALES MATERIALS. THE ACCURACY AND COMPLETENESS OF THE INFORMATION PROVIDED HEREIN AND THE OPINIONS STATED HEREIN ARE NOT GUARANTEED OR WARRANTED TO PRODUCE ANY PARTICULAR RESULTS, AND THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY INDIVIDUAL. NEITHER THE PUBLISHER NOR AUTHOR SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

Trademarks: Hungry Minds and related tradedress are trademarks of Hungry Minds, Inc. All other trademarks are the property of their respective owners. Hungry Minds, Inc., is not associated with any product or vendor mentioned in this book.

 is a trademark of
Hungry Minds Hungry Minds, Inc.

About the Authors

Damir Bersinic has over seventeen years of industry experience working with Oracle, SQL Server, Microsoft Windows NT/2000, BackOffice, and other advanced products. He is President and founder of Bradley Systems Incorporated, a Microsoft Certified Partner focusing on database, Internet, and system integration consulting and training. He holds a number of industry certifications including Oracle Certified Professional (DBA) for Oracle 7.3, 8, and 8i; Microsoft Certified Systems Engineer (MCSE); Microsoft Certified Database Administrator (MCDBA); Microsoft Certified Trainer (MCT); and Certified Technical Trainer (CTT). His extensive work with Oracle, SQL Server, Windows NT/2000, and BackOffice has enabled him to provide high-level consulting and other assistance to clients in Canada, the United States, and other parts of the globe. He is an avid fan of Formula 1 and CART racing and can be found in front of his large-screen TV on almost all race weekends. When he's not working, he enjoys time with his wife Visnja and son Anthony. Damir can be reached via e-mail at damir@bradsys.com.

Todd Ross is a Senior Oracle and UNIX instructor, as well as the Corporate Oracle Stream Leader, for TMI-Learnix Ltd. He is a Certified Oracle Professional in Oracle 7.3, 8.0, and 8i. He also runs an Oracle and Informix consulting company that specializes in database and SQL tuning as well as corporate backup and recovery strategies. Over the past eight years, Todd has been involved in Oracle implementations ranging from Oracle 6 to 8i on several different operating systems such as VAX, UNIX, NT, and 2000 all around the world. Todd is currently working with Oracle9i and trying to spend as much time as possible at home with his wife and two young children.

Yury Sabinin is an Oracle Certified Professional (DBA) living in Vancouver, British Columbia, Canada, with his wife and son. He has been involved in the industry since 1987. He is currently a full-time trainer with TMI-Learnix Ltd., teaching Oracle and other courses. He is also a consultant at SYM IT Services Ltd., a Vancouver consulting firm. In his spare time (not that he has any), he collects mechanical watches and tries to stay as far away from computers as possible.

Credits

Acquisitions Editor

Terri Varveris

Project Editor

Barbra Guerra

Technical Editor

Mike Hordila

Copy Editors

Barbra Guerra

Dennis Weaver

Susan Christophersen

Editorial Manager

Kyle Looper

Project Coordinator

Dale White

Graphics and Production Specialists

Laurie Petrone, Jill Piscitelli,

Jacque Schneider, Brian Torwelle,

Jeremy Unger

Quality Control Technicians

Laura Albert, David Faust,

John Greenough, Andy Hollandbeck,

Robert Springer

Permissions Editor

Laura Moss

Media Development Specialist

Travis Silver

Proofreading and Indexing

TECHBOOKS Production Services

I would like to dedicate this book to what a colleague of mine refers to as the lightbulbs—the look in the eye of a student or reader when the pieces fall into place. As you continue your journey in Oracle certification with this book, may you be blessed with many lightbulbs.

Damir Bersinic

Preface

Preparing to become an Oracle Certified Professional in the database administration (DBA) track is a journey that starts with a single step. If you have already passed the “Introduction to Oracle: SQL & PL/SQL” exam you have taken that first step and are well on your way. The next step in the process is to take and pass the “Oracle8i: Architecture and Administration” exam, a task which this book is designed to help you accomplish.

In order to become an Oracle Certified Professional (OCP) in the DBA track for Oracle8i, you will need to pass the following examinations:

- ◆ Exam 1Z0-001: Introduction to Oracle: SQL & PL/SQL
- ◆ Exam 1Z0-023: Oracle8i Architecture and Administration
- ◆ Exam 1Z0-025: Oracle8i Backup and Recovery
- ◆ Exam 1Z0-024: Oracle8i Performance and Tuning
- ◆ Exam 1Z0-026: Oracle8i Network Administration

The Oracle8i certification path is designed to lead you to the goal of becoming an OCP in a gradual process. There is no requirement to take the exams in the order they are intended so you do not need to take the exam for which this book prepares you right away. If you prefer to gain more experience and take another one of the exams first, this is acceptable. The only requirement to become an Oracle Certified Professional in the DBA track is that you pass all five exams, in no particular order.

For each of the above exams, Oracle provides a set of objectives that you will be tested on and required to be familiar with. The objectives for the “Oracle8i: Architecture and Administration” exam are listed in Appendix C, as well as a cross-reference of which chapters in this book provide information on that objective. The complete set of objectives for the entire track may be found in the *Oracle Certified Professional Program Candidate Guide, Oracle8i Certified Database Administrator Track* available on Oracle’s Web site at <http://www.oracle.com/education/certification/index.html?ocpguides.html>. **The exact link to the Adobe Acrobat PDF file for the DBA track is:** http://www.oracle.com/education/downloads/dba8i_cg.pdf.

The Candidate Guide document should be reviewed to gain a full understanding of the Oracle Certified Professional Program, as well as how to schedule your exams for your geographical area.

What's In This Book?

This book is divided into six parts.

- ◆ **Part I— Overview of Oracle Architecture and Administration.** Like in most things, it is always good to start at the beginning. This first part of the book does exactly that by introducing you to the architectural elements of Oracle so that you have a better sense of what components make up an Oracle database and instance. You will then find out what tools are available to administer an Oracle database. This part presents the 40,000-foot view of Oracle. All of the information you are introduced here is explained in much more detail in the parts of the book that follow.
- ◆ **Part II— Creating and Administering an Oracle Instance and Database.** After receiving the 40,000-foot view of Oracle in Part I of this book, Part II takes you to the next logical progression— creating a database and learning more about what each of the database components is used for and how to administer them. You will be shown how an instance is configured and started, and the steps required to create a database from scratch. This will be followed by information on how to manage key Oracle database files including the control files, redo log files, and datafiles. The emphasis of this part of the book is on the administration of Oracle's physical elements.
- ◆ **Part III— Administering Storage for Oracle Objects.** Knowing how to create a database, tablespaces, and datafiles is only one part of administering Oracle. You also need to be aware of issues surrounding the proper location of various objects in Oracle that require storage (segments). This part of the book, divided into five chapters that first introduce you to the different types of segments in an Oracle database, then follows with a discussion on each segment's particular storage needs, and finishes with a discussion on how to enforce data integrity in the database and the impact of constraints. The focus of this part of the book is the logical structure of the Oracle database and how to ensure that the database is operating as efficiently as possible. Even though a thorough discussion on performance tuning is beyond the scope of this book, Part III will provide hints on the efficient use of space for each segment type.
- ◆ **Part IV— Managing Oracle Data.** During your career as a database administrator there will be times when you will need to load large amounts of data into your database, move data from one database to another, or simply reorganize data in an existing database to improve performance and provide for more efficient storage. The use of Import and Export utilities, and other means to reorganize the data in the database to ensure efficient data retrieval, and how to load data into an Oracle database using SQL*Loader are covered in the two chapters that make up Part IV.
- ◆ **Part V— Administering Oracle Security.** This part of the book deals with database security, from understanding how configure security profiles,

creating and managing users, assigning permissions and the different types of permissions available, all the way to simplifying the administration of permissions using roles. As a database administrator it will be your responsibility to ensure that the database is not compromised and that only those individuals who need access have it.

- ◆ **Part VI—National Language Support.** Oracle introduced National Language Support to make it easier to deploy databases that will be accessed by users in multiple geographic areas simultaneously. This feature allows national language sorting, error message display, and much more. The single chapter in this part allows you to gain an understanding of how to use this ability to the benefit of your users.

The book has been divided into these parts from the standpoint of organization and logic and not to satisfy the exam requirements in a one-to-one mapping. Each objective of the exam is covered by material in the book. Appendix C lists the exam objectives and the relevant section of the book where each is covered.

The appendixes of the book provide a number of useful items:

- ◆ **Appendix A** outlines the programs and other information found on the CD-ROM provided. These include an electronic copy of the book in PDF format, a sample exam with 300 or so questions that will help you to prepare to write the real exam, and many other useful software programs.
- ◆ **Appendix B** contains a sample exam with the same number of questions as found in the actual Oracle exam. The styles of questions in this appendix are designed to closely emulate the type of questions you can expect on the exam and therefore help to prepare for it. Answers to each question are also provided, along with an explanation of why the correct answer is correct and the others are incorrect.
- ◆ **Appendix C** provides a mapping of the “Oracle8i: Architecture and Administration” exam objectives to the chapters and sections where these objectives are covered in the book. If you find that you are weak in a particular area, this appendix will let you focus your study by pointing you to the correct part of the book with the information you require.
- ◆ **Appendix D** contains information on what to do to properly prepare for the exam, and what some of the rules are.
- ◆ **Appendix E** lists the objects used by the exercises at the end of each chapter, as well as providing a hard copy of the scripts used to create and populate the objects.
- ◆ **Appendix F** recommends some other books, Web sites, and other resources that may be useful in helping you prepare for the exam, as well as increase your knowledge of relational databases and Oracle.

How to Use This Book

When asked what the best place to start would be, a wise individual once said *at the beginning*. This same truth holds true here. The parts and chapters of this book are meant to build upon concepts presented in previous chapters. Similarly, the labs of each chapter may also depend upon the results of labs in previous chapters. While this last point is not a hard and fast rule, it should be kept in mind. The recommended practice is to read the book from beginning to end and absorb the material in the order presented.

Each chapter, as well, follows a consistent structure. You are first presented with a series of questions to test your knowledge of the material in the chapter. This is useful if you feel that you are quite familiar with the topics about to be presented—the Chapter Pre-Test questions allow you to verify that you have a good grasp of the information.

Following the Chapter Pre-Test, the main body of the chapter is presented. You will be introduced to concepts and shown examples of code, as appropriate. Feel free to try the code as you are reading the material to verify that you get the same results as the author. In some cases, the same steps may be followed in the labs at the end of the chapter, so if the code does not work, check that section of the chapter.

Each chapter ends with a Key Point Summary, designed to reinforce the main elements presented in the chapter, and then presents a series of exam-style questions to test your understanding of the material. This is followed by one or more scenario-based questions to further test your understanding of the material. You are then asked to complete some Lab Exercises to provide you with some hands-on experience in the material. You are free to do the labs before answering the Quick Assessment exam-style questions.

Answers for the Chapter Pre-Test, Quick Assessment, Scenarios, and Lab Exercises are provided at the end of the chapter. These are designed to allow you to confirm the results of your work, and will provide additional information where necessary.

After you have read each chapter, you should use the CD-based exam to further test your knowledge. When you feel that you are ready, you can then try your hand at the Practice Exam in Appendix B. If you feel that you have sufficient information, go ahead and schedule your exam.

The night before taking the exam, review each chapter's Key Point Summary to make sure you understand the material, and answer the Chapter Pre-Test questions again. After passing the exam, the book can also make an excellent reference for any DBA, so keep it handy.

Conventions Used in This Book

Whenever command syntax was presented it will appear similar to the following:

```
CREATE [GLOBAL TEMPORARY] TABLE [schema.]tablename  
  (columnname datatype [NULL | NOT NULL] [DEFAULT expression]  
  [, ...]);
```

- ◆ Any elements of the command that are required are presented in uppercase, such as the words CREATE and TABLE in the above example.
- ◆ Any elements of the command language that are optional are presented in uppercase and enclosed in square brackets (“[” and “]”), such as the [GLOBAL TEMPORARY] option in the above example.
- ◆ Any element that is a substitution variable for an object name in a code example is presented in lowercase, such as the word “tablename” or “columnname” or “datatype” in the above example.
- ◆ Any element that is an optional substitution variable for an object name component, or an object name, such as the word “schema” in the above example, will be all lowercase and enclosed in square brackets.
- ◆ Any optional element of the command syntax will always be enclosed in square brackets.
- ◆ Any portion of the command syntax where mutually exclusive choices need to be made, will separate those choices by a pipe symbol “|”. An example of this is the choice between NULL and NOT NULL in the above syntax.
- ◆ Any element that can be repeated several times within the code syntax will be represented by three dots (...).
- ◆ Any element that is underlined denotes the default option for the command.

What the Icons Mean

Throughout the book, we’ve used *icons* in the left margin to call your attention to points that are particularly important.

Objective

The Objective icon indicates that the material being presented in the following section directly maps to an objective on the “Oracle8i: Architecture and Administration” exam.

Note

Note icons are used to tell you that something is important and perhaps a concept that may help you master the task at hand or something fundamental for understanding subsequent material.



Tip icons indicate a more efficient way of doing something, or a technique that may not be obvious.



The Exam Tip icon is used to help you focus your studies more precisely. Throughout the book many topics are covered in more detail than necessary to pass the exam. The Exam Tip icon provides information on which elements you will most likely be tested on.



Caution icons are used when performing a certain operation incorrectly or in a particular way can cause problems down the road.



The Cross Reference icon is used to refer you to other sections of the same chapter or other chapters that have more to say on a subject.



The In The Real World icon is used to present information on how experienced DBAs perform certain tasks and what really happens versus what Oracle may want you to think is happening. This icon also may present an alternative method of performing a certain task.

Acknowledgments

I would like to acknowledge the hard work put in by my contributing authors on this title. It was a new experience to some of you but you all came through it with flying colors. This was greatly helped by the assistance of the entire Hungry Minds team, who pushed us when we needed to be pushed, but let us do what we needed to do otherwise.

Each of us would also like to acknowledge the contributions our families made to this project. Their understanding when would rather be with them, but needed to write, helped make the process less painful. They also know that the smoother the process was for us involved, the sooner we would be able to spend more time with them. To our wives, husbands, and children: we are always grateful for your support and understanding, even if we don't show it all the time.

Finally, if you read the short snippets of information on the authors contributing to this book, you will note that two out of three work for the same company: TMI-Learnix. A special thank you goes out to Tim Mabey and Mia Hempey who not only allowed me to grab some of their trainer's free time and get them involved on this project, but also actively encouraged it. Working with you folks has always been, and continues to be, a pleasure.

Contents at a Glance

Preface	viii
Acknowledgments	xiv

Part I: Overview of Oracle Architecture and Administration 1

Chapter 1: Oracle8i Architecture	3
Chapter 2: Getting Started with Oracle8i Server	35

Part II: Creating and Administering an Oracle Instance and Database 89

Chapter 3: Managing an Oracle Instance	91
Chapter 4: Creating a Database	139
Chapter 5: Creating Data Dictionary Views and Standard Packages	193
Chapter 6: Maintaining the Control File	253
Chapter 7: Maintaining Redo Log Files	273
Chapter 8: Managing Tablespaces and Datafiles	325

Part III: Administering Storage for Oracle Objects 387

Chapter 9: Storage Structure and Relationships	389
Chapter 10: Managing Rollback Segments	425
Chapter 11: Managing Tables	485
Chapter 12: Managing Indexes	579
Chapter 13: Maintaining Data Integrity	629

Part IV: Managing Oracle Data 683

Chapter 14: Loading Data	685
Chapter 15: Reorganizing Data	731

Part V: Managing Oracle Security 797

Chapter 16: Managing Password Security and Resources	799
Chapter 17: Managing Users	845
Chapter 18: Managing Privileges	891
Chapter 19: Managing Roles	973

Part VI: National Language Support	1021
Chapter 20: Using National Language Support	1023
Appendix A: What's on the CD-ROM	1065
Appendix B: Practice Exam	1071
Appendix C: Objective Mapping	1095
Appendix D: Exam Tips	1103
Appendix E: Database Schema for Labs	1107
Appendix F: Suggested Readings, Web Sites, and Other Resources	1125
Index	1129
End-User License Agreement	1168

Contents

Preface	viii
Acknowledgments	xiv

Part I: Overview of Oracle Architecture and Administration 1

Chapter 1: Oracle8i Architecture	3
Oracle Architectural Overview	5
Oracle instance	7
Oracle database	18
Other key Oracle files	20
Processing SQL Statements	22
Connecting to an instance	22
Statement and transaction processing	24
Key Point Summary	26
Assessment Questions	28
Answers to Chapter Questions	30
Chapter Pre-Test	30
Assessment Questions	32
Chapter 2: Getting Started with Oracle8i Server	35
Oracle Universal Installer	37
Optimal Flexible Architecture (OFA)	42
Privileged Users	43
Authenticating privileged users	45
Oracle Tools for Administration	53
Interactive/command-line administration tools	53
Oracle Enterprise Manager	60
Other graphical administration tools	76
Key Point Summary	78
Assessment Questions	80
Scenarios	83
Lab Exercises	83
Answers to Chapter Questions	85
Chapter Pre-Test	85
Assessment Questions	86
Scenarios	88

Part II: Creating and Administering an Oracle Instance and Database

89

Chapter 3: Managing an Oracle Instance 91

Overview of Starting and Stopping an Oracle Instance	93
Parameter File (INIT.ORA)	94
Required INIT.ORA parameters	96
Other key INIT.ORA parameters	97
Storing saved configurations using Instance Manager	102
Starting Up and Shutting Down an Oracle Instance	103
Startup/shutdown stages	103
The STARTUP command	104
The SHUTDOWN command	108
Dynamic performance views	111
Managing Oracle Instance Settings	115
Displaying parameter values	115
Dynamically changing parameter values	116
Managing User Sessions	119
Managing the ALERT File and Trace Files	122
Key Point Summary	125
Assessment Questions	127
Scenarios	129
Lab Exercises	130
Answers to Chapter Questions	131
Chapter Pre-Test	131
Assessment Questions	132
Scenarios	134
Lab Exercises	135

Chapter 4: Creating a Database 139

Overview of Creating and Managing a Database	141
Creation Prerequisites	142
Operating system considerations	142
Permissions	144
Planning database file locations	144
Creating a Database	149
Using the Oracle Database Configuration Assistant	149
Creating a database manually using the CREATE DATABASE command	165
Troubleshooting Database Creation Problems	178
Key Point Summary	179

Assessment Questions	182
Scenarios	185
Lab Exercises	185
Answers to Chapter Questions	186
Chapter Pre-Test	186
Assessment Questions	187
Scenarios	188
Lab Exercises	189

Chapter 5: Creating Data Dictionary Views and Standard Packages 193

Overview of Data Dictionary Views, Standard Packages, and Database Event Triggers	195
Data dictionary base tables and views	197
Uses of the Oracle data dictionary	199
Data Dictionary View Types	199
DBA_ Views	200
USER_ Views	202
ALL_ Views	204
Special data dictionary views	205
Dynamic performance views	206
Stored Program Units	209
Stored program unit types	209
Benefits of stored program units	210
Stored PL/SQL program units	211
Oracle data dictionary packages	228
View stored object information	230
Creating Data Dictionary Views	232
Scripts to create data dictionary views and packages	232
Other scripts and data dictionary objects	237
The Next Step	239
Key Point Summary	239
Assessment Questions	241
Scenarios	244
Lab Exercises	244
Answers to Chapter Questions	245
Chapter Pre-Test	245
Assessment Questions	246
Scenarios	248
Lab Exercises	248

Chapter 6: Maintaining the Control File 253

Overview of the Control File	255
Control File Contents	256
Creating the Control File	257

Protecting the Control File	258
Multiplexing the control file	259
Backing up the control file	260
Displaying Information about the Control File	261
Views using the control file	263
Key Point Summary	264
Assessment Questions	265
Scenarios	266
Lab Exercises	267
Answers to Chapter Questions	269
Chapter Pre-Test	269
Assessment Questions	270
Scenarios	270

Chapter 7: Maintaining Redo Log Files 273

Overview of the Online Redo Log Files	275
Planning Redo Log Files	277
Redo log file structure	277
Multiplexing redo log files	281
Using Online Redo Log Files	281
Log writer	282
Log switches	282
Checkpoints	283
Controlling log switches and checkpoints	283
Archiving Redo Log Files	287
NOARCHIVELOG mode	287
ARCHIVELOG mode	289
Maintaining Redo Log Files	291
Obtaining information about redo log files and archiving	291
V\$LOGFILE	292
V\$DATABASE	293
V\$INSTANCE	294
ARCHIVE LOG LIST	294
Managing Redo Log Files	295
Adding online redo log groups	295
Adding online redo log members	296
Dropping online redo log groups	296
Dropping online redo log members	298
Moving or renaming online redo log files	299
Clearing online redo log files	300
Using Oracle Enterprise Manager	300
Troubleshooting LGWR Problems	300
Using LogMiner	302
How LogMiner Works	302
Finishing LogMiner sessions	305
Obtaining information about the LogMiner analysis	305
Features and restrictions of LogMiner	306
Key Point Summary	306
Assessment Questions	308

Scenarios	312
Lab Exercises	314
Answers to Chapter Questions	317
Chapter Pre-Test	317
Assessment Questions	318
Scenarios	320
Lab Solutions	321

Chapter 8: Managing Tablespaces and Datafiles 325

The Oracle Database Storage Hierarchy	327
Physical components	328
Logical components	329
Tablespaces	333
Tablespace types	334
Tablespace contents	336
Creating tablespaces	339
Maintaining tablespaces	345
Getting information about tablespaces	359
Guidelines for tablespaces	362
Key Point Summary	363
Assessment Questions	366
Scenarios	368
Lab Exercises	369
Answers to Chapter Questions	372
Chapter Pre-Test	372
Assessment Questions	373
Scenarios	375
Lab Exercises	377

Part III: Administering Storage for Oracle Objects 387

Chapter 9: Storage Structure and Relationships 389

Overview of Logical Storage Structure Components	391
Types of Segments	392
Table	392
Index	392
Cluster	393
Index-organized table	393
Partitions	393
Rollback segment	396
Temporary segment	396
LOB segment	397
Nested table	397
Bootstrap segment	397
Storage Clause Precedence	398
Extents	401
Automatic and manual allocation of extents	401

Block Space Utilization	402
Database block contents	402
Block space utilization parameters	404
Getting Information about Database Storage Structures	406
Planning the Location of Segments	409
Key Point Summary	412
Assessment Questions	414
Scenarios	416
Lab Exercises	417
Answers to Chapter Questions	419
Chapter Pre-Test	419
Assessment Questions	420
Scenarios	421
Lab Exercises	422

Chapter 10: Managing Rollback Segments 425

The Use of Rollback Segments	427
Purpose of Rollback Segments	430
Transaction rollback	430
Transaction recovery	431
Read consistency	431
Types of Rollback Segments	431
SYSTEM rollback segment	432
non-SYSTEM rollback segments	432
DEFERRED rollback segments	433
Rollback Segment Operations	433
How transactions use rollback segments	433
Growth of rollback segments	435
Shrinkage of rollback segments	436
Creating Rollback Segments	438
Guidelines for creating rollback segments	439
Creating rollback segments using OEM Storage Manager	442
Getting Information about Rollback Segments	443
DBA_ROLLBACK_SEGS	443
V\$ROLLNAME	445
V\$ROLLSTAT	445
V\$TRANSACTION	451
Maintaining Rollback Segments	453
Bringing rollback segments online	453
Using OEM to bring rollback segments online	455
Taking rollback segments offline	455
Changing rollback segment storage parameters	459
Changing rollback segment storage parameters using OEM	459
Shrinking rollback segments	460
Dropping rollback segments	462

Troubleshooting Rollback Segments	463
Insufficient space for transactions	463
Read consistency error	464
Blocking session	464
Error in taking a tablespace offline	466
Planning Rollback Segments	467
Online transaction processing environments	467
Data warehousing environments	468
Key Point Summary	468
Assessment Questions	470
Scenarios	472
Lab Exercises	473
Answers to Chapter Questions	476
Chapter Pre-Test	476
Assessment Questions	477
Scenarios	477
Lab Solutions	478
Chapter 11: Managing Tables	485
Data Storage Basics	488
Basic data storage structures	488
Structure of a row	492
Built-in Oracle datatypes	493
Large object datatypes	498
The ROWID and UROWID datatypes	499
Collection types	502
Creating Tables	506
The CREATE TABLE syntax	507
Creating tables using Oracle Schema Manager in OEM	514
Temporary tables	514
Creating tables on clusters	517
Guidelines for creating tables	522
Modifying Tables	530
Altering storage parameters	530
Manually allocating extents	532
Moving tables	534
Handling unused space	537
Truncating tables	541
Dropping a table	542
Dropping a column	542
Retrieving Table Information	545
DBA_OBJECTS	545
DBA_TABLES	546
DBA_SEGMENTS	549
DBA_EXTENTS	551
DBA_TAB_COLUMNS	552
DBMS_ROWID Package	553
Key Point Summary	555

Assessment Questions	557
Scenarios	561
Lab Exercises	563
Answers to Chapter Questions	567
Chapter Pre-Test	567
Assessment Questions	568
Scenarios	570
Lab Solutions	571
Chapter 12: Managing Indexes	579
Index Basics	581
Logical elements of indexes	582
Physical elements of indexes	585
Types of Indexes	585
B-tree indexes	585
How indexes are used by SQL statements	587
Effect of DML on indexes	587
Bitmap indexes	593
Comparing B-tree and bitmap indexes	596
Creating B-Tree Indexes	597
The CREATE INDEX command	597
Creating indexes using OEM	601
Modifying Indexes	604
Altering storage parameters	604
Allocating and deallocating index space	605
Rebuilding indexes	605
Rebuilding indexes online	607
Coalescing indexes	608
Dropping indexes	608
Getting Information on Indexes	609
Data dictionary views	609
Getting index information using OEM	615
Key Point Summary	616
Assessment Questions	618
Scenarios	620
Lab Exercises	621
Answers to Chapter Questions	622
Chapter Pre-Test	622
Assessment Questions	623
Scenarios	624
Lab Exercises	624
Chapter 13: Maintaining Data Integrity	629
Overview of Data Integrity in Oracle	631
Integrity Constraints in Oracle	634
Types of constraints	634
Constraint states	642
When are constraints checked?	647
Special considerations for constraints	649

Implementing Constraints	654
Modifying Constraints	659
Disabled novalidate	659
Disabled validate	660
Enabled novalidate	660
Enabled validate	661
Getting Constraint Information	663
DBA_CONSTRAINTS	663
DBA_CONS_COLUMNS	665
Other helpful queries	666
Key Point Summary	666
Assessment Questions	669
Scenarios	672
Lab Exercises	673
Answers to Chapter Questions	674
Chapter Pre-Test	674
Assessment Questions	676
Scenarios	677
Lab Exercises	678

Part IV: Managing Oracle Data

683

Chapter 14: Loading Data 685

Overview of Loading Data into Oracle Databases	687
Loading Data Using Direct-Load INSERTs	688
NOLOGGING option	690
Parallel direct-load INSERTs	691
Pros and cons of direct-load INSERTs	695
Loading Data Using SQL*Loader	696
Files used by SQL*Loader	697
Invoking SQL*Loader from the command line	702
Invoking SQL*Loader using Oracle Enterprise Manager	705
Comparing conventional and direct-path loads	711
Guidelines for Using SQL*Loader	714
Troubleshooting SQL*Loader	715
Key Point Summary	717
Assessment Questions	719
Scenarios	721
Lab Exercises	722
Answers to Chapter Questions	723
Chapter Pre-Test	723
Assessment Questions	724
Scenarios	725
Lab Exercises	726

Chapter 15: Reorganizing Data	731
Overview of Reorganizing Data	733
Moving Data	735
The create table as select (CTAS) method	735
The alter table method	737
The Import and Export utilities	738
Export modes	738
Moving data scenarios	743
Export types	744
Exporting Data Using the Export Utility	745
Interactive mode	749
Command Line mode	751
Exporting using Oracle Enterprise Manager	753
Importing Data Using the Import Utility	756
Interactive mode	760
Command Line mode	762
Importing using Oracle Enterprise Manager	770
Import behavior	772
Export and import guidelines	775
Transportable Tablespaces	776
Implementing Transportable Tablespaces	777
Transportable Tablespace uses and guidelines	779
Checking for self-contained tablespaces	780
Key Point Summary	782
Assessment Questions	784
Scenarios	787
Lab Exercises	787
Answers to Chapter Questions	789
Chapter Pre-Test	789
Assessment Questions	790
Scenarios	790
Lab Exercises	792

Part V: Managing Oracle Security

797

Chapter 16: Managing Password Security and Resources	799
Overview of Oracle Security	801
Profiles	803
Overview of profiles	804
Profile settings	805
Creating and managing profiles	813
Key Point Summary	827
Assessment Questions	828
Scenario	832

Lab Exercises	832
Answers to Chapter Questions	834
Chapter Pre-Test	834
Assessment Questions	836
Scenarios	837
Lab Exercises	839
Chapter 17: Managing Users	845
Overview of User Management	847
Users and schemas	847
Guidelines for Creating Users	849
Authentication mechanisms	849
Quotas	850
Tablespace assignment	852
Other user creation guidelines	852
Managing Users	853
Creating users	853
Modifying users	863
Dropping users	866
Getting Information about Users	869
Key Point Summary	871
Chapter Assessment	873
Scenarios	876
Lab Exercises	877
Answers to Chapter Questions	879
Chapter Pre-Test	879
Assessment Questions	880
Scenarios	881
Lab Exercises	883
Chapter 18: Managing Privileges	891
Overview of Managing Privileges	893
Types of Privileges	894
System privileges	894
Object privileges	914
Auditing Privilege Usage	932
Types of auditing	933
Database auditing guidelines	934
Implementing database auditing	936
Getting information about auditing	946
Key Point Summary	952
Assessment Questions	954
Scenarios	957
Lab Exercises	958

Answers to Chapter Questions	959
Chapter Pre-Test	959
Assessment Questions	960
Scenarios	962
Lab Exercises	964

Chapter 19: Managing Roles 973

Overview and Benefits of Roles	975
Characteristics of roles	975
Benefits of roles	976
Implementing and Using Roles	978
Pre-defined roles in Oracle8i	978
Creating roles	982
Managing privileges with roles	986
Modifying Roles	990
Managing roles for users	991
Querying role information in the data dictionary	999
Guidelines for using roles	1003
Key Point Summary	1005
Assessment Questions	1006
Scenario	1009
Lab Exercise	1009
Answers to Chapter Questions	1010
Chapter Pre-Test	1010
Assessment Questions	1011
Scenario	1012
Lab Exercise	1013

Part VI: National Language Support

1021

Chapter 20: Using National Language Support 1023

Overview of National Language Support (NLS)	1025
Choosing a Character Set and National Language Character Set for Your Database	1026
Character sets and encoding schemes	1027
Character sets and datatypes	1029
Guidelines for choosing a character set and NLS Character set	1030
Specifying Language-Dependent Behavior in Oracle	1032
INIT.ORA parameters for configuring NLS behavior	1032
Using the NLS_LANG environment variable to configure NLS settings	1035
Changing NLS settings for a session	1036
Using NLS in Oracle8i	1038
Sorting	1038
Using NLS parameters in SQL functions	1042
Using Import/Export, SQL*Loader, and NLS	1044

Getting Information about NLS Settings	1045
Key Point Summary	1050
Assessment Questions	1052
Scenario	1055
Lab Exercises	1056
Answers to Chapter Questions	1057
Chapter Pre-Test	1057
Assessment Questions	1058
Scenario	1059
Lab Exercises	1060
Appendix A: What's on the CD-ROM	1065
Appendix B: Practice Exam	1071
Appendix C: Objective Mapping	1095
Appendix D: Exam Tips	1103
Appendix E: Database Schema for Labs	1107
Appendix F: Suggested Readings, Web Sites, and Other Resources	1125
Index.	1129
End-User License Agreement	1168

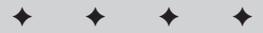
Overview of Oracle Architecture and Administration

As is true with most things, it is always good to start at the beginning. This first part of the book does exactly that by introducing you to the architectural elements of Oracle so that you have a better sense of what components make up an Oracle database and instance. You then find out what tools are available to administer an Oracle database.

Chapter 1 provides a pretty comprehensive overview of the architectural components that make up Oracle. You will learn what the difference between an Oracle database and an Oracle instance is. You will also find out what components make up an Oracle instance from shared memory structures to background processes. You will then learn what is meant by the term *Oracle database* and which files are part of a database. Then other key files will be presented, followed by a discussion on how a SQL statement is processed and what components of Oracle are involved.

Chapter 2 will deal more with administration and a lot less with architecture. The different tools available to administer an Oracle instance and database will first be introduced to you. This will be followed by a discussion on what a privileged user is and why this is so important in the administration of Oracle. You will also learn how to grant a user the necessary permissions to be a privileged user and which users have them when a database is first created.

As mentioned at the outset, this part presents the 40,000-foot view of Oracle. All of the information you are introduced here is explained in much more detail in the parts of the book that follow.



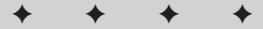
In This Part

Chapter 1

Oracle8i Architecture

Chapter 2

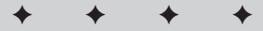
Getting Started with
Oracle8i Server



Oracle8i Architecture

EXAM OBJECTIVES

- ◆ Oracle Architectural Components
 - Describe the Oracle server architecture and its main components
 - List the structures involved in connecting a user to an Oracle instance
 - List the stages in processing queries, DML statements, COMMITS



CHAPTER PRE-TEST

1. List the various components of the Oracle database.
2. What does the term “Oracle instance” refer to?
3. How many required background processes are there in Oracle8i, what are they, and what does each do?
4. Which optional background process is strongly recommended to ensure good database recoverability and what does it do?
5. If a user issues a COMMIT command to end a transaction, what must take place before the user is notified that the transaction has been committed?
6. Which Oracle file provides information on the configuration of the instance and what naming convention does it follow?
7. On which computer does the user process typically reside? How about the server process?
8. Can more than one instance access data in a single Oracle database at the same time?
9. What is stored in the shared pool?
10. Which Oracle database files are written to most often?
11. What are the three phases of processing an SQL statement?

The first phase of being able to pass the “Oracle8i: Architecture and Administration” exam is to understand, at a relatively high level, how Oracle works. Just as a pilot needs to understand the basics of flight and how an airplane is put together, a database administrator (DBA) of an Oracle database needs to understand what parts make up both the Oracle instance and the database, how these are configured, and how client applications access the database to query and modify data. However, the pilot of an airplane is not the person who has all the answers. Where an aircraft mechanic has more knowledge about the inner workings of the jet engine, for example, the database administrator does not necessarily need to know intimate details about how the different components of Oracle work but does need to understand the interrelationships between them.

The goal of this first chapter is to provide you with the basics of the Oracle architecture and how Oracle processes SQL statements. This information is built upon throughout the remainder of the book, so this is not to be a detailed discussion but more of a 40,000-foot view.

Oracle Architectural Overview



Describe the Oracle server architecture and its main components

The whole point of a relational database management system (RDBMS) is to store and supply data to clients who request it. Each RDBMS does this in its own way, so knowing how one works does not necessarily mean that you can figure out the rest. Oracle, being one of the more sophisticated RDBMSs out there, allows for a great deal of flexibility in its configuration and operation. Part of this is possibly due to its architectural design.

The architecture of Oracle is configured in such a way as to ensure that client requests for data retrieval and modification are satisfied efficiently while maintaining database integrity. The architecture also ensures that, should parts of the system become unavailable, mechanisms of the architecture can be used to recover from such failure and, once again, bring the database to a consistent state, ensuring database integrity. Furthermore, the architecture of Oracle needs to provide this capability to many clients at the same time so performance is a consideration when the architecture is configured.

In understanding the Oracle architecture and how it is used to process client requests, several terms need to be introduced. Figure 1-1 displays a diagram of the architectural components that make up a typical Oracle configuration. The terms shown in the diagram and briefly explained below deal with shared memory structures, processes, and datafiles that are used by Oracle. The terms that you will read about in more detail in this chapter and that are referred to throughout the book include:

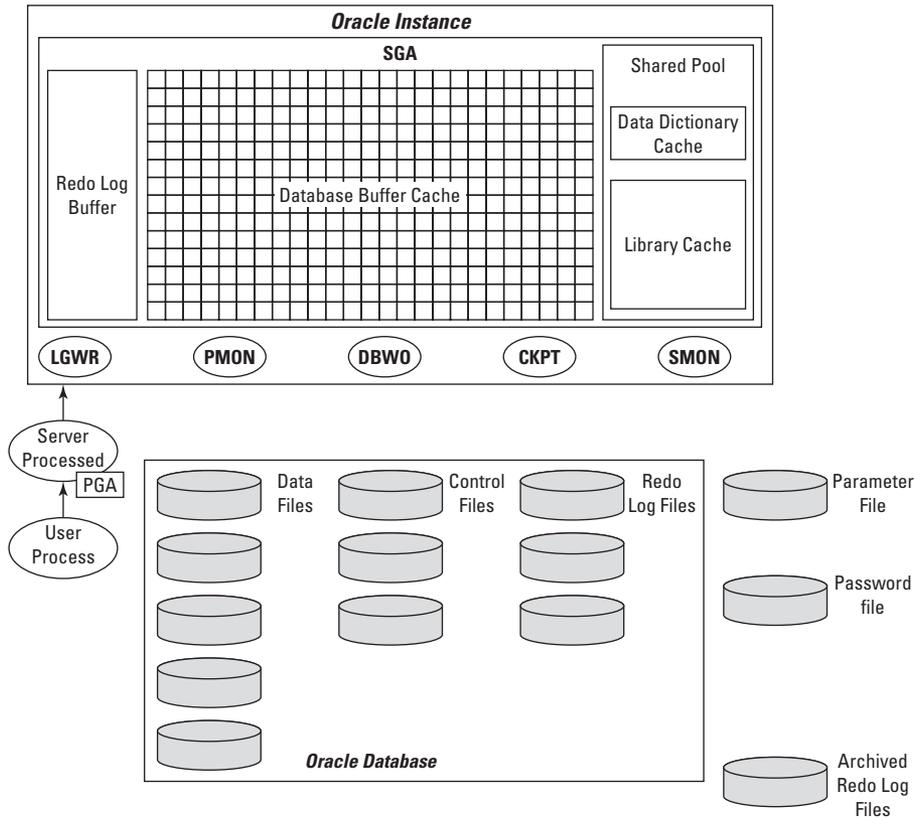


Figure 1-1: Oracle's architectural components include the instance, database, and other files and processes.

- ♦ **Oracle Instance** — The instance is a combination of a memory structure shared by all clients accessing the data, and a number of background processes that perform actions for the instance as a whole.

The shared memory structure is called the SGA, which stands for System Global Area or Shared Global Area, depending on who you ask. Either term is equally acceptable and the acronym SGA is the most common way to refer to this memory structure.

Oracle also includes a number of background processes that are started when the instance is started. These include the database writer (DBW0), system monitor (SMON), process monitor (PMON), log writer (LGWR), and checkpoint process (CKPT). Depending on the configuration of your instance and your requirements, others may also be started. An example of this is the archiver process (ARC0), which will be started if automatic archiving of log files is turned on.

- ♦ **Oracle Database**—The database consists of three types of files. Datafiles, of which there can be many depending on the requirements of the database, are used to store the data that users query and modify. The control file is a set of one or more files that keeps information about the status of the database and the data and log files that make it up. The redo log files are used to store a chronological record of changes to the data in the datafiles.



More information is provided about the control file in Chapter 6. Chapter 7 provides additional insight into redo log files, while datafiles are covered in more detail in Chapter 8.

- ♦ **User Process**—The user process is any application, either on the same computer as the database or on another computer across a network that can be used to query the database. For example, one of the standard Oracle query tools is SQL*Plus—a user process. Another example of a user process is Microsoft Excel or an Oracle Financials application. Any application that makes use of the database to query and modify data in the database is considered a user process. The user process does not have to come from Oracle Corporation; it only needs to make use of the Oracle database.
- ♦ **Server Process**—The server process is a process launched when a user process makes a connection to the instance. The server process resides on the same computer as the instance and database and performs all of the work that the user process requests. As you will find out in more detail later in this chapter, the server process receives requests from the user process in the form of SQL commands, checks their syntax, executes the statements and returns the data to the user process. In a typical Oracle configuration, each user process will have a corresponding server process on the Oracle server to perform all the work on its behalf.
- ♦ **Other Oracle Files**—The remainder of the files that can be found on the hard disk of the server computer include the Oracle parameter files (also known as the INIT.ORA file), which has information on how the instance should be configured and how operations on the database will be performed/optimized, the password file, which is used to authenticate privileged users that can stop and start the instance, as well as perform other actions, and the archived redo log files, which are a copy of the redo log file that has been filled up. Archived redo log files exist to help the DBA ensure good recoverability of the database.



The parameter file is covered in more detail in Chapter 3 while the password file is explained in Chapter 2.

Oracle instance

As shown in Figure 1-2, the Oracle instance is made up of a shared memory structure (the SGA), which is composed of a number of distinct memory areas. The other part of the instance is the set of background processes, both required and optional, that perform work on the database.

The instance is always associated with one, and only one, database. This means that when an instance is started, the `DB_NAME` parameter in the Oracle parameter (`INIT.ORA`) file specifies which database the instance will be connected to, while the `INSTANCE_NAME` parameter (which defaults to the value of the `DB_NAME` parameter) specifies the name of the instance. The configuration of the instance is always performed through parameters specified in the `INIT.ORA` file and one environment variable—`ORACLE_SID`—which is used to determine which instance to start and perform configuration operations on when on the same server as the database.

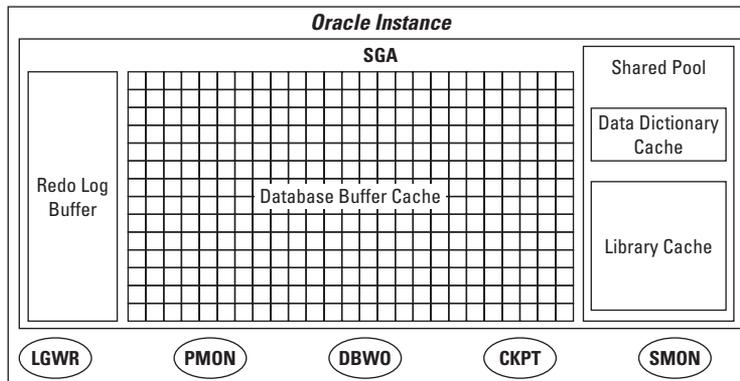


Figure 1-2: The Oracle instance is composed of a number of shared memory areas and background processes.

One of the main objectives of the instance is to ensure that connections by multiple users to access database data are handled as efficiently as possible. One way it accomplishes this is by holding information in the datafiles in one of its shared memory structures—the database buffer cache—to allow multiple users reading the same data to retrieve that data from memory instead of disk since access to memory is about a thousand times quicker than access to a disk file.

Another reason that the instance is important is that, when multiple users access Oracle data, allowing more than one to make changes to the same data can cause data corruption and cause the integrity of the data to become suspect. The instance facilitates locking and the ability for several users to access data at the same time.



Tip

It is important to remember that a user process, when attempting to access data in the database does not connect to the database but to the instance. When specifying what to connect to from a user process, you always specify the name of the instance and not the name of the database. The instance, in this way, is the gatekeeper to the database. It provides the interface to the database without allowing a user to actually touch the various files that make up the database.

System Global Area (SGA)

The SGA is a shared memory structure that is accessed by all processes in order to perform database activity, such as read and write data, log changes to the log files, and keep track of frequently executed code and data dictionary objects. The SGA is allocated memory from the operating system on which the instance is started, but the memory that is allocated to it is managed by various Oracle processes. The SGA is composed of several specific memory structures, as shown in Figure 1-2. These include:

- ♦ **Shared Pool**—The *Shared Pool* is an area of SGA memory whose size is specified by the INIT.ORA parameter SHARED_POOL_SIZE. The default value for SHARED_POOL_SIZE is 3,000,000 bytes, or just under 3MB in versions of Oracle prior to 8.1.7, and 8,000KB bytes, or just under 8MB in Oracle 8.1.7. The size of the shared pool remains constant while the instance is running and can only be changed by shutting down and restarting the instance, after modifying the value in the INIT.ORA file.



For information on how to configure, start, and stop an Oracle instance, please refer to Chapter 3.

The shared pool is divided into two main areas of memory—the *data dictionary cache* (also called the *dictionary cache* or *row cache*) and the *library cache*. The data dictionary cache is used to store a cached copy of information on frequently accessed data dictionary objects. The information cached includes the name of the object, permissions granted on the object, dependency information, and so on. The data dictionary cache also includes information on the files that make up the database and what tablespaces they belong to, as well as other important information.

When a server process needs to determine what the name “Students” refers to, it queries the data dictionary cache for that information and, if the information cannot be found, it reads the information from the datafile where the data dictionary is located and then places it in a cache for others to read. The information in the cache is stored using a least-recently-used (or LRU) algorithm. This means that information that is frequently requested remains in the cache while information that is only occasionally required is brought into the cache and flushed out if space is required to bring other information in.

You cannot manually size the data dictionary cache—Oracle does this dynamically and automatically. If more memory is required to cache data dictionary information, which may be the case in a database with many objects, the cache is made larger to accommodate the requests. If the memory is needed by the library cache component of the shared pool, some memory may be freed up and removed from the data dictionary cache.

The other major component of the shared pool is the library cache. The library cache is used to store frequently executed SQL statements and PL/SQL program units such as stored procedures and packages. Storing the parsed statement along with the execution plan for the commands sent to the server

in memory allows other users executing the same statement (that is, identical in every way including case of statement text, spaces and punctuation) to reduce the work required by not having to re-parse the code. This improves performance and allows the code to appear to run quicker.

The library cache is broken up into a series of memory structures called shared SQL areas that store three elements of every command sent to the server: the text of the SQL statement or anonymous PL/SQL block itself, the parse tree or compiled version of the statement, and the execution plan for the statement that outlines the steps to be followed to perform the actions for the SQL statement or PL/SQL block.

Each shared SQL area is assigned a unique value that is based upon the hash calculated by Oracle from the text of the statement, the letters and case thereof in it, spacing, and other factors. Identical statements always hash out to the same value whereas different statements, even returning the same result, hash out to different values. For example, the following two statements use two shared SQL areas because their case is different, though they both return the same information:

```
SELECT * FROM DBA_USERS;  
select * from dba_users;
```

One of the goals for ensuring good performance of the applications accessing data in the database is to share SQL areas by ensuring that the statement returning the same result are identical, thereby allowing each subsequent execution following the first of the same SQL statement to use the execution plan created the first time a command is run. The preceding two statements are considered inefficient because they would need to allocate two shared SQL areas to return the same results. This would consume more memory in the shared pool (once for each statement), as well as cause the server process to build the execution plan each time.

Like the data dictionary cache, the library cache also works on an LRU algorithm that ensures that statements that are frequently executed by users remain in the cache while those executed infrequently or just once be aged out when space is required. Also like the data dictionary cache, you cannot specifically size the library cache — Oracle sizes it automatically based upon the requirements of the users and statements sent to the server, as well as memory allocated to the shared pool with the `SHARED_POOL_SIZE` parameter.

- ♦ **Database Buffer Cache** — The *database buffer cache* is used to store the most recently used blocks from the datafiles in memory. Because Oracle does not allow a server process to read data from the database directly before returning it to the user process, the server process always checks to see if a block it needs to read is in the database buffer cache, and, if so, retrieve it from the cache and return the rows required to the user. If the block the server process needs to read is not in the database buffer cache, it reads the block from the datafile and places it in the cache.

The database buffer cache also uses an LRU algorithm to determine which blocks should be kept in memory and which can be flushed out. The type of access can also have an impact on how long a block from the datafile is kept in the cache. If the situation where a block is placed in the cache as a result of an index lookup, the block is placed higher in the list of blocks to be kept in the cache than if it were retrieved as a result of a full table scan, where every block of the table being queried is read. Both placing the datafile blocks in the database buffer cache in the first place, and their importance in being kept there for a long or short period, is designed to ensure that frequently accessed blocks remain in the cache. If a full table scan of a large table caused each block read during the full table scan to flush another block out of cache, a single table scan of a single table would cause all blocks to be flushed—hence the distinction.

The database buffer cache is sized by a couple of Oracle initialization parameters. The `INIT.ORA` parameter `DB_BLOCK_SIZE` determines the size, in bytes, of each block in the database buffer cache and each block in the datafile. The value for this parameter is determined when the database is created and cannot be changed. Essentially, each block in the database buffer cache is exactly the same size, in bytes, as each database block in the datafiles. This makes it easy to bring datafile blocks into the database buffer cache—they are the same size. The default for `DB_BLOCK_SIZE` is 2048KB—too small in almost all cases.



For a discussion on factors that can help determine the proper setting of `DB_BLOCK_SIZE`, please refer to Chapter 4.

The other `INIT.ORA` parameter that is used to determine the size of the database buffer cache is `DB_BLOCK_BUFFERS`. This parameter defaults to 50, which is also its minimum value. The total amount of memory that will be used for the database buffer cache is `DB_BLOCK_BUFFERS * DB_BLOCK_SIZE`. For example, setting `DB_BLOCK_BUFFERS` to 2,000 when the `DB_BLOCK_SIZE` is 8,192, allocates 2000×8192 or 16MB of RAM for the database buffer cache. When sizing the database buffer cache, you need to consider the amount of physical memory available on the server and what the database is being used for. If users of the database are going to make use of full table scans, you may be able to have a smaller database buffer cache than if they frequently accessed the data with index lookups. The right number is always the balance between a sufficiently high number so that physical reads of the datafiles are minimized and not too high a number so that memory problems take place at the operating system level. How to arrive at this is the subject of the “Oracle8i: Performance and Tuning” exam and not one covered in this book.

- ♦ **Redo Log Buffer**—Before a change to a row or the addition or removal of a row from a table in the database is recorded to the datafiles, it is recorded to the redo log files and, even before that, to the *redo log buffer*. The redo log buffer is essentially a temporary storage area for information to be written to the redo log files.



For information on how redo log files work and the best way to configure them, please refer to Chapter 7.

When a server process needs to insert a row into a table, change a row, or delete a row, it first records the change in the redo log buffer so that a chronological record of changes made to the database can be kept. The information, and the size of the information, that will be written to the redo log buffer, and then to the redo log files, depends upon the type of operation being performed. On an INSERT, the entire row is written to the redo log buffer because none of the data already exists in the database, and all of it will be needed in case of recovery. When an UPDATE takes place, only the changed columns' values is written to the redo log buffer — not the entire row. If a DELETE is taking place, then only the ROWID (unique internal identifier for the row) is written to the redo log buffer, along with the operation being performed. The whole point of the redo log buffer is to hold this information until it can be written to the redo log file.

The redo log buffer is sized by the INIT.ORA parameter LOG_BUFFER. The default size depends on the operating system that is being used but is typically four times the largest operating system block size supported by the host operating system. It is generally recommended that the LOG_BUFFER parameter be set to 64KB in most environments since transactions are generally short.

The redo log buffer is a circular buffer, which means that entries have been written to the redo log file and the space occupied by those entries can be used by other transactions. This is possible because the log writer (LGWR) background process flushes the contents of the redo log buffer to the redo log files whenever a commit occurs or whenever any one transaction occupies more than one-third of the space in the buffer. This essentially means that the redo log files are the most write-intensive files in a database and that the redo log buffer can also be kept relatively small and still be efficient.

While the database buffer cache, shared pool, and redo log buffer are a required part of the SGA, the SGA also may have additional shared memory areas, based upon the configuration of the instance. Two of the most common additional shared memory structures are the large pool and the Java pool.

The large pool is sized by the INIT.ORA parameter LARGE_POOL_SIZE whose minimum value is 0 but may actually allocate a different amount of memory based upon the values of other INIT.ORA parameters such as PARALLEL_AUTOMATIC_TUNING. The large pool is used for memory structures that are not directly related to the processing of SQL statements. An example of this is for holding blocks in memory when a backup or restore operation through Oracle Recovery Manager (RMAN) is taking place. Another use of the large pool is for sort space in a multi-threaded server (MTS) environment.



For more information on the large pool and multi-threaded server (MTS) see the *Oracle8i Concepts* manual in the Oracle documentation set. The *Oracle8i Backup and Recovery Guide* contains information about Oracle Recovery Manager (RMAN) and its use of the large pool.

The Java pool is used to store Java code and its execution plan. It is used for Java stored procedures and functions and other classes that you have created that will be run in the Java virtual machine (JVM) that resides on the Oracle server. The minimum value for `JAVA_POOL_SIZE` is 0, but Oracle will always allocate a minimum of 32,768 bytes in Oracle 8.1.6 or higher (in Oracle 8.1.5 the minimum value was 1,000,000 bytes and setting the parameter `JAVA_POOL_SIZE` to a value less than that would generate an error). The default value, if not set in the `INIT.ORA` file, is 20,000KB or 20MB.

Background processes

Aside from the shared memory structures, the SGA also includes a number of background processes that perform actions that deal with database operations for all users. You have already been introduced to the log writer and how it is used to write information from the redo log buffer to the redo log files. Other background processes exist as well to perform different actions. In detailing these, you need to distinguish between required and optional background processes.

Oracle8i has five required background processes. If any of these processes are killed or terminate for any reason, the instance is considered to have crashed and instance recovery (that is, stopping and re-starting the instance), and even database recovery may be needed. The required background processes, as shown in Figure 1-2, are the following:

- ♦ **SMON**—*System Monitor (SMON)* is a background process that does exactly what you would expect—it monitors the health of the instance as a whole and ensures that the data in the database is consistent. To ensure database and instance integrity, when you start the instance before the database is opened and available to users, SMON rolls forward any committed transactions found in the redo log files and rolls back any uncommitted transactions. Because all changes to the data in the database are recorded to the redo log buffer and then the redo log files, this means that anything that has taken place before the instance crashed is properly recorded to the datafiles. Once this is completed, the database will be opened and its data available to users for querying and modification.



During instance startup SMON only actually performs a roll forward before the database is opened. Rollback of uncommitted transactions takes place after the database is opened and before users access any data blocks requiring recovery. This is known as “delayed transaction rollback” and is there to ensure that the database can be opened as quickly as possible so that users can get at the data. Before any data that was not committed prior to the instance going down is queried or modified, SMON ensures that a rollback takes place to bring it to a consistent state.

SMON also does some clean-up work by coalescing any free space in the datafiles to make it contiguous. When a table, index, or other object that requires storage is dropped or truncated, this frees up the space that was previously used by that object. Because a single object could be made up on many sets of database blocks called extents, and these extents could be of different sizes, SMON coalesces (or combines) these extents into larger chunks of free disk space in the datafiles so that they may be allocated to other objects, if needed. The reason for this is that if the extents were left at their original size, a CREATE TABLE statement may fail if it cannot find an extent of the size it requested — while free space would still exist in the datafile.



For more information on tablespaces, datafiles, segments, and extents see Chapter 8 and Chapter 9.

As well as coalescing free space, SMON also de-allocates temporary segments in datafiles that belong to permanent tablespaces to ensure that they do not occupy space required by tables, indexes, or other permanent objects. Temporary segments are created when a sort being performed cannot completely be performed in memory and disk space is needed as a temporary storage area. When this disk space is created on a tablespace holding other permanent objects such as tables, indexes, and the like, SMON needs to get rid of the temporary segment as quickly as possible to ensure that a table or index does not run out of disk space if the space is needed.

- ♦ **PMON** — *Process Monitor (PMON)* is a required background process that does exactly what the name says — it monitors server and background processes to ensure that they are operating properly and have not hung or been terminated. If a server process dies unexpectedly, it is up to PMON to rollback any transaction that the process was in the middle of, release any locks that the process held, and release any other resources (such as latches) that the process may have held. When an Oracle server process dies, these actions are not automatically performed — it is up to PMON to do so.

You should note that PMON might not perform these actions immediately after the process has terminated. In the first place, how does PMON know for sure that a process is hung and not just sitting idle? For example, a user could be connected to the instance through SQL*Plus and then decide to go for lunch while still connected. Should his or her process be terminated? Perhaps, but generally PMON cannot make that decision. It waits for a clue that the process is no longer doing even simple things, such as communicating its presence. The time PMON may wait to determine this can be quite lengthy and cause others to be locked out from portions of the database. PMON will eventually safely rollback the transaction, release locks, and clean up other resources. In other words — you may have to wait a while.

- ♦ **DBW0** — The *Database Writer (DBW0)* is a background process that performs a very specific task — it writes changed data blocks (also known as *dirty buffers*) from the database buffer cache to the datafiles. Whenever a change is made to data on a block in the database buffer cache, the buffer where the

change was made is flagged for writes to the datafile. The database writer process, of which there could be several, writes the changed blocks to the datafiles whenever a checkpoint occurs, or at other pre-defined intervals.

DBW0 does not write to the datafiles with the same frequency that LGWR writes to the redo log files. The main reason for this is that the minimum size of a write to the datafiles is `DB_BLOCK_SIZE`, which is at least 2,048 bytes. LGWR writes to the redo log files no matter how much information needs to be recorded, and it could be as little as 30 bytes, if only a single change of a small column is made in a transaction. Therefore DBW0 writes are more expensive, in terms of disk I/O than LGWR writes. Because DBW0 writes are more expensive, they are bunched and take place when one of the following takes place:

- **The number of dirty buffers exceeds a pre-defined threshold.** The threshold level is different for each operating system and dependent on the values of other `INIT.ORA` parameters but essentially is designed to ensure that a process can always find a clean (that is, stable and already written to the hard disk) buffer in the cache to write its information to. It is also preferable that the clean buffer be cold — not accessed for a longer period of time and therefore lower on the LRU list of buffers.
- **A checkpoint has taken place.** A checkpoint can be initiated manually by the DBA issuing the command `ALTER SYSTEM CHECKPOINT` or automatically by a redo log file group being filled up and LGWR needing to switch to a different redo log file group. In either case, a checkpoint forces DBW0 to write all dirty buffers to disk to ensure that at the time a checkpoint takes place all the datafiles are consistent with each other. A checkpoint also causes the `CKPT` process to update all datafile headers and the control files to ensure that the information is consistent across the database.
- **A server process cannot find a free buffer.** Each time a server process needs to bring data into the database buffer cache, it scans the LRU list of buffers to determine if a clean free block exists. If it cannot find one after scanning a pre-defined number of buffers, this will trigger DBW0 to write all dirty buffers to the datafiles and thereby create clean buffers that the process can use for its information.
- **A timeout occurs.** Oracle ensures that a write takes place to the datafiles every three seconds, if needed so that the datafiles do not become too out-of-sync with the information in the redo log files and the database buffer cache. It also helps to ensure that changes to the data are written to the database ahead of checkpoints so that less information needs to be written to the disk at that point in time.

In a typical Oracle8i instance, the number of database writers is set to one and others are not started. If you have many hard disks and datafiles in your database, you may want to increase the number of database writer processes. The Oracle initialization parameter `DB_WRITER_PROCESSES` is used to configure the number of database writers. Its default value is 1, which starts a process called DBW0. Increasing the value of `DB_WRITER_PROCESSES` starts

additional database writers up to a maximum of 10—DBW1 to DBW9. You should only use more database writers if your system is write-intensive and you have datafiles on several disks.

- ♦ **LGWR**—The *Log Writer* process, as mentioned previously, writes data from the redo log buffer to the redo log files at defined thresholds. It performs this whenever an implicit or explicit commit occurs; when more than 1MB of data exists in the redo log buffer; when the log buffer is more than one-third full with data from a transaction (that is, when a single transaction occupies a large portion of the redo log buffer); when DBW0 needs to write to the datafiles, forcing LGWR to write out changes from the redo log buffer to the redo log files beforehand; or when three seconds have elapsed since the last LGWR write. For any instance there is only one LGWR process and this number cannot be changed.



Of the times that LGWR writes to the redo log files, the most common reason for doing so is that a commit has taken place. The most unlikely reason for an LGWR write is that the redo log buffer contains more than 1MB of changes. Very few databases have a need for a large redo log buffer and having one that is extremely large can leave you open to losing a large number of changes and being unable to fully recover the database in case of instance failure.

One important element to keep in mind is what actually happens when a transaction commits. When the user issues the COMMIT statement to commit the transaction, LGWR is instructed to flush data in the redo log buffer to the redo log files. If for any reason LGWR is unable to write to the redo log files, the transaction cannot commit and will be rolled back. Oracle requires that a physical record (that is, a write to a disk file) exist in order for the transaction to be considered committed. If the physical record cannot be created (that is, a write to the redo log files cannot take place because the redo log file is unavailable due to a disk crash or other occurrence), the commit cannot complete and the user is notified that the transaction was aborted.

- ♦ **CKPT**—The *Checkpoint* process is responsible for one thing—updating control files and datafiles whenever a checkpoint takes place. Checkpoints can take place when a DBA issues the command ALTER SYSTEM CHECKPOINT, when a redo log file group fills up and the LGWR initiates the checkpoint, or when the values specified by the INIT.ORA parameters LOG_CHECKPOINT_INTERNAL, LOG_CHECKPOINT_TIMEOUT, or FAST_START_IO_TARGET are exceeded.

Prior to Oracle8 the CKPT process was optional and not required. Whenever a checkpoint occurred, LGWR would update datafile headers and control files. However, this would also mean that LGWR could not write to the redo log files at the same time as it was performing checkpoint updates, which slowed the system down. For this reason, the CKPT process was made a required background process in Oracle8.

The required background processes, along with the SGA, provide the basic functionality required for an Oracle instance to operate. However, depending on the configuration of your database and what Oracle options you are using, additional background processes can also be started. Some of the more common optional background processes include:

- ♦ **ARC0** — The *Archiver Process* is used to write redo log files that have been filled and switched from to one or more archive log destinations. In Oracle8i, unlike previous versions, you can configure multiple archiver processes using the INIT.ORA parameter LOG_ARCHIVE_MAX_PROCESSES, which defaults to one. If you specify a large number of archive log destinations and have configured archiving for the instance, having more than one process can improve performance.



Some information on archiving and the archiver process is presented in Chapter 7. For a more detailed discussion, refer to the *Oracle8i Backup and Recovery Guide* in the Oracle8i documentation set.



Because archiving really deals with backup and recovery issues, it is not covered in this book in great detail, nor will you be asked questions on how it works when taking the "Oracle8i: Architecture and Administration" exam. You do need to know what it is at a basic level (see Chapter 7).

- ♦ **Snnn** — The *Shared Server Process* is used in a multi-threaded server (MTS) environment to process requests from database users. Unlike in a typical dedicated server configuration where each user process is assigned a dedicated server process to perform work on its behalf, a multi-threaded server configuration shares the server processes among all the user processes, thereby making the use of resources on the computer more efficient. The number of shared server processes is configured by the INIT.ORA parameters MTS_SERVERS (which defaults to 0) and MTS_MAX_SERVERS (which defaults to 20). If MTS_SERVERS is set to 0, the multi-threaded server is not configured on the instance and no shared server processes are launched.
- ♦ **Dnnn** — The *Dispatcher Process* is also used in an MTS configuration. When a request to initiate an MTS connection to the instance is received from a user process, the dispatcher is the one that is assigned to the process. The same dispatcher can be used to service requests from many users and passes those requests to a queue where they are picked up by a shared server process and executed. The results are then placed in a queue for the dispatcher that requested it. The dispatcher picks up the results and transmits them to the client. The dispatcher process acts somewhat like Louie in the '80s television series "Taxi" — it gets calls from clients (user process) and then dispatches the request to a taxi (shared server) that ferries the client (does the work) and then notifies the dispatcher that it is done (places the request in the queue to be picked up and sent to the user process).

The configuration of dispatchers is performed by setting the INIT.ORA parameters MTS_DISPATCHERS (default of 0) and MTS_MAX_DISPATCHERS (default of 5).



The *Oracle8i Concepts* manual provides more information on multi-threaded server and the use of shared server and dispatcher processes.

- ♦ **LCK0** — The *Lock Process* provides inter-instance locking between nodes in an Oracle Parallel Server environment. When using Oracle Parallel Server, more than one instance can access the same database. When users connected to one instance need to allocate locks to change data in the database, the LCK0 process ensures that a user connected to another instance does not already have the lock before allowing the requesting user to get it. This is done to ensure that data in the database remains consistent at all times, even when accessed by more than one instance.
- ♦ **RECO** — The Recoverer Process is used in distributed database environments and only when the DISTRIBUTED_TRANSACTIONS Oracle initialization parameter is set to a value higher than zero, the default. In this situation it will be started automatically and will be responsible for resolving any failed distributed transactions with data residing on other nodes in the distributed database configuration. Essentially it makes sure that all databases in a distributed transaction are in a consistent state and that a distributed transaction does not commit on one node while not on another.



For more information on Oracle Parallel Server and the LCK0 process, refer to the *Oracle8i Parallel Server Concepts* manual and the *Oracle8i Parallel Server Setup and Configuration Guide*. Distributed database information and the Recoverer Process can be found in the *Oracle8i Distributed Database Systems* manual.

Oracle database

An Oracle instance by itself only provides the mechanism to access the database. If the database is not created, having the instance started allows you to create it. However, to get any real use out of the instance, the database must exist. The database is where the data is stored — both the metadata (data about data, also known as the data dictionary) and user data, such as the Orders table or Customers table or LastName index.

An Oracle database is composed of a series of operating system disk files, as shown in Figure 1-3. The three key types of files that Oracle uses are datafiles, control files, and redo log files. Each file type is required for a database to operate properly and the loss of one or more of the files belonging to the database usually requires that recovery be initiated.

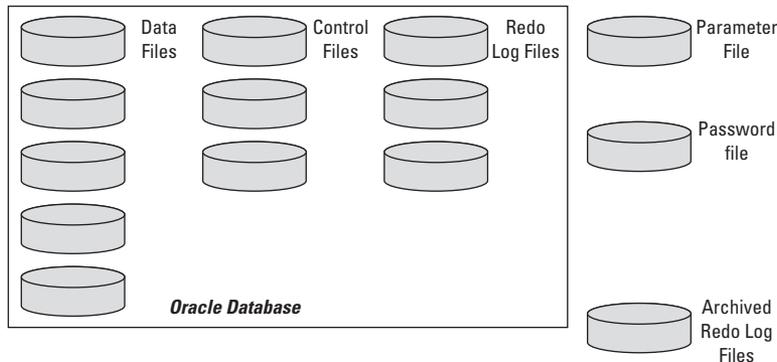


Figure 1-3: The Oracle database consists of datafiles, control files, and redo log files. Other files are also used by Oracle, but are not considered part of the database.

Datafiles

Oracle datafiles contain the actual information that the database stores. When you create a table or index, it is stored in a datafile (a physical element of storage) that belongs to a tablespace (a logical element of storage). Datafiles store anything that is considered a segment in Oracle, such as tables, indexes, clusters, partitions, large objects (LOBs), index organized tables (IOTs), rollback segments, and temporary segments. Anything that requires storage in the database—whether created by the user or by Oracle itself—is stored in datafiles. In fact, even the configuration of the database itself and the datafiles, redo log files, tables, indexes, stored procedures, and other objects that exist in the database are stored in a datafile.



A large portion of this book deals with the storage of objects in an Oracle database and how to do this efficiently. Chapter 8 outlines the basics of storage in Oracle8i, while chapters 9 to 12 deal with how object storage is configured within datafiles.

Datafiles in Oracle8i have certain characteristics, such as:

- ♦ A datafile can only be associated with one database. It is not possible to create a datafile that will be part of two databases at the same time. You can, when using Oracle Parallel Server, have two instances access the same databases and datafile.
- ♦ A datafile belongs to a logical storage element called a tablespace. A single datafile can only be associated with one tablespace and will only store data that is configured to reside on that tablespace.
- ♦ Datafiles can be configured to have a fixed size, or they can have an attribute set to allow them to grow, should no free space within the datafile be found. If you configure a datafile to autogrow, you can also configure a maximum size for the datafile to grow to, or set no limit (no recommended).

- ♦ Datafiles are organized internally into database blocks of the same size as the value of the `DB_BLOCK_SIZE` parameter. Each unit of storage inside the datafile is of `DB_BLOCK_SIZE`.
- ♦ Datafiles can be read by any server process in order to place blocks of data in the database buffer cache. Datafiles are normally written to only by the `DBW0` process to minimize the possibility of corruption.

Redo log files

Redo log files contain a chronological record of all changes that have been made to the database. They are written to by the `LGWR` process and operate in a circular fashion, which means that when one redo log file fills up, it is closed and another redo log file is opened for writes. When the second one fills up it is closed and the first, or another redo log file, is opened for writes, and so on.

Each Oracle database must have at least two redo log file groups with one redo log file per group, or the database will cease to allow changes to the data.



The proper configuration of redo log files in an Oracle database and their operation is discussed in more detail in Chapter 7.

Control files

When an Oracle instance is started, one of the first files opened and read is the control file. The control file contains information about what files makeup the database and when the last bit of information was written to them. If the information in the control file and one of the datafiles or redo log files does not match, the instance cannot be opened because the database is considered suspect. This means that some sort of recovery may be required and you will need to deal with it.

The location and number of control files is set by the `INIT.ORA` parameter `CONTROL_FILES`. A database must have at least one control file, although two or more are recommended. If a control file cannot be opened for any reason, the instance cannot start and the database will not be accessible.



Control files and their configuration are covered in more detail in Chapter 6.

Other key Oracle files

While datafiles, redo log files, and control files make up the Oracle database, other files are also required to make the instance work properly, to determine who is allowed to start and stop the instance, and to ensure good recoverability. The files available for this purpose, as shown in Figure 1-3, are the parameter (or `INIT.ORA`) file, the password file, and the archived redo log files.

Parameter files

The Oracle parameter file, also known as the INIT.ORA file, contains the parameters and values that are used to start and configure an Oracle instance. The parameter file may also contain settings for parameters that determine how Oracle behaves in processing a query. An example of the latter is the OPTIMIZER_MODE parameter, which can determine whether Oracle should use statistics in calculating the execution plan for a specific query.

The parameter file can be stored in any location on the hard drive of the computer where the instance will be started and can have any name. The default location for the parameter file is operating system dependent, although the name always defaults to INITSID.ORA, where *SID* represents the name of the instance it will be used for.



For information on how to configure a parameter file and the various parameters that are required to be contained therein, please refer to Chapter 3.

Password file

In order for a user to be able to start and stop the instance, special privileges called SYSDBA or SYSOPER are required. The password file, which is created by using the Oracle Database Configuration Assistant or the ORAPWD utility, lists the users that have been granted one or both of the privileges mentioned. When a user issues the STARTUP or SHUTDOWN command, the password file is checked to ensure that the user has the appropriate privileges.



For more information on how to configure the password file refer to Chapter 2. Chapter 18 provides an explanation of the exact actions permitted when a user is granted SYSDBA or SYSOPER and how to grant these special privileges to users.

In order to add users to the password file, the INIT.ORA parameter REMOTE_LOGIN_PASSWORD_FILE must be set to EXCLUSIVE. The default for this parameter is SHARED and does not allow you to add users to the file.

Archived redo log files

Archived redo log files are copies of full redo log files that have been closed by LGWR and are created by the ARC0 process when automatic archiving is turned on and the database is in ARCHIVELOG mode.

An Oracle database by default runs in NOARCHIVELOG mode, which means that as redo log files become full, they are not copied or backed up in any way. If the redo log files need to be written to again because other log files have also become full and a log switch has taken place, the information in the previously full log is overwritten with no record of the changes within it ever being recorded. From a recovery standpoint, running a database in NOARCHIVELOG mode essentially means that if anything goes wrong, you must restore your last full database backup and re-enter any data that change since then.

When the database is in ARCHIVELOG mode, Oracle does not allow a redo log file to be overwritten until it is archived, that is, copied to a different location, by the ARC0 process. Until the redo log file is archived it cannot be written to. If the ARC0 process is not started indicating that automatic archiving has not been configured by using the LOG_ARCHIVE_START, and LOG_ARCHIVE_DEST_1 Oracle initialization parameters, users will not be able to make changes to the database until archiving takes place.



Although backup and recovery, and archiving are not covered on the “Oracle8i: Architecture and Administration” exam, some additional information on this topic can be found in Chapter 7. The *Oracle8i Backup and Recovery Guide* provides more detailed information on this topic.

Having an archived copy of the redo log file enables you to perform data recovery up to the point of failure, when combined with the current redo log file. ARCHIVELOG mode also allows you to recover the database to a specific point in time or a particular transaction, which provides flexibility from disastrous user error, such as someone issuing the command DELETE FROM CUSTOMERS and then committing the transaction. An archived redo log file is simply an image copy of a redo log file that has been completely filled by LGWR and a log switch to another log file group has taken place. However, having a copy of that redo log file ensures that your critical data can be recovered with more flexibility.

Processing SQL Statements

Objective

List the structures involved in connecting a user to an Oracle instance

List the stages in processing queries, DML statements, COMMITs

Knowing what all the pieces that make up the database and instance are still may not provide a clear picture of how Oracle works. Just like knowing how the jet engine, wings, fuselage, and other parts of an airplane make it run won't help you learn to fly it, knowing what the Oracle architectural components are does not tell you how it works. Someone needs to teach you how to fly an airplane, or understand how Oracle stores and retrieves data.

In understanding how Oracle processes SQL statements and which elements are involved in the process, you first need to understand that the processing of SQL statements originates outside of Oracle but ultimately uses the instance and database to return the results to the user.

Connecting to an instance

In order to submit SQL statements for processing to Oracle, you must first establish a connection to the instance. The process of doing so, as shown in Figure 1-4, may include a number of components, such as the user process, Net8, the physical network, the server computer, server process, the instance and, finally, the database.

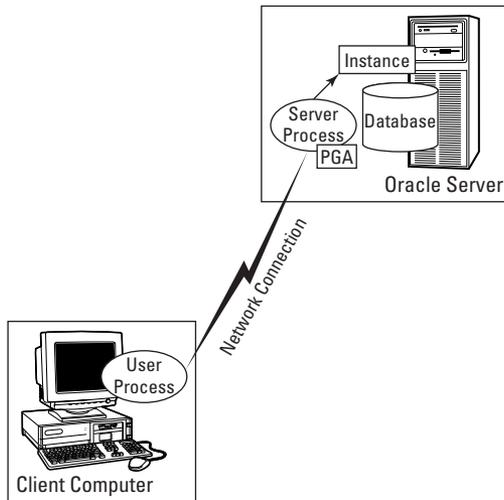


Figure 1-4: In processing SQL statements, a number of components are used when a client sends a request for data to the server.



**In the
Real World**

While a network connection between the client (user process) and the server is not required, this is the more typical configuration. It is also possible to have both the user process and server process on the same computer, as would be the case if you were running SQL*Plus on the same machine as the database. With the exception of the network connection, all other elements of the way requests are sent to the server and the results returned to the user process remain the same.

The process of connecting to an instance is initiated when a user starts an application on the client computer that makes use of data in an Oracle database. By specifying a connect string (username, password, and the instance to connect to), the user instructs the client application to start the process of establishing a connection to the Oracle instance (not the database).

After the user has provided a username and password, as well as the name of the instance to connect to, the Net8 client component on the client computer attempts to resolve the name of the instance in any of a number of ways that have been configured for the client. One of these methods is to use a file on the local client computer called TNSNAMES.ORA to lookup the instance name and determine which machine the instance resides on and the network protocol that needs to be used to communicate with it. Another method is to contact an Oracle Names server to determine the same information. Whichever way is configured for Net8 to use, the process of determining the location of the instance will be transparent to the user process, unless an error is encountered.



The configuration of the Net8 client and server components is beyond the scope of this book and the “Oracle8i: Architecture and Administration” exam. If you want to find out how Net8 works and the various components involved in configuring network support for Oracle, consult the *Net8 Administrator’s Guide*, a part of the Oracle documentation set.

Once Net8 has determined on which computer the instance to be connected to resides, it sends a connection request with the username and password along the network using the protocol configured to communicate to that computer. On the server computer, the *listener* process of Net8 receives the request and launches a dedicated server process on behalf of the user (or connect the client to a dispatcher, in an MTS connection request) and verify the username and password. If the username and password are not correct, the user is returned an error and needs to try again; if correct, the user is connected to the instance. At this point, the communication between the user process and the server process is direct and the *listener* is no longer involved—until another connection attempt is made.

In processing of client requests, the server process receives the request from the user process and, in the processing of the SQL statement, makes use of an area of memory allocated explicitly to it—the Process Global Area (or Program Global Area) or PGA for short. The PGA is private to each server process launched on the server and is used to allocate memory used to perform sorts, store session information (such as the username and associated privileges for the users), keep track of the state of various cursors used by the session, and stack space for keeping track of the values of variables and the execution of PL/SQL code for the session. In a multi-threaded server (MTS) environment, some of this information is kept in the large pool because server processes are shared in an MTS configuration, but a typical dedicated server environment allocates a PGA when a server process starts and de-allocates it when the process terminates.

Statement and transaction processing

The user and server processes, the PGA, and the SGA, along with the other processes that make up the Oracle instance, all work together when a SQL statement is sent to the Oracle server to query or update data. When a user issues any SELECT, INSERT, UPDATE, or DELETE statement, Oracle must go through several steps to process these queries. Consider the processing of the following statement:

```
UPDATE Courses
SET RetailPrice = 1900
WHERE CourseID = 101;
```

When this statement is executed, Oracle goes through the following steps. Oracle first parses the statement to make sure that it is syntactically correct. The *parse* phase is typically done once for any SQL statement and will not need to be performed

again if the same statement is executed by any user, even the same one that sent it across in the first phase. Oracle always attempts to minimize the amount of parsing that needs to be performed because it is quite CPU-intensive, and having to parse many statements increases the amount of work that needs to be performed by the server process.

During this *parse* phase, Oracle (that is, the server process) first determines whether the statement is syntactically correct. If not, an error is returned to the user and no further work is performed; if so, Oracle next determines whether the objects that are referenced (in this case, the Courses table) are available for the user and whether the user has permission to access the object and perform the required task (that is, the UPDATE). It does this by locating information about the object in the data dictionary cache or, if this information is not in cache, by reading the information from the datafiles where the data dictionary resides and placing it in the cache. By placing information about the object in the cache, it ensures that future requests for the object are performed quicker, in case other users are also referencing the Courses table. If the user does not have permissions or the object does not exist, an error is returned to the user.

When the object is located and the user has permissions, the next element of the parse phase is to apply parse locks on the objects being referenced by the statement (the Courses table) to ensure that no one makes a structural change to the object while it is being used, or drops the object. The server process next checks whether the statement has been previously executed by anyone by calculating a unique hash value for the statement and checking the shared pool to see if the shared SQL areas contain the hash value calculated. If so, then Oracle does not need to build the execution plan (the series of tasks to be performed to satisfy the query). It can simply keep the execution plan that was previously created and use it in the next phase of processing. If it cannot find the execution plan, indicating this is the first time the statement is being run, or the statement is no longer in the shared pool and has been aged out, Oracle then builds the execution plan and places it in the shared SQL area in the shared pool. Oracle then proceeds to the execute phase of processing.

During the *execute* phase, Oracle runs the execution plan in the shared pool and performs whatever tasks are contained therein. This includes locating the relevant blocks of data in the Database Buffer Cache, or, if they are not in the cache, the *server* process reads the datafiles where the data resides and loads the data blocks into the Database Buffer Cache within the SGA.

The server process then places a lock on the data being modified (in this case, the row containing course 101). This lock prevents other users from updating the row at the same time you are updating it. Oracle then updates a rollback segment block and a data segment block in the database buffer cache, and records these changes in the redo log buffer. It places the data in the row prior to the update in the rollback block and the new value in the data block.

The Rollback Segment is used for two purposes:

- ♦ Read consistency: Until the change is committed, any user who executes a query for the retail price of course 101 sees the price prior to the upgrade. The new value is not visible until the update is committed.
- ♦ If the system crashes before the transaction is committed, or if the user issues an explicit ROLLBACK command, the data in the rollback segment can be used to return the row to its initial state.

When the modifications to the data are first initiated, the transaction is assigned a unique value called a System Change Number (SCN), indicating that the change is to be synchronized with the datafile, as well as a timestamp, indicating when the change took place. The changes are processed and placed in the redo log buffer to be written to the redo log files by the LGWR process. The process continues until all the data has been updated and the entire query has been satisfied.

The final phase of processing is the *fetch* phase. For a SELECT statement, the *fetch* phase of processing returns the actual data to the user, and it is displayed in SQL*Plus, or the user process that made the request. For an UPDATE operation, or any data manipulation language (DML) statement, the *fetch* phase simply notifies the user that the requisite number of rows has been updated.

When other statements are part of the same transaction, the same series of steps (that is, parse, execute, and fetch) take place for each statement until the user issues a COMMIT or ROLLBACK statement. When the transaction is committed or rolled back, Oracle ensures that all information in the redo log buffer pertaining to the transaction is written to the redo log files, in the case of a COMMIT, or the data blocks are restored to their previous state, in the case of a ROLLBACK, and removes all locks. Oracle also erases the values held in the rollback segment. This means that once a transaction is committed, it is no longer possible to roll it back, except by performing a database restore and recovery.

Key Point Summary

In preparing for the “Oracle&#i DBA: Architecture and Administration” exam, you should remember these important points regarding the Oracle instance, database, and processing of SQL statements:

- ♦ An Oracle instance is made up of a shared memory structure called the System Global Area, or SGA, and background processes that perform work for all users of the database.
- ♦ An Oracle database is composed of datafiles, used to store data, redo log files, which store a chronological record of changes to the data, and control files that record information about the state of the datafiles and redo log files.

- ♦ Other key Oracle files include the parameter or INIT.ORA file, which contains parameters used to configure the instance and tweak the behavior of Oracle, the password file, which stores the names of users granted SYSDBA and SYSOPER privileges, and the archived redo log files, used to store image copies of redo log files that have been filled.
- ♦ The SGA is composed of the shared pool, the database buffer cache, and the redo log buffer (as well as other optional components such as the large pool and Java pool).
- ♦ The shared pool contains the library cache where frequently executed statements and their execution plans are stored, and the data dictionary, or row, cache used to cache privileges and other information about frequently accessed database objects. The shared pool is sized by the SHARED_POOL_SIZE Oracle initialization parameter.
- ♦ The database buffer cache is used to store blocks from the datafiles in memory to allow for faster data retrieval and processing. All I/O requests from clients need to read and write data from/to the database buffer cache. Each buffer in the database buffer cache is the size specified by the DB_BLOCK_SIZE parameter, which is the same size as the blocks in the datafiles.
- ♦ The redo log buffer is used to store changes to data in the database before they are written to the redo log files. It, as well as the redo log files, stores changes to both the data blocks as well as the rollback segment blocks so that both a before and after image are recorded in the redo log files.
- ♦ Oracle8i has five required background processes that are started when the instance starts, and must be running in order for the instance to remain up. These include DBW0, used to write changed or dirty blocks from the database buffer cache to the datafiles; LGWR, which writes information from the redo log buffer to the redo log files; SMON, which performs database recovery and coalesces free space in the datafiles; PMON, which monitors processes and rolls back transactions and de-allocates any resources if a server process fails; and CKPT, which is used to update datafile headers and control files when a checkpoint takes place.
- ♦ A user process is any client application that can connect to the instance and issue requests for data. Examples of user processes include SQL*Plus, Server Manager line mode, Oracle Enterprise Manager, a Web server, and many others.
- ♦ A server process is a process launched on the same computer as the Oracle instance that performs all of the work requested by the user process. The server process also allocates an area of memory called the Process (Program) Global Area, or PGA, to store session state information, cursor state data, a stack for keeping track of variables and PL/SQL program execution, as well as an allocation of memory for sorts required by statement processing.
- ♦ The user process, server process, instance, background processes, and the database work together to perform the necessary operations to allow users to manipulate the data in the database while also ensuring its integrity.



STUDY GUIDE

This chapter introduced a lot of new concepts regarding the way Oracle works and the architecture of the product. A complete understanding is required before you write the “Oracle8i: Architecture and Administration” exam. Use this section of the chapter to test your knowledge of the material presented.

Assessment Questions

1. You created a database with a block size of 8KB. You want to allocate a database buffer cache of 32MB when the instance starts. Which two parameters must you specify in the INIT.ORA file for your instance? (Choose two correct answers.)
 - A. DB_BLOCK_SIZE=8192
 - B. DB_BLOCK_SIZE=8000
 - C. DB_BUFFER_POOL=32M
 - D. DB_BLOCK_BUFFERS=4000
 - E. DB_BLOCK_BUFFERS=32M
2. Which of the following background processes would not cause an instance crash if it failed? (Choose the best answer.)
 - A. DBW0
 - B. ARC0
 - C. LGWR
 - D. SMON
 - E. PMON
3. Which process will send a syntax error message that will eventually be read by the user? (Choose the best answer.)
 - A. User Process
 - B. Background Process
 - C. Server Process
 - D. Oracle Process
 - E. Dispatcher Process

4. Which of the following files must be available in order for the instance to start and the database to be opened for user access? (Choose all correct answers.)
 - A. Control File
 - B. Archived Redo Log File
 - C. Datafile
 - D. Redo Log File
 - E. Parameter File

5. Which process is responsible for reading database blocks from the datafile and placing them in the database buffer cache? (Choose the best answer.)
 - A. DBW0
 - B. LGWR
 - C. User Process
 - D. Server Process
 - E. SMON

6. In order for archiving of redo log files to take place, what mode must the database be in? (Choose the best answer.)
 - A. ARCHIVELOG
 - B. NOARCHIVELOG
 - C. RECOVERABLE
 - D. OPEN
 - E. ARC0

7. In order to set the size of the data dictionary cache in the SGA, which INIT.ORA parameter must you configure? (Choose the best answer.)
 - A. ROW_CACHE_SIZE
 - B. DD_CACHE_SIZE
 - C. DB_BLOCK_BUFFERS
 - D. SHARED_DD_CACHE_SIZE
 - E. SHARED_POOL_SIZE

8. Which phase of statement processing will read the data requested by a SELECT statement and return the rows to the user process? (Choose the best answer.)
- A. Parse
 - B. Prepare
 - C. Execute
 - D. Fetch
 - E. Commit
9. Which two additional background processes need to be configured to support a multi-threaded server (MTS) configuration? (Choose two correct answers.)
- A. RECO
 - B. Dnnn
 - C. LCKn
 - D. Snnn
 - E. ARCO
10. Which of the following will cause DBW0 to write dirty buffers to the datafiles? (Choose all correct answers.)
- A. Three seconds expire since the last write.
 - B. A checkpoint occurs.
 - C. A server process finds a free buffer.
 - D. A table scan takes place.
 - E. A transaction commits.

Answers to Chapter Questions

Chapter Pre-Test

1. An Oracle database is composed of datafiles used to store data, redo log files, which store a chronological record of changes to the data, and control files that record information about the state of the datafiles and redo log files. Only these three sets of files are the database itself, while other files may exist to support the database or instance.
2. The term “Oracle instance” refers to a shared memory structure, the System Global Area, or SGA (composed of the shared pool, database buffer cache, and redo log buffer) and a set of background processes, such as DBW0, LGWR, CKPT, SMON, PMON, and others. These work together to enable multiple users to access data in the database.

3. Oracle8i has five required background processes. These are DBW0, which writes dirty buffers from the database buffer cache to the datafiles; LGWR, which writes information from the redo log buffer to the redo log files; CKPT, which updates datafile headers and control files when a checkpoint takes place; SMON, which performs recovery and coalesces free space; and PMON, which monitors the status of other processes and performs rollback and releases resources in case they fail.
4. The archiver process (ARC0), in conjunction with setting the database in ARCHIVELOG mode and enabling automatic archiving, can help in ensuring good recoverability of your data. The ARC0 process copies a redo log file to an archive log destination on disk or tape thereby helping to create a full chronological record of changes to the database which can aid in data recovery.
5. If a user issues a COMMIT command to end a transaction, a write to the redo log files from the redo log buffer must be successfully performed by LGWR before the user is notified that the transaction has been committed. If LGWR cannot write to the redo log files for any reason, the transaction cannot commit and is rolled back.
6. The Oracle parameter file, also referred to as the INIT.ORA file, provides information on the configuration of an Oracle instance. It is commonly named using the convention INITsid.ORA, where *sid* represents the name of the instance that the file is associated with.
7. The user process typically resides on the client computer — that is, the one that the user is working with, although it can be any computer where the client application to be used to connect to the instance is located. For example, if you are accessing data in an Oracle database through a Web page, the Web server is the client computer and the Web application is the user process and not the browser on the end user's computer.

The server process must reside on the same computer where the instance to be connected to is running.
8. In Oracle Parallel Server environments it is possible to have more than one instance access the same Oracle database. The LCKn process ensures database integrity across instances by coordinating locking requests.
9. The shared pool consists of the library cache, which is used to store the text and execution plan of SQL statements and PL/SQL blocks requested in shared SQL areas. The data dictionary, or row, cache is also part of the shared pool and is used to cache information, such as privileges, of frequently accessed objects.
10. The redo log files are written to most often. This is because LGWR writes information to the redo log files whenever a commit occurs, or the redo log buffer is one-third full with data from a single transaction, or three seconds have elapsed. In an environment where many changes are made to the database by many users, the redo log files cause a lot of write-oriented I/O to take place.

11. The three phases of processing SQL statements are parse, execute, and fetch. The parse phase is where the SQL statement is checked for syntactical accuracy, privileges for the user are verified, and the execution plan is built. The execute phase runs the execution plan created in the parse phase, and the fetch phase returns the rows requested to the user.

Assessment Questions

1. **A, D.** The database buffer cache is configured by setting the values of `DB_BLOCK_SIZE` (in bytes) and `DB_BLOCK_BUFFERS`, to specify the number of blocks of `DB_BLOCK_SIZE` to allocate in the database buffer cache. Because you had a database block size of 8KB, or 8,192 bytes, and you wanted 32MB for the database buffer cache, you need to allocate 32,000KB/8KB or 4,000 buffers.
2. **B.** If the `ARC0` process crashed, the instance would not crash because all of the required background processes (`DBW0`, `LGWR`, `SMON`, `PMON`, `CKPT`) are still running. However, if `ARC0` failed that would cause archiving to fail and may disallow any changes to data in the database until the problem is corrected.
3. **C.** The server process performs the parsing, execution, and fetching of data requested by the user process. If a syntactical error is found during the parse phase, the server process returns the error to the user process, which can notify the user.
4. **A, C, D, E.** In order for an instance to start, the parameter file containing instance configuration information must be available and the control file must be read to determine the names and locations of the datafiles for the database. In order for the database to be opened and users to be able to access the database, all datafiles and redo log files must also be available. Archived redo log files are not needed.
5. **D.** The server process reads blocks from the datafiles and places them in the database buffer cache if they are not already there. The `DBW0` process writes dirty blocks from the database buffer cache to the datafiles, but does not read them into the cache.
6. **A.** In order for archiving to take place, the database must be configured and running in `ARCHIVELOG` mode. If this is not true, Oracle does not allow any archiving of redo log files.
7. **E.** The data dictionary cache is part of the shared pool and is sized by the `INIT.ORA` parameter `SHARED_POOL_SIZE`. There is no way to size the data dictionary cache explicitly even though the shared pool also contains the library cache. Oracle dynamically adjusts the memory split between these two components as required.

8. **D.** The fetch phase of statement processing reads the data and returns the required rows to the user. There is no prepare or commit phase for processing SQL statements.
9. **B, D.** In order to have a working MTS configuration, the shared server processes (*Snnn*) and dispatcher processes (*Dnnn*) must be started and running. The initial number of shared server and dispatcher processes to be started is configured using the `MTS_SERVERS` and `MTS_DISPATCHERS` parameters.
10. **A, B.** Of the situations presented only the three-second timeout and checkpoint cause `DBW0` to write dirty buffers to disk. `DBW0` also writes dirty buffers to disk when a server process cannot find a free buffer, or `LGWR` signals it to do so. Transaction commits, though causing `LGWR` to write to the redo log file, do not cause `DBW0` to write to datafiles. A table scan has no effect on `DBW0` operations.

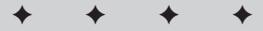
Getting Started with Oracle8i Server

EXAM OBJECTIVES

- ◆ Getting Started with the Oracle Server
 - Identify the features of the Universal Installer
 - Set up operating system and password file authentication
 - List the main components of Oracle Enterprise Manager and their uses

2

CHAPTER



CHAPTER PRE-TEST

1. What methods are supported in Oracle to authenticate privileged users?
2. What can a user who has been granted the SYSDBA role do that someone granted the SYSOPER role cannot?
3. What is Optimal Flexible Architecture?
4. Which command-line tools can you use to start and stop instances? What is the preferred tool, according to Oracle?
5. What is the architecture of Oracle Enterprise Manager and what are its advantages?
6. Is it possible to administer Oracle databases using graphical tools without connecting to an Oracle Management Server, and which tool provides the most similarity to the OEM console?
7. How can you change the password for the user INTERNAL on a UNIX system?
8. If you wanted to configure operating system authentication, what Oracle initialization parameter must be configured and what value must be specified?
9. Is it possible to perform a fully scripted installation of Oracle? If so, how would you do this; if not, why not?
10. What two users exist when a database is first created and which is configured as a privileged user for password file authentication?

The first step in being able to configure an instance and manipulate data in a database is to install the Oracle software and become familiar with the various administration tools available. The Oracle Universal Installer, a Java-based application common to all major Oracle8i platforms, performs the installation of Oracle database software and additional products. Once the software is installed, you can use command-line programs such as SQL*Plus, Server Manager line mode, or SQL*Worksheet to configure and administer the instance and database. If you want to perform enterprise-wide administration of databases across the entire company, you can also make use of Oracle Enterprise Manager.

Oracle Universal Installer

Objective

Identify the features of the Universal Installer

Oracle has a slightly different installation process for each platform on which it is available. It is always a good idea to read the platform-specific documentation that is provided in print form, or on the CD-ROM or other media, prior to starting the installation process. In making the job of installing Oracle appear similar on each platform where it is available, the actual installation of Oracle is performed using the Oracle Universal Installer, which is common to all major Oracle platforms including WindowsNT/2000, Sun Solaris, HP-UX, AIX, Linux, and others.

The Oracle Universal Installer has a number of characteristics and features that help the DBA to perform similar installations on different platforms:

- ♦ **Java-based**— The Oracle Universal Installer is written in Java, and looks and feels the same on any platform that a Java engine runs. It also makes Oracle's job easier because Java engines are available for almost any platform in use today. Therefore, modifying and enhancing the Oracle Universal Installer is easier for Oracle as the code only needs to be written once for it to run on all supported platforms.
- ♦ **Dependency checking**— When you use the Oracle Universal Installer to install products on your computer, it automatically checks to see which other products might also need to be installed in order for your choice to function properly. The Universal Installer then determines if the required components are already on the computer and, if not, selects them for installation as well.
- ♦ **Multiple Oracle home support**— The Oracle Universal Installer will keep track of all the Oracle home directories that exist on the target computer. Multiple Oracle homes are required if you want to install both the Oracle server and additional development tools, or multiple versions of the Oracle server on the same computer. The Universal Installer ensures that each product that requires a separate Oracle home has it created and keeps track of which products and versions are installed in each Oracle home.

- ♦ **National language support** — When installing Oracle software, the Universal Installer checks to see what the computer's regional/globalization settings are and configures itself to adhere to these settings. It also does the same for the software that is being installed to ensure that the interactive experience that the user is expecting is delivered. There is nothing worse than being a user in France who is being forced to install a product by following English menus — the French don't like that.
- ♦ **Web-based Installation** — When you are prompted by the Oracle Universal Installer for the location of the software that you are installing, you can now specify a physical or network disk location using UNC names of NFS mount-points, or you can specify a URL where the files can be found. This enables you to create a Web page that would be used to invoke the Oracle Universal Installer and then point the user to a server close to them that contains the package files for the application being installed. This makes large-scale deployments easier.
- ♦ **Unattended installation** — The Oracle Universal Installer can be invoked from the command line and passed the name of a response file that has all the parameters required for the installation to proceed. Depending on the operating system that you are using to perform the installation, the syntax for invoking the Oracle Universal Installer from the command line differs slightly. For example, to invoke the Oracle Universal Installer on Windows NT/2000, the syntax is

```
setup -responsefile respfile [-silent] [-nowelcome]
```

The syntax on a Sun Solaris platform is

```
runInstaller -responsefile respfile [-silent] [-nowelcome]
```

The Oracle Universal Installer can be invoked from the CD-ROM or from the operating system on which the software will be installed. On a Windows NT/2000 computer, the Oracle Universal Installer is installed in the Program Files\Oracle\oui\install folder on the same hard disk that the operating system is located (that is, the hard disk pointed to by the %SystemDrive% environment variable). On a Sun Solaris computer, the Oracle Universal Installer is located in the INSTALL/install/solaris directory. On all UNIX-based systems, like Solaris, the name of the directory where the file is located and the name of the executable itself are case sensitive.

The `-nowelcome` command-line option tells the Oracle Universal Installer to not display the Welcome screen when started. The default is to display the Oracle Universal Installer Welcome screen. The `-silent` option tells the Oracle Universal Installer to not tell the user what is happening during the installation, but to simply perform all of the tasks specified in the response file.

- ♦ **Intelligent uninstallation** — Once you install the product using the Universal Installer, it keeps a record of the installation and allows you to uninstall a portion of the product or the product in its entirety. While performing the uninstall, the Universal Installer prompts you if you need to uninstall additional

components—or if the uninstall will cause other products to fail, which of those products must also be removed or the specific portion of the uninstall affecting them cancelled.

- ♦ **Support for user-defined packages**—The Universal Installer enables you to add your own components to the list of packages to be installed when it is invoked. In this way, you can install the Oracle server software and your own software at the same time. Furthermore, if specific utilities need to run during the installation process, the Universal Installer enables you to invoke them automatically from your installation script.

As is fairly evident from the list of features presented, the Oracle Universal Installer is a vast improvement over the various scripts and other methods used by Oracle to install its software in the past. Furthermore, it also provides you with information on what is installed on your computer in the various Oracle homes that you may have configured.

The following step by step guides you through the process of determining what software is already installed on your computer, as well as available for installation, by invoking the Oracle Universal Installer. The step by step assumes that you are running on a Windows NT/2000 platform, so make appropriate changes if you are using Solaris, Linux, or another operating system.

STEP BY STEP: Invoking the Oracle Universal Installer

1. If you have already installed Oracle software on your computer, invoke the Oracle Universal Installer from the menu or command line, as appropriate for your platform. On Windows NT/2000, click the Start Menu ⇨ Programs ⇨ Oracle Installation Products ⇨ Universal Installer.

If you have not installed any Oracle software on your computer, insert the Oracle software CD-ROM in your CD-ROM drive. If you have configured AutoPlay on your CD-ROM drive, the Universal Installer starts automatically; if not, start the installer by locating the SETUP.EXE in the root of the CD-ROM drive and running it.

The Oracle Universal Installer will start by displaying its splash screen and then present the main Welcome screen, as shown in Figure 2-1. If you invoked the Universal Installer with the `-nowelcome` switch, you won't see the Welcome screen, but rather, you'll be presented with a screen similar to Figure 2-2, which shows the location of the source files for installation and where the software will be installed.



Figure 2-1: The Oracle Universal Installer displays the Welcome screen when invoked.

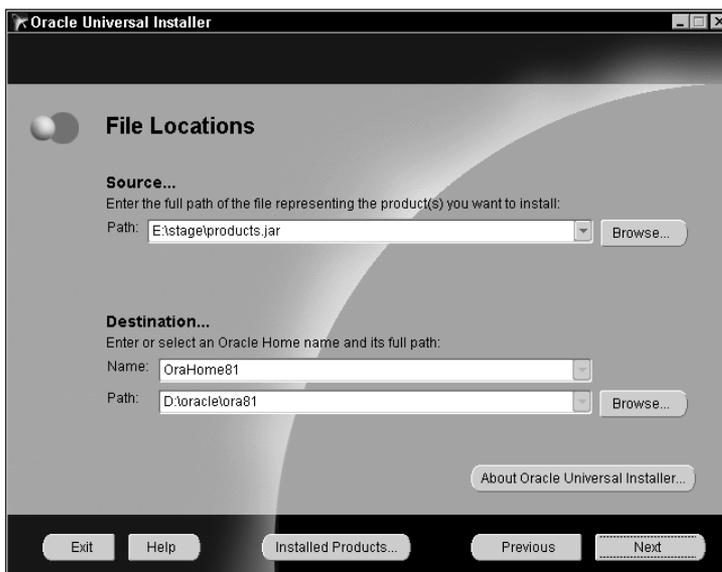


Figure 2-2: Specifying the `-nowelcome` startup switch for the Oracle Universal Installer presents the default source and destination location for the software to be installed.

If you invoked the Universal Installer with the `-nowelcome` command-line switch, exit and invoke it normally.

2. To get a list of installed products, click the Installed Products button and you will be presented with the Inventory dialog box, similar to Figure 2-3, after a short period of time. If you click the plus sign next to an Oracle home, you will be able to see a list of products in that Oracle home.

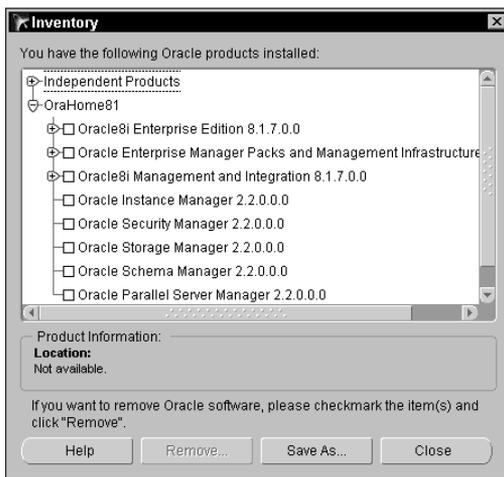


Figure 2-3: Clicking the Installed Products button on the main Welcome screen will present the Inventory dialog box, listing all Oracle homes on the computer.

Clicking a product component will enable the Remove button, allowing you to uninstall it, as shown in Figure 2-4.

3. Click Close on the Inventory dialog box to be returned to the main Welcome screen.
4. If you want to proceed with the installation of the product, click Next in the Welcome screen, at which point you will be presented with the File Locations screen, as shown previously in Figure 2-2.

If you want to remove products, you can also click the Deinstall Products button, which will also present the Inventory dialog box, shown previously in Figure 2-3.

5. Click the Exit button and, when prompted, confirm that you want to exit the Oracle Universal Installer.

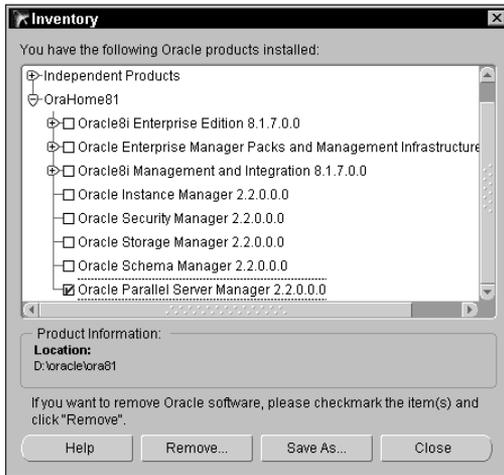


Figure 2-4: Clicking a product will enable the Remove button, allowing you to remove the product and its components from the computer.

Optimal Flexible Architecture (OFA)

With the release of Oracle8, Oracle introduced us to the Optimal Flexible Architecture, or OFA. OFA is a method of organizing datafiles and database components to make it easy for a DBA to locate files and administer the database. The Oracle Universal Installer, when creating a starter database, conforms to OFA rules in the creation of that database. Furthermore, the Universal Installer creates a file and directory structure that makes compliance with OFA easy to adhere to in the creation of additional database using the Database Configuration Assistant.

The Optimal Flexible Architecture was developed by Oracle's consulting services to make the performance and monitoring of Oracle databases easier. OFA specifies that at least three sets of directories should be used to reduce contention and provide good performance. One set of directories is used to store Oracle binary files such as the Oracle executables themselves, as well as associated support files that should normally not be changed. A second set of directories is used to store control files, redo log files, and other administrative files like the INIT.ORA file for each database on the computer. Finally, a third set of directories is used to store all the datafiles. Each set of directories should be on a separate physical hard disk, and further manual optimization may also be required to ensure good performance.

While OFA is not perfect, it does provide the basis for good performance and easier administration, including

- ♦ A structured approach for locating the various files that are required and used by Oracle. This structured approach, when followed, enables any DBA to easily become familiar with any database and server that they are asked to administer.
- ♦ Easier administration of databases while performing such tasks as backing up and restoring databases because of a familiar file and directory structure. If you need to create additional datafiles, you can figure out where to put the file by adhering to the OFA structure.
- ♦ Because the OFA configuration makes use of multiple physical disks on the computer, this makes for improved performance of the databases that use it by reducing disk contention for datafiles, binary files, and redo log files. While simply adhering to OFA principles gives you optimal performance for your databases and server, it provides a starting point for further performance monitoring and tuning.
- ♦ If you have multiple Oracle homes on the same computer, or are running multiple versions of Oracle on the same computer, each version can adhere to OFA principles and thereby make it less likely that files required by one version of Oracle, or one Oracle package, will overwrite those of another version or package. OFA helps to separate potentially conflicting files, thereby making administration easier and contention less likely.

When using the Oracle Universal Installer, and installing and configuring any version of Oracle after 8.0 on your computer, OFA and its principles are adhered to by the Universal Installer and other Oracle utilities.

Privileged Users

Objective

Set up operating system and password file authentication

In order to administer any database, an individual must be assigned appropriate rights and privileges. Each individual that needs to administer the database should have a user account created in the database. This user account needs to be granted the appropriate privileges to create other users, create and manage objects, and generally perform any tasks that are needed. All of the system privileges required to perform full database administration are granted to the DBA role, and then the role is granted to users. A role is simply a named set of privileges that may be granted. When you create a database, two user accounts are automatically created that are assigned the DBA role. They are the user SYS with a password of CHANGE_ON_INSTALL and the user SYSTEM with a password of MANAGER. In Oracle, the username and password are not case sensitive.



For more information on Oracle database security, users, system privileges, and roles, refer to Part V. This chapter won't provide an in-depth discussion on the security elements in Oracle, but rather gives an overview of how privileged users work in Oracle.

When you create a database, the user SYS is created first and then all data dictionary objects are created within the SYS schema. A schema is a collection of all database objects that are owned by a user. This essentially means that the user SYS owns the data dictionary and therefore cannot be removed. The user SYS is also granted all system privileges by being granted the DBA role when the database is created. The user SYSTEM, during the database creation process, is granted the DBA role. This user is also given full privileges on all data dictionary objects and, for the most part, has all the privileges that SYS has.

While Oracle automatically creates these two user accounts when you create a database, you will probably want to do a couple of things to make the administration of your own databases more closely meet your organizational requirements. One of the first things that you should do is change the passwords for the SYS and SYSTEM users — otherwise, anyone with a copy of this book is able to gain full administrative rights on your database. The second thing you should do is create additional user accounts and assign them the privileges required to perform database administration of your database. Each database is its own security domain, so you may have to create the same user account in multiple databases. The user accounts you create in the database, as well as SYS and SYSTEM, are authenticated by Oracle when a connection to the instance is made and the database is open.



For information on how to start and shut down an instance, and the various states (NOMOUNT, MOUNT, and OPEN), see Chapter 3.

Now this is all well and good if the database already exists and you have one or more instances working against the database, but what if you need to create the database in the first place? Are there any restrictions on who can do this? The answer is — yes.

Oracle allows users that have been granted the DBA role in a database to administer it in its entirety, but in order to create databases or to start an instance that users will connect to so that they may gain access to database objects, you need to be connected as a privileged user. A privileged user is any individual with a user account that is allowed to perform privileged operations such as startup and shutdown of the instance, database recovery, and so on. These users have to be authenticated not by looking in the data dictionary for the user account and verifying that the username and password presented are valid — because the data dictionary is not yet available — but by other mechanisms.

Authenticating privileged users

When authenticating privileged users, because Oracle cannot be certain that the database is available or the instance is even started, other methods need to be used. Oracle8i supports authenticating privileged users either through the operating system or by means of a password file, which is created by running the ORAPWD utility program or automatically when using the Database Configuration Assistant. Figure 2-5 provides a flow chart of the process Oracle follows in authenticating privileged users, depending on whether the user is physically logged on to the computer where the database resides or is performing administration from a remote computer.



Tip

In previous versions of Oracle, you could use the `CONNECT INTERNAL` syntax to connect to an Oracle instance as a privileged user. While this syntax is still supported, it is no longer recommended as Oracle9i will discontinue support for the syntax. Because of backward compatibility, and the fact that many DBAs still make use of this syntax, when you create a database using the Database Configuration Assistant, or run the ORAPWD utility to create a password file, you will still be asked to specify a password for the INTERNAL user.

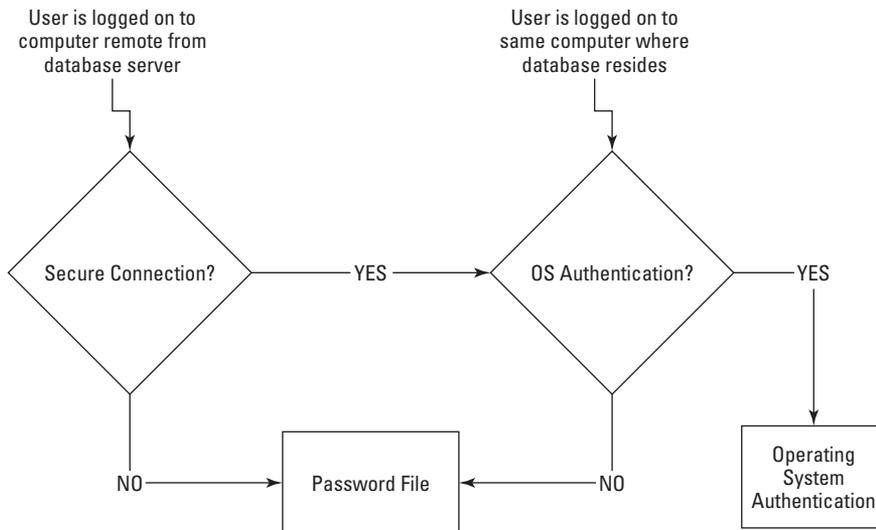


Figure 2-5: Oracle allows the use of either the operating system or a password file when authenticating privileged users.

When authenticating as a privileged user, the first question that needs to be asked is “Are you logged in and sitting at the computer where the database resides, or is the computer from which you wish to perform a privileged operation not the one where the instance resides?” If you are connected remotely, in order to be authenticated using the operating system, you must have a secure connection. A secure connection is typically one that has some sort of encryption and makes use of the Oracle Advanced Security Option. Secure connections include authentication using Kerberos, and encryption using Secure Sockets Layer (SSL) or IPSec.



For more information on Oracle’s Advanced Security Option, refer to the *Oracle Advanced Security Administrator’s Guide* in the Oracle documentation set.



You won’t be tested on how to configure and make use of the Advanced Security Option on the Oracle®i: Architecture and Administration exam. Knowing that a secure connection is required to use operating system authentication from a remote computer should suffice.

If you want to perform remote privileged administration of your database but do not have a secure connection, you need to make use of password file authentication. Password file authentication enables an Oracle user account that is created in the database to be granted special roles called SYSDBA or SYSOPER and have the usernames granted these privileges stored in a file on disk in an encrypted fashion.

The SYSDBA/SYSOPER and OSDBA/OSOPER roles

When a privileged user is authenticated using either password file or operating system authentication, they are granted special privileges to start and stop an instance, as well as perform other tasks, by being assigned special roles. When using password file authentication, the user can be assigned either the SYSOPER or SYSDBA role. When using operating system authentication, the OSOPER and OSDBA roles are granted to the user, depending on the group membership and other configuration options selected for the user account.

The SYSOPER (and OSOPER) role grants the user most privileges to start and stop an instance, but does not enable the user to issue the CREATE DATABASE command to create a database. The SYSDBA (or OSDBA) role adds this privilege to the list provided by SYSOPER. The full list of privileges that are available to users that have been granted these roles are shown in Table 2-1.



For more information on the meaning of some of these privileges and the WITH ADMIN OPTION, refer to Chapter 18.

Table 2-1
Commands Available to SYSOPER/OSOPER and SYSDBA/OSDBA

<i>Privilege</i>	<i>Commands Available (Including All Variations)</i>
SYSOPER OSOPER	STARTUP SHUTDOWN ALTER DATABASE MOUNT ALTER DATABASE OPEN ALTER DATABASE BACKUP CONTROLFILE ALTER TABLESPACE BEGIN BACKUP ALTER TABLESPACE END BACKUP ALTER DATABASE RECOVER DATABASE ALTER DATABASE ENABLE RESTRICTED SESSION ALTER DATABASE DISABLE RESTRICTED SESSION ALTER DATABASE ARCHIVELOG
SYSDBA OSDBA	SYSOPER privileges WITH ADMIN OPTION CREATE DATABASE RECOVER DATABASE UNTIL

Configuring operating system authentication

The way you configure a user account to use operating system authentication and be granted the associated privileges depends on the operating system on which the database resides. For example, setting up operating system authentication on a UNIX-based system, such as Sun Solaris, is different from doing the same on a Windows NT/2000 computer. The one common element for configuring operating system authentication on all platforms is the setting for the Oracle initialization parameter `REMOTE_LOGIN_PASSWORDFILE`. In order to make use of operating system authentication, this parameter needs to be set to `NONE` in the `INIT.ORA` file corresponding to the instance, as follows:

```
REMOTE_LOGIN_PASSWORDFILE=NONE
```

The parameter cannot be changed dynamically, which means that once set you need to shut down and restart the instance for the change to take effect. The default value for this parameter is `NONE`, although if you create a database using the Oracle Database Configuration Assistant, the `INIT.ORA` file will be modified to make the default for the database being created `EXCLUSIVE`. The third option, which should not be used, is `SHARED`.



For information on the `INIT.ORA` file and starting and stopping an instance, as well as how to modify Oracle initialization parameters, see Chapter 3.

Configuring operating system authentication on UNIX systems

To configure operating system authentication on UNIX-style systems, you need to perform the following:

1. Add the operating system user to the group that was created before the Oracle software was installed (usually called *dba*) that owns the Oracle software. In other words, when you install Oracle on a UNIX-based system, you need to create a group that has full permissions on the Oracle software. The Oracle Universal Installer, when installing the Oracle software, grants the group specified during the installation process administrator and operator privileges to all Oracle databases.

In order to use operating system authentication, add the UNIX user account to this group.

2. Ensure that the `REMOTE_LOGIN_PASSWORDFILE` parameter has been set to `NONE` in the `INIT.ORA` file for the database and instance you want the user to be able to administer.
3. Verify that the user is a member of the group by reviewing the `/etc/group` and `/etc/passwd` files, or other appropriate mechanisms for your variant of UNIX.
4. If you have configured everything properly, attempt to connect to the instance as a privileged user from a command-line utility such as SQL*Plus or Server Manager line mode by issuing one of the following commands:

```
CONNECT / AS SYSDBA;  
CONNECT / AS SYSOPER;
```

If the command succeeds and you receive a “Connected” message, you have successfully implemented and tested operating system authentication on your UNIX-based server.

Configuring operating system authentication on Windows NT/2000 systems

To configure operating system authentication on a Windows NT/2000 system, you need to perform the following tasks:

1. On the computer where the database resides, create a new local group called `ORA_DBA` or `ORA_OPER`, to be granted the `OSDBA` or `OSOPER` roles to all instances on that server.

You can also create a local group to be granted `OSDBA` or `OSOPER` privileges to a specific instance by naming the group `ORA_SID_DBA` or `ORA_SID_OPER`, where `SID` represents the name of the instance you want to administer. For example, if you want to create a group to administer the `CERTDB` instance on the Windows NT computer, you can create a group called `ORA_CERTDB_DBA`, the members of which will be granted the `OSDBA` role on the `CERTDB` instance.



Tip

In order for OS authentication to work properly on Windows NT computers, the group must be a local group on the computer and not a global group on the domain. On Windows 2000 computers in Windows 2000 Active Directory domains, a domain local group can be used instead of a local group on the server where the instance will be started, but only if you are using Oracle 8.1.7 or later.

2. Add the Windows NT/2000 user accounts (either domain user accounts or local user accounts) to the group you have created. If you want to follow Microsoft's recommended A-G-L-P strategy for user and group assignment, you can add a global group from the domain to the local group you created. This makes administration of privileged users easier in the long run.
3. In the SQLNET.ORA file on the server, make sure that you have the following line:

```
SQLNET.AUTHENTICATION_SERVICES=(NTS)
```

4. Ensure that the INIT.ORA file for the instance you want to be administered by using operating system authentication includes the following line:

```
REMOTE_LOGIN_PASSWORDFILE=NONE
```

5. If you have configured everything properly, attempt to connect to the instance as a privileged user from a command-line utility such as SQL*Plus or Server Manager line mode by issuing one of the following commands:

```
CONNECT / AS SYSDBA;  
CONNECT / AS SYSOPER;
```

If the command succeeds and you receive a "Connected" message, you have successfully implemented and tested operating system authentication on your Windows NT/2000-based server.

Configuring password file authentication

While using operating system authentication may seem easier and enable you to grant individuals the ability to administer several different databases, it also permits those individuals that can administer operating system users to allow anyone else full administrative control over your databases. For example, if BobW is a domain administrator on your Windows NT domain, he can grant any user in the domain membership in the ORA_DBA local group on an Oracle server computer. If this is what you want, there is no problem; if not, password file authentication enables you, as an Oracle administrator, to determine who is allowed to perform privileged operations on your databases.

In order to configure password file authentication, you need to run a utility called ORAPWD that creates a password file for a database. Each database will have a different password file, which means that, unlike with operating system authentication where a single OS user can be granted administrative rights to several databases, password-authenticated users are users specific to an Oracle database for whom the SYSDBA and SYSOPER privileges will need to be granted explicitly.

Creating a password file on a UNIX-based server

To create a password file on a UNIX-based Oracle database server computer, follow these steps:

1. Log on to the computer where the password file is to be created as a member of the *dba* group that owns the Oracle software, or a system administrator.
2. Run the password file utility by passing the appropriate parameters, as follows:

```
orapwd file=filename password=password entries=number
```

where “filename” refers to the name of the password file, “password” is the password to be assigned to the special user INTERNAL, and “entries” is the maximum number of users to whom SYSDBA and SYSOPER privileges will be granted through this password file. To change any of this information, you need to delete and then re-create the password file.

The location and name of the file is important. Though the password file does not have to be placed here initially, before it can be used the password file must reside in the location pointed to by the \$ORACLE_HOME/dbs directory and *must* have a name of the format *orapwSID* where *SID* is the value of the ORACLE_SID environment variable for the instance and the rest of the filename is lowercase (UNIX is case sensitive).

For example, to create a password file for the CERTDB instance that supports granting privileged administrative access for up to ten users and sets the INTERNAL password to *orapass1*, issue the following command:

```
orapwd file=$ORACLE_HOME/dbs/orapwCERTDB entries=10 password=orapass1
```

In this example, the ORACLE_HOME environment variable has been properly initialized to the location of the Oracle software.

3. Ensure that the Oracle software (that is, *dba* group or the UNIX user that was used to install Oracle in the first place) has write privileges to the password file. You can either use the *chmod* command to modify the privileges on the file, or change the ownership of the file to the same user that owns the Oracle software using the *chown* command, or change the group designation and privileges for the file using the *chgrp* and *chmod* commands. For example, the following command changes the ownership of the password file to be the same as the user used to install Oracle — *oracle8i* — which grants the Oracle software full privileges on the file:

```
chown oracle8i $ORACLE_HOME/dbs/orapwCERTDB
```

If the Oracle software cannot write to the file, you won't be able to add users to the password file. It is important that you verify that the group that owns Oracle, or the user that was used to install the Oracle software, has write privileges to the password file.

4. Set the REMOTE_LOGIN_PASSWORDFILE parameter in the appropriate INIT.ORA file to EXCLUSIVE, as shown here:

```
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
```



Tip

The REMOTE_LOGIN_PASSWORDFILE parameter also accepts a value of SHARED. However, setting the parameter to SHARED does not enable you to grant SYSDBA or SYSOPER privileges to users, but does enable more than one instance to access the database and have individuals authenticated as SYS or INTERNAL. Unless you are using Oracle Parallel Server, do not set the value of this parameter to SHARED. If you are using Oracle Parallel Server, consider using operating system authentication instead.

5. Connect to the instance as the user INTERNAL or SYS and grant others the SYSDBA or SYSOPER privileges, as follows:

```
SQL> CONNECT internal/orapass1
Connected.
SQL> GRANT SYSDBA TO SYSTEM;
```

Grant succeeded.

```
SQL>
```

For each additional user that you want to be able to perform administrative or operator-related functions, you need to grant the SYSDBA or SYSOPER privileges.

Creating a password file on a Windows NT/2000-based server

To create a password file on a Windows NT/2000-based Oracle database server computer, follow these steps:

1. Log on to the computer where the password file is to be created as a member of the local Administrators group.
2. Run the password file utility by passing the appropriate parameters, as follows:

```
orapwd file=filename password=password entries=number
```

where “filename” refers to the name of the password file, “password” is the password to be assigned to the special user INTERNAL, and “entries” is the maximum number of users to whom SYSDBA and SYSOPER privileges will be granted through this password file. To change any of this information, you need to re-create the password file.

The location of the password file is different on a Windows NT/2000 computer than a UNIX system. On Windows-based servers, the password file needs to be located in the directory pointed to by the %ORACLE_HOME%\DATABASE path and *must* have a name of the format pwd.SID.ORA where SID is the value of the ORACLE_SID environment variable for the instance.

If you want to specify a different location for the password file of a database on your Windows NT/2000 computer, you can add a Registry entry called `ORA_SID_PWFIL`, where *SID* is the name of the instance for which the parameter applies. If you want to change the default for all password files, specify a path when configuring the `ORA_PWFIL` Registry entry under the appropriate Oracle home.

For example, to create a password file for the `CERTDB` instance that supports granting privileged administrative access to ten users and sets the `INTERNAL` password to `orapass1`, issue the following command:

```
orapwd file=%ORACLE_HOME%\DATABASE\pwdCERTDB.ORA entries=10
password=orapass1
```

In this example, the `ORACLE_HOME` environment variable has been properly initialized to the location of the Oracle software.

3. Set the `REMOTE_LOGIN_PASSWORDFILE` parameter in the appropriate `INIT.ORA` file to `EXCLUSIVE`, as shown here:

```
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
```

4. Connect to the instance as the user `INTERNAL` or `SYS` and grant others the `SYSDBA` or `SYSOPER` privileges, as follows:

```
SQL> CONNECT internal/orapass1
Connected.
SQL> GRANT SYSDBA TO SYSTEM;

Grant succeeded.

SQL>
```

When you run the Oracle Database Configuration Assistant to create a new database, it automatically creates a password file for you. If you issue the `CREATE DATABASE` command manually, you need to create the password file by running `ORAPWD`. If you want to change the number of entries supported by the password file, you need to delete the old one and create a new one by running the `ORAPWD` utility, and reassign the `SYSDBA` and/or `SYSOPER` privileges to any user that may have had them previously, as this information will be lost when the file is deleted.

The reason you need to configure privileged users, if not already clear, is quite simple—you cannot create a database unless you are a privileged user. If you cannot create additional databases with Oracle, then what's the point?

Oracle Tools for Administration

Once you have configured either operating system or password file authentication to perform privileged operations, you need to choose a tool with which to administer your database and instance. Because no one tool serves all purposes, and because there are both pros and cons to command-line (that is, interactive) and graphical tools, Oracle provides a variety to choose from. Of the interactive/command-line variety, we have SQL*Plus (both a Windows and non-Windows version), Server Manager line mode, and SQL*Plus Worksheet, a Windows-based tool with a slightly different interface from SQL*Plus. Graphical tools for administering Oracle are based on Oracle Enterprise Manager, and, unlike command-line tools, enable you to administer several databases and servers at the same time using a single interface. There are also a series of assistants (sometimes referred to as *wizards*), which enable you to perform repetitive tasks (such as creating a database) easier and prompt for the proper answers and parameters. These, along with tools for managing network elements of Oracle, as well as migration and other facets can also be found, if selected during the installation of the Oracle database or other components.



Tip

Oracle 8.1.7 on Windows NT/2000 computers also includes a Microsoft Management console snap-in called the Oracle Administration Assistant for Windows NT. You can use the Oracle Administration Assistant for Windows NT to configure and manage operating system authentication (that is, add users and delete users who will be granted the OSDBA or OSOPER roles), modify Registry entries, as well as manage roles and those users who will be able to administer each database. The Oracle Administration Assistant for Windows NT MMC snap-in is designed to enable those administrators familiar with performing Windows NT/2000 administration to have a familiar interface to also administer security elements of Oracle servers.

Interactive/command-line administration tools

Oracle8i provides two main command-line or interactive utilities to query and administer a database. These are SQL*Plus and Server Manager line mode. Prior to Oracle8, the functionality provided by Server Manager line mode was not available in SQL*Plus. However, as Oracle released 8.1.5 and later versions of the software, the capabilities of both of these programs became essentially identical. Today, commands that only worked in Server Manager line mode also work in SQL*Plus, and vice versa. The main reason for this change is that having two command-line utilities for basically the same task is inefficient where one will do. For this reason, you should use SQL*Plus instead of Server Manager line mode.

Server Manager line mode

Server Manager line mode is invoked by typing **svrmgrl** at the operating system command prompt, or shell, to bring up the Server Manager prompt, as follows:

```
C:\>set ORACLE_SID=ORCL

C:\>CD ORCL

C:\ORCL>svrmgrl

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SVRMGR> connect internal;
Connected.
SVRMGR> select status from v$instance;
select status from v$instance
*
ORA-01034: ORACLE not available
SVRMGR>
```

Server Manager line mode is typically used on the same computer where the database resides. For this reason, you should set up environment variables to make sure that you are going to connect to the proper instance and administer the one you want. The “set ORACLE_SID=ORCL” command on a Windows NT/2000 computer tells it to connect to the ORCL instance on the local machine when invoking Server Manager line mode. You can also connect to a remote database using the appropriate “username/password@instance” syntax, as shown here:

```
C:\CERTDB>svrmgrl

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SVRMGR> connect system/manager@certdb.delphi.bradsys.com
Connected.
SVRMGR>
```

Once you have connected to the instance using Server Manager line mode, you can execute any SQL statement that you have privileges for, including the ability to start up and shut down an instance, or create a database.

Within Server Manager line mode there are a number of shortcuts that you can execute to provide information about the database, such as the command ARCHIVE LOG LIST to determine archiving information about the database, as well as commands that can set the display of output when running Server Manager line mode.

The “show all” command inside the utility will provide information on the current values for set commands, such as “charwidth” or “numwidth,” as well as other information, as is visible from the following output:

```
SVRMGR> show all
Instance                local
Spool                   OFF
Timing                 OFF
Termout                ON
Echo                   OFF
Stoponerror            OFF
Autorecovery           OFF
Logsource               <default>
Maxdata                20480
Numwidth               10
Charwidth              80
Longwidth              80
Datewidth              9
Labwidth               32
Compatibility          NATIVE
Retries                infinite
Server Output          OFF
Autoprint              OFF
Fetchrows              infinite
Appinfo                OFF (USERTEXT : Oracle Server Manager)
SVRMGR>
```

The meaning of these settings should be familiar to you at this stage of your Oracle experience because it parallels SQL*Plus settings that you should already be familiar with.



It is very unlikely that you will be tested on the settings available in Server Manager line mode when taking the Oracle8i: Architecture and Administration exam. This is partly because Oracle is focusing on SQL*Plus for database administration and partly because the level of detail is a bit esoteric.

Server Manager line mode provides a simple command help facility that displays a list of commands when you type **help** on the command line, as shown here:

```
SVRMGR> help
```

The following are SIMPLIFIED syntax descriptions. For complete syntax descriptions, please refer to the Oracle Server Manager User's Guide.

```
STARTUP      [DBA] [FORCE] [PFILE=filespec] [EXCLUSIVE | SHARED]
             [MOUNT dbname | OPEN dbname] [NOMOUNT]
```

```
SHUTDOWN    [NORMAL | IMMEDIATE | ABORT]
```

```
MONITOR     For graphical modes only, bring up a monitor
```

```
ARCHIVE LOG [START] [STOP] [LIST] [NEXT] [<n>] [ALL] ['destination']
```

```

RECOVER      { [DATABASE [MANUAL] ] | [TABLESPACE ts-name [,tsname]] }

CONNECT      [username [/password] ] [INTERNAL] ['@'instance-spec]
DISCONNECT

SET          options: INSTANCE, ECHO, TERMOUT, TIMING, NUMWIDTH, CHARWIDTH
SHOW        LONGWIDTH, DATEWIDTH, AUTOPRINT and for SHOW: ALL, SPOOL

EXIT
REM
            SQL statements can also be executed.

SVRMGR>

```

Server Manager line mode can also have the name of a SQL script to run passed to it when it is invoked by using the following syntax:

```
svrmgr1 command=["SQL string" | @<scriptname>]
```

Whatever follows the “command=” portion of the command line can be a SQL command, enclosed in double quotes, or the name of an ASCII file containing SQL and PL/SQL code preceded by the @ sign.

SQL*Plus

Server Manager line mode won't be supported in future versions of Oracle. The reason it existed as a separate utility from SQL*Plus for such a long time is that SQL*Plus would not allow you to connect to an instance that was not yet started, nor would the CREATE DATABASE command be allowed in SQL*Plus. For privileged operations, you had to use Server Manager line mode. These restrictions are now removed and any valid command in Server Manager line mode is also valid in SQL*Plus.

SQL*Plus, like Server Manager line mode, has a command-line interface invoked by typing **sqlplus** on the command line, as well as a Windows-based graphical version whose command file is “sqlplusw.exe”. The commands supported by both versions are the same, and the way that they work is almost identical, except that one presents a graphical window, which can be scrolled up and down to review past work, whereas the other has an interface similar to Server Manager line mode.

When invoking SQL*Plus (or SQL*PlusW) from the command line, you have the option to pass it parameters. The available command-line parameters are displayed when invoking SQL*Plus with a dash “-” as the only parameter, as shown here:

```

C:\CERTDB>sqlplus -
Usage: SQLPLUS [ [<option>] [<logon>] [<start>] ]
where <option> ::= - | -? | [ [-M <o>] [-R <n>] [-S] ]
      <logon>  ::= <username>[/<password>][@<connect_string>] | / | /NOLOG
      <start>  ::= @<filename>[.<ext>] [<parameter> ...]
      "-" displays the usage syntax
      "-?" displays the SQL*Plus version banner
      "-M <o>" uses HTML markup options <o>

```

```
"-R <n>" uses restricted mode <n>  
"-S" uses silent mode
```



The complete list of SQL*Plus command-line options can be found in the SQL*Plus User's Guide and Reference in the Oracle documentation set, but some of the more important and interesting options are discussed here.

First, during invocation you have three elements that you can include: options, logon credentials, and a script or file to start when SQL*Plus is invoked. Options are command-line switches that enable you to format the output of a SQL query using HTML (the `-M` option), restrict certain commands from being available when SQL*Plus is running to reduce its functionality (the `-R` option), or have SQL*Plus run in silent mode so that it does not echo back results to the user or display its banner (the `-S` option).

Logon credentials specify which user account will be used to connect to an instance, as well as the instance to connect to (the default instance specified by the `ORACLE_SID` environment variable if only a username and password are specified, or a full instance name conforming to Net8 requirements). You can also specify whether or not you want the connection attempt to log the user on to a database or not, with the `/NOLOG` logon option—the default is to try and log the user on. If you don't specify the username and password on the command line, you are prompted for it, as shown here:

```
C:\CERTDB>sqlplus  
  
SQL*Plus: Release 8.1.7.0.0 - Production on Wed Jul 25 10:24:31 2001  
  
(c) Copyright 2000 Oracle Corporation. All rights reserved.  
  
Enter user-name:
```

The final command-line parameter is the name of a SQL script to run, preceded by the `@` sign, as well as any optional parameters that the script may call for. If the script file ends in a `.SQL` extension, there is no need to specify it. A file ending in any extension besides `.SQL` must have the full filename passed as a parameter. All SQL script files are plain ASCII text files created with a text editor such as `vi` or `Notepad` and should not contain anything besides SQL and PL/SQL code, SQL*Plus commands, or comments.

For example, to invoke SQL*Plus from the command line connecting to the CERTDB instance as system with a password of `MANAGER`, and executing a script called `CRNEWTAB.SQL`, issue the following command:

```
C:\CERTDB>sqlplus system/manager@certdb.mars.bradsys.com @crnewtab  
  
SQL*Plus: Release 8.1.7.0.0 - Production on Wed Jul 25 11:36:06 2001
```

(c) Copyright 2000 Oracle Corporation. All rights reserved.

```
Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production
```

Table created.

```
Disconnected from Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production
```

C:\CERTDB>

SQL*Plus Worksheet

A graphical program that presents a somewhat different interface to SQL*Plus, but provides much the same functionality, is SQL*Plus Worksheet. As shown in Figure 2-6, the interface is a split-screen affair with the top of the screen being used to enter commands to be sent to the database and instance while the bottom half of the screen is the results. The lightning bolt symbol on the left of the SQL*Plus Worksheet is used to submit the highlighted commands on the top part of the screen (or everything if no text is highlighted) to the server, where the results are displayed on the bottom.

Aside from this difference in presentation, the functionality available in SQL*Plus Worksheet is about the same as SQL*Plus.



SQL*Plus Worksheet is also designed to work with Oracle Enterprise Manager and the Management Server. For this reason, it has a greater amount of overhead than SQL*Plus or Server Manager line mode. Performance of the tool is far worse than SQL*Plus, so many DBAs (this author included) do not use it on a regular basis.

Other command-line utilities

Oracle also includes a number of other command-line utilities that are used to perform specific tasks and not necessarily for general administration. These include tools to transfer data between Oracle databases, load data from external sources into Oracle, and create the necessary services on Windows NT/2000 computers.

Import and Export

The Import utility (IMP) and the Export utility (EXP) are used to transfer data from one Oracle database to another. They can also be used to move objects and data from one schema to a different schema in the same or a different database. You can export and import an entire database (assuming you have the proper privileges), an entire schema, or individual objects such as tables and views.

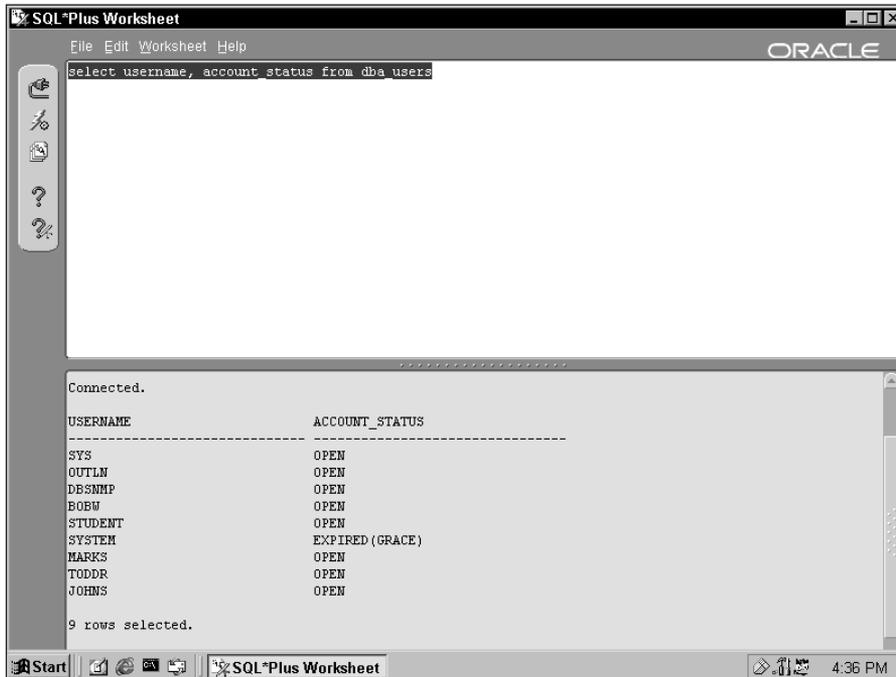


Figure 2-6: SQL*Plus Worksheet presents a split-screen interface to execute SQL statements against an Oracle instance.

Cross-Reference

The Import and Export utilities are covered in detail in Chapter 15.

SQL*Loader

SQL*Loader is a command-line utility that enables you to import data from external sources into one or more Oracle tables. The idea behind this utility is to enable you to have a mechanism whereby data from other databases, such as IBM DB2, Sybase, Microsoft SQL Server, and others, that has been converted into ASCII format can be loaded into an Oracle database with conditional logic applied during the load.

SQL*Loader can only read ASCII files and requires a structure of some kind to exist within the file in order to determine what a record and a field (that is, row and column) are.

Cross-Reference

Using SQL*Loader to load data into Oracle databases is covered in greater detail in Chapter 14.

ORADIM

On Windows NT/2000 environments, before you can start an Oracle instance, you need to create services on the host computer that will run in the background to support the instance. Furthermore, if you want to delete a database and instance, you also need a clean way to remove the services that support it. The ORADIM

utility is used to create, modify, and delete services that are used for databases you create on Windows NT/2000 platforms.



The ORADIM utility and the creation of a database are covered in more detail in Chapter 4.

Oracle Enterprise Manager

Objective

List the main components of Oracle Enterprise Manager and their uses

If you are comfortable with the command line and familiar with all the commands you need to administer your Oracle databases and instances, you may not necessarily have a need for a graphical tool that hides some of what is actually happening under the covers. The ideal configuration for administration is to use command-line tools for quick and simple administration where appropriate, and graphical tools, like Oracle Enterprise Manager (OEM), for a more complete picture of what you have in your environment. In other words, both have a place and can complement each other.



The Oracle8i: Architecture and Administration exam tends to focus on how to perform most administrative tasks using SQL commands instead of the GUI tools (and this book tends to focus on the command line accordingly). An understanding of which GUI tools are available and how to configure Oracle Enterprise Manager is required for the exam.

Oracle Enterprise Manager, or OEM, is a series of applications and services that work together to provide a seamless and widespread administrative framework for Oracle databases, instances, and other Oracle services such as Oracle Application Server. By providing a single point of administration for all elements of Oracle in an enterprise, it can make the job of managing a large number of databases in a large organization easier.

Oracle Enterprise Manager components are written in Java to allow for easier portability between platforms supported by Oracle. Because Java is supported on many platforms, and because Oracle strictly adheres to the Sun Microsystems Java standards, this enables OEM to work the same way on every platform where it is available. Furthermore, the interface for each of the OEM tools is virtually identical on Windows and UNIX-based systems. The one downside of Java is speed — OEM is not the most lightning-fast set of tools that Oracle provides.

OEM management packs

When you purchase Oracle8i and perform a typical installation on your server, you will also be prompted to install OEM. This installation does not include all of the various options available, but only the Database Management Pack. The Database Management Pack includes a number of components including Security Manager (for adding, modifying, and removing users, roles, and privileges), Storage Manager (for administering tablespaces, rollback segments, redo log files, datafiles, and

archived redo log files), Instance Manager (for starting and stopping instances and modifying initialization parameters), and Schema Manager (for managing database objects such as tables, views, procedures, and others). A number of other tools and wizards for performing database backups, loading data, and so on are also included. The Database Management Pack is sufficient for doing basic administration of any Oracle database and is shipped free of charge with any version of Oracle.

OEM also includes a number of other management packs that need to be purchased separately and can provide greater functionality and make OEM more useful. These include

- ♦ **Diagnostic Pack** — The Diagnostic Pack includes Oracle Trace and Data Viewer (used to create, schedule, manage, and view tracing information from the database, Net8, and other Oracle components), Performance Manager (a tool to present a graphical view of performance statistics for a database or instance), Capacity Planner (used to plan storage and other requirements in the future by collecting and analyzing statistics of database activity), TopSessions (a real-time session monitoring tool to determine which user sessions are consuming the most memory, I/O, and other resources), and Advanced Events (a collection of additional event objects that can be monitored using the Enterprise Manager console).
- ♦ **Tuning Pack** — The Tuning Pack includes several tools to help you tweak database performance. These include Oracle Expert (an “intelligent” wizard that can recommend changes to database object structure and instance settings to increase database performance by collecting environment, workload, initialization parameter, and database structure information), Tablespace Manager (used to detect and fix and space management problems that may exist in the database by reorganizing database objects), and SQL Analyze (used to collect, analyze and edit SQL statements with a view to improving database performance).
- ♦ **Change Management Pack** — The Change Management Pack includes tools to help you compare database objects in different databases by creating and comparing baselines (Database Capture for baseline generation, and Database Diff for comparing baselines), make rapid changes to one object in a database (Database Quick Change) or one or more objects in one or more databases (Database Alter), as well as propagate objects from one or more schemas in one database to one or more schemas in one or more databases (Database Propagate). It also includes a central management tool for administering change management called Plan Manager.
- ♦ **Application Management Pack** — Used to administer other Oracle applications such as WebForm servers, and Workflow subsystems.

Oracle Enterprise Manager architecture

The first step in understanding how useful OEM can be is to understand the pieces that make it up. It is not a single interface, but rather a set of components that all work together on different machines to provide the enterprise-wide picture needed. Figure 2-7 presents the three-tier Oracle Enterprise Manager architecture that was

introduced with Oracle Enterprise Manager 2.0 and is in use today. The current release (at the time of this writing) is OEM 2.2, which includes a number of improvements but does not alter the basic architecture of the product.

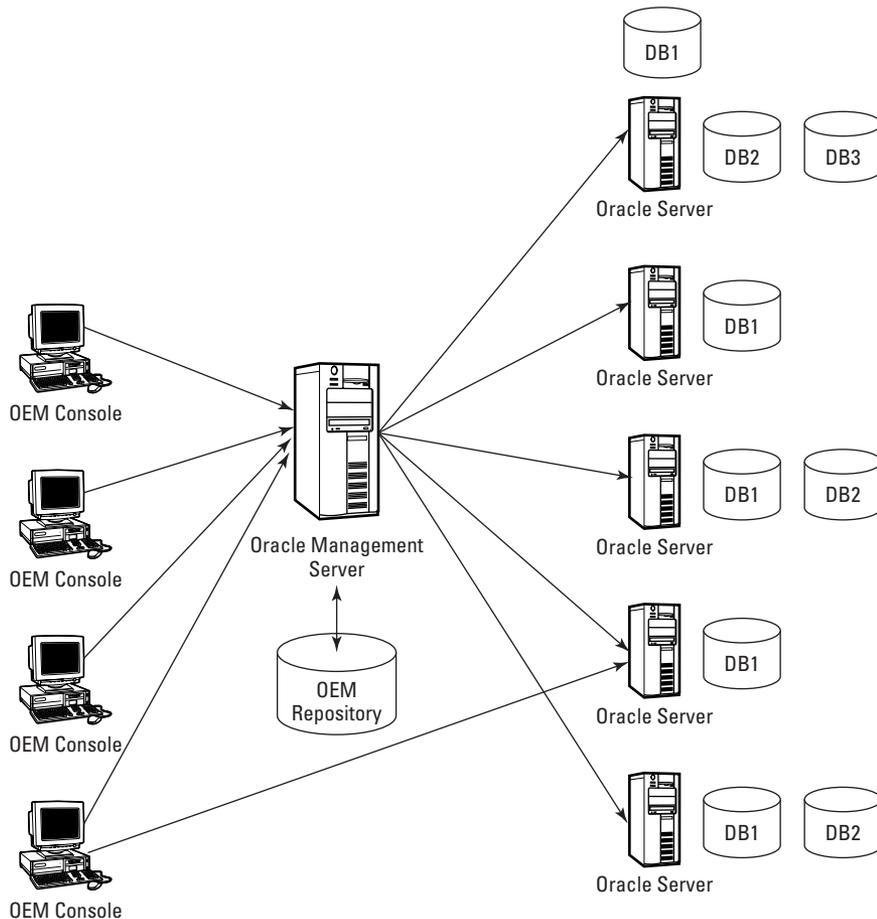


Figure 2-7: Oracle Enterprise Manager consists of a client component (the OEM console), the Management Server, and the databases to be administered on servers where the Oracle Intelligent Agent is started.

The first component of the OEM three-tier architecture is the client. The client, typically the OEM console, is any tool that connects to the Oracle Management Server to request services and authenticate as a user. Each time a client application (OEM console, DBA Studio, SQL*Plus Worksheet, and so forth) is started, you will be prompted to connect to an Oracle Management Server (OMS) and provide the username and password of an administrator configured on the Management Server. This

is not an Oracle user, but rather an OMS user that has been created when the Management Server was installed, or afterwards to grant individuals the ability to perform enterprise-wide administration.

As you can see from the instance of DBA Studio shown in Figure 2-7, the client may not necessarily need to connect to a Management Server in order to perform administration of a database. Clients that do not need to connect to OMS include DBA Studio, SQL*Plus Worksheet, Security Manager, Instance Manager, Storage Manager, and Schema Manager. The OEM console, however, always requires that a Management Server connection be established first.

The heart of the OEM three-tier architecture is the Oracle Management Server, which itself connects to an Enterprise Manager Repository stored in an Oracle database, either on the same server that is running OMS or on another computer. The main purpose of OMS is to provide a centralized repository of databases and nodes (that is, servers) to manage, as well as jobs and events to be monitored. OMS also keeps track of which individuals have been granted privileges to perform administrative tasks and to what extent (that is, can they create other users or jobs, receive event notifications, and so on). Having a shared repository enables you to configure administrative tasks on one location and then have multiple users perform these operations from their own consoles. Being able to assign permissions to users also gives you greater control over the capabilities that are granted to users.

A single Oracle Management Server can communicate to one and only one repository, but a single repository can be used by more than one Oracle Management Server. This is because the repository is simply a database and the OMS is considered a database client connection. Having more than one OMS connect to the same repository enables you to scale enterprise-wide management better by having certain administrative tasks be run on one OMS while another OMS can be used for a different purpose, thereby better balancing the workload.

If you dig deeper into what actually makes up the Oracle Management Server, you find that it is composed of a set of services that interact with the repository and a component on each database server (or node) that is being managed, called the Oracle Intelligent Agent. The services that exist on each Management Server include

- ♦ **Discovery Service** — The Discovery Service is used to scan nodes on the network to determine which instances and databases reside on them. The Discovery Service queries the Oracle Intelligent Agent on each node contacted for a list of databases and instances and then adds them to the list of objects that could be managed by OEM and the Oracle Management Server. When you first invoke Oracle Enterprise Manager, you are always asked to perform node discovery to determine which databases are available for you to administer.

- ♦ **Job Scheduling Service** — The Job Scheduling Service is responsible for storing the names and definitions of jobs that you want to execute on specific nodes and/or databases in the repository, and then submitting them, on the schedule set, to the Oracle Intelligent Agent of the appropriate node for execution. The Job Scheduling Service also stores job success, failure, and error information in the repository for review by administrators, and also can be configured to send an e-mail or pager notification on job failure, completion, or both.
- ♦ **Event Management Service** — The Event Management Service enables you to configure certain events, such as unexpected instance or node shutdown, or the crossing of a threshold limit, to be monitored and notification of the event taking place being sent to an administrator through e-mail or pager notification. You can also configure a “fixit” job to be automatically launched to correct the problem and thereby enable you to sleep at 3 a.m., because most problems that require intervention usually happen just when you want to get some sleep.
- ♦ **Security Service** — The Security Service ensures that, because OMS now includes a repository shared among potentially many users, only those users that have been granted permissions to view or manage jobs, events, nodes, databases, and so forth do so. It checks permissions before a user attempts to perform an action and also authenticates users who request to connect to the Management Server.

Configuring the Oracle Management Server

In order to make use of Oracle Enterprise Manager console, you must first configure an Oracle Management Server on one node and create the repository in one of your existing databases. Depending on the number of nodes and databases to be monitored, you may want to configure the Management Server and have the database holding the repository on the same node. In fact, most implementations to-date tend to take this approach because the overhead of the repository on the database is not too great, and the OMS itself does not take up too many resources. If you need to increase the performance of the OMS, simply create another one and point it to the original repository.

The following step-by-step shows you how to create a repository and configure an Oracle Management Server on a Windows NT/2000 computer. If you are using Linux, or another UNIX variant, the procedure will be similar, but consult your platform-specific documentation for any caveats.

STEP BY STEP: Configuring the Oracle Management Server and Creating a Repository

1. Make sure that you have installed Oracle8i on a computer and created at least one database. If you choose the typical installation when running the Oracle Universal Installer, choose the option to create a starter database and use the database it creates.

Make sure that you have also selected to install Oracle Enterprise Manager as part of the installation.

2. From the Start Menu, navigate to your Oracle home and then select Enterprise Manager, and then Configuration Assistant, as shown in Figure 2-8.

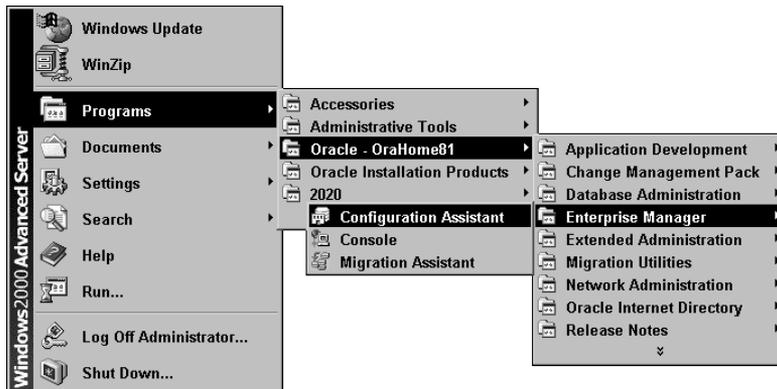


Figure 2-8: The Configuration Assistant in the Enterprise Manager program group is used to configure the Oracle Management Server.

3. On the main Configuration Assistant screen, select “Create a new repository,” as shown in Figure 2-9. Note that you can also use the Configuration Assistant to drop a repository you no longer use, modify OMS startup settings, or upgrade a repository to a more current version (for example, 2.0 to 2.2). Click Next when you have made your selection.

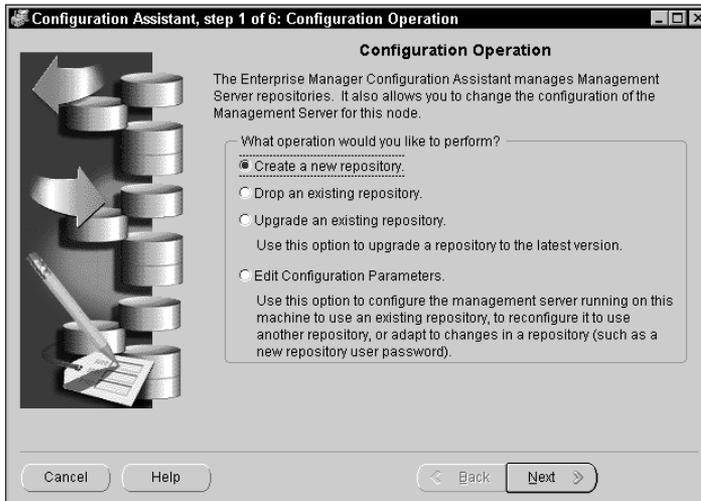


Figure 2-9: The Configuration Assistant main startup screen presents a list of actions that can be performed. Select “Create new repository” to create a new repository.

4. On the next screen, you need to provide the username, password, and Net8 connect string of the database that will host the repository, as shown in Figure 2-10. Make sure that the user you connect as has been assigned the DBA role and has full privileges on the database, because you may need to create a user account and/or tablespace as part of the configuration process.

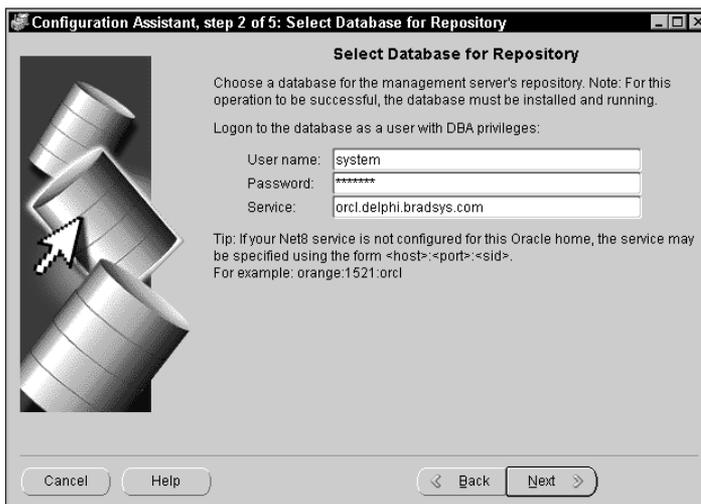
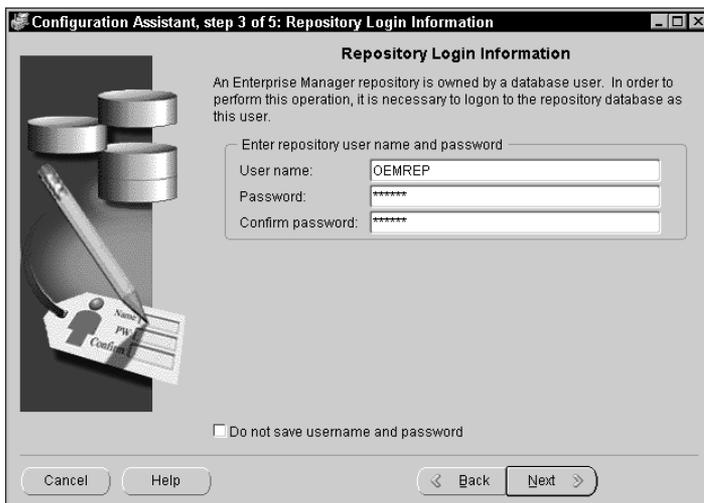


Figure 2-10: You must provide a username and password, as well as the Net8 instance name where the repository is to be stored. The username provided must have the DBA role assigned.

5. After clicking Next, you are prompted for the name of the user who will be the owner of the repository. The user does not need to exist in the database, because the account will be created by the Configuration Assistant. Also, specify a password and confirm the password where prompted, as shown in Figure 2-11.



The screenshot shows a window titled "Configuration Assistant, step 3 of 5: Repository Login Information". The window contains a section titled "Repository Login Information" with the following text: "An Enterprise Manager repository is owned by a database user. In order to perform this operation, it is necessary to login to the repository database as this user." Below this text is a form with three input fields: "User name:" with the value "OEMREP", "Password:" with "*****", and "Confirm password:" with "*****". To the left of the form is an illustration of a person's ID card with fields for "Name", "PW", and "Confirm". Below the form is a checkbox labeled "Do not save username and password" which is currently unchecked. At the bottom of the window are buttons for "Cancel", "Help", "Back", and "Next".

Figure 2-11: Choose a username and password for the user in whose schema the repository will be created and then click Next.

By default, the Configuration Assistant stores the username and password in the OMS configuration to ensure a smooth connection. If you do not want the password saved (for security reasons), you can check the box selected. In most cases, it is recommended to leave the box unchecked because the OMS will not start if it cannot connect to the repository.

6. If the user account does not exist, you are prompted for the default and temporary tablespaces to be assigned to the user. The tablespaces must already exist in the database and should be created prior to running the Configuration Assistant. You should create a separate tablespace to host the repository. Select the appropriate tablespaces and click Next, as shown in Figure 2-12.



Creating of tablespaces is covered in Chapter 8. Configuring a default and temporary tablespace for a user, and its meaning, is covered in Chapter 17.

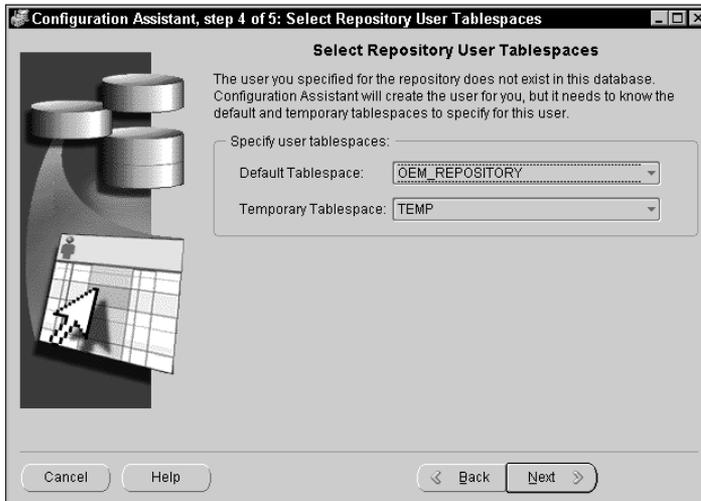


Figure 2-12: The Configuration Assistant prompts you to select a default and temporary tablespace for the repository owner. Use the drop-down list boxes to make your selections and click Next to continue.

7. The Create Repository Summary screen, shown in Figure 2-13, displays a list of the tasks that are performed by the Configuration Assistant. Click on Finish to start the process of creating the repository and configuring the Oracle Management Server.

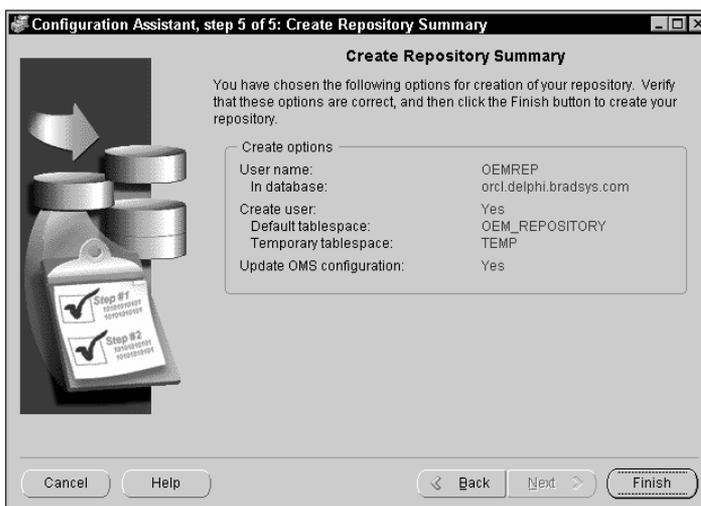


Figure 2-13: The Configuration Assistant main startup screen presents a list of actions that can be performed. Select “Create new repository” to create a new repository.

8. The Configuration Assistant Progress dialog box appears indicating the status and progress of the configuration, as shown in Figure 2-14. If you want more detailed information on what is happening during the process, you can click Show Details, which presents detailed progress information on the text box on the right, as shown in Figure 2-15.

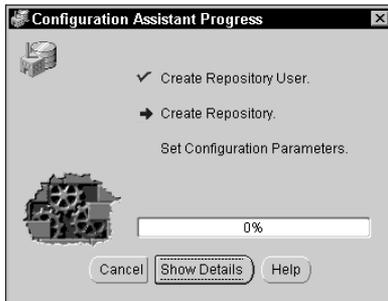


Figure 2-14: The Configuration Assistant Progress dialog box displays the status of the repository and OMS configuration.

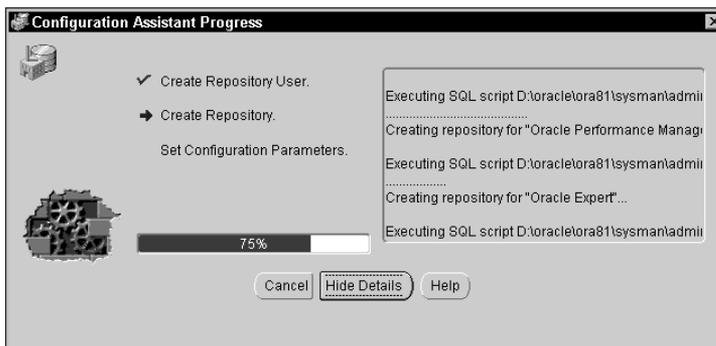


Figure 2-15: Selecting Show Details on the Configuration Assistant Progress dialog box presents this screen with a text box tracing each step of execution.

9. After the “Processing Complete” message indicating 100 percent of the work is complete is displayed, click Close. You have now created the repository and configured the local OMS to use it.

Once you have created the repository and configured the Oracle Management Server, ensure that the service is started by using Control Panel/Services (on Windows NT 4.0) or the Services MMC snap-in (in Windows 2000) to start the service. If the Management Server service is not started on the computer, any

connection attempts to it will fail. Also, ensure that the Oracle Intelligent Agent service is started. On UNIX-based computers, your platform-specific documentation provides information how to start the background daemons that are used to run the agent and OMS on the computer.

The next step in the successful use of Oracle Enterprise Manager is to start the OEM console and perform node discovery as presented in the following step by step.

STEP BY STEP: Starting and Using the OEM Console

1. From the Enterprise Manager program group, select Console to invoke the OEM console.
2. When prompted for a username and password, and management server to connect to, enter **sysman** for the username and **oem_temp** for the password, and select Browse to locate the server you configured the repository on in the previous step by step, as shown in Figure 2-16. Click OK to connect to the Management Server.



Figure 2-16: Provide the username and password, as well as Oracle Management Server to connect to, when prompted.

The first time that the OEM console is invoked to connect to the Management Server, you must connect with the default superuser and password combination outlined above. Later you will create additional users and change the default password.

If the Oracle Management Server is not started or the name provided for it is incorrect, a dialog box similar to Figure 2-17 is displayed. Correct the error and try again.

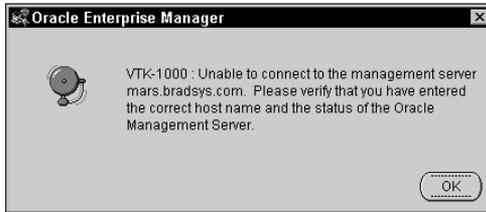


Figure 2-17: This error indicates that the Management Server specified cannot be contacted. Correct the error and try again.

3. If this is the first time that you have connected to the OMS, you are prompted to enter a new password for the *sysman* superuser, as shown in Figure 2-18. Provide the new password and confirm it, and then click Change to continue.



Figure 2-18: The first time you connect to the OMS, you will need to change the password for the *sysman* user.

4. You will then be presented with the main OEM console screen, as shown in Figure 2-19. The screen is divided into four parts.

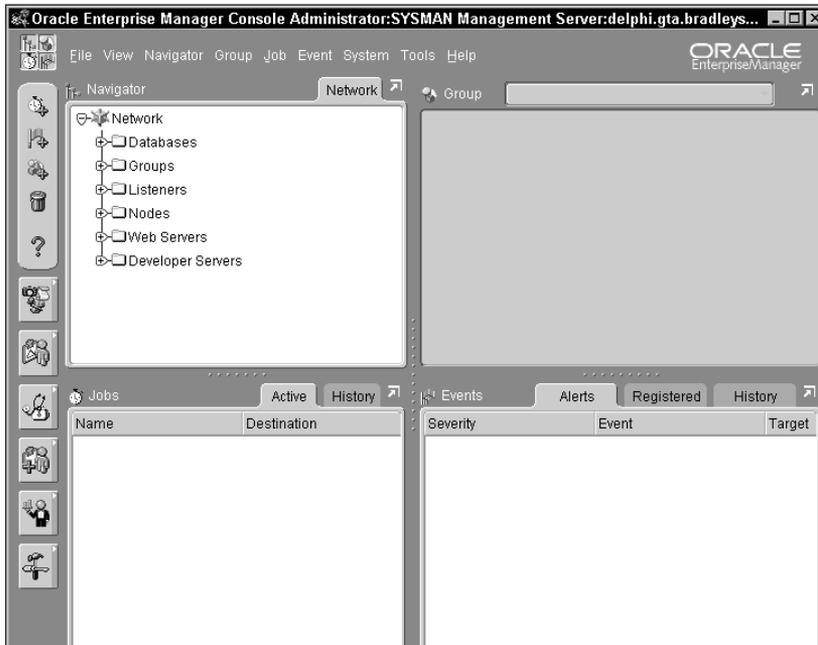


Figure 2-19: The main OEM console screen consists of navigator, group (map), job, and event panes.

Starting with the top left, you have what is called the *navigator* pane, which enables you to navigate the different databases, nodes (servers), listeners, Web servers, and other Oracle objects that can be managed by OEM.

The top-right pane is known as the group (or map) pane and can be used to diagrammatically show the location of nodes and databases, and other objects available in the enterprise. You can use a map that is any bitmap image file that you create to show the location of servers. Oracle provides some default maps for different parts of the world like Europe, the U.S., and the world itself.

The bottom-right pane is the event system pane, where the list of events configured on the Management Server is displayed, as well as any alerts that may exist. Any events you configure, register, or assign to be monitored on a database or computer are shown here.

The bottom-left pane is the job system pane, where the list of configured jobs and their execution status information are displayed. Jobs can be created by using a wizard that OEM can invoke, or by using the Tool (Task) Control Language (TCL) that the jobs are created in.

The OEM console also includes a menu where you can invoke other tools, configure users and preferences, get help on how to use OEM, and much more.

5. The first time you invoke OEM, it automatically registers all databases that exist on the same computer as the OMS. If you want to register additional nodes and databases for administration, select Discover Nodes from the Navigator menu, as shown in Figure 2-20. This invokes the Discovery Wizard and presents the introductory screen for the wizard, as shown in Figure 2-21 on which you click Next to start the discovery process.

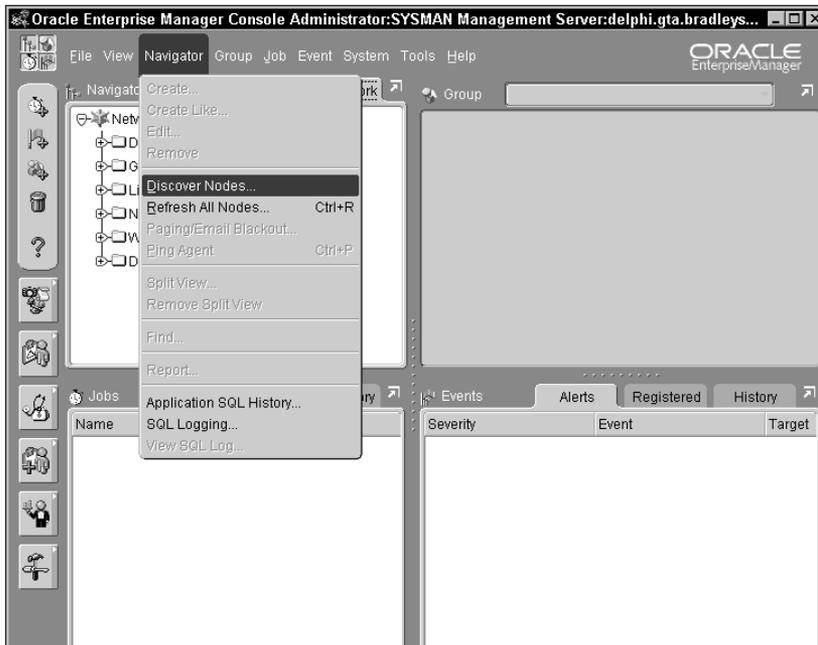


Figure 2-20: Select Discover Nodes from the Navigator menu to add databases and other servers to the repository.

6. On the Specify Node screen shown in Figure 2-22, enter the IP address, DNS name, or NetBIOS name of the computers whose Oracle databases and services you want to discover. Make sure that the Oracle Intelligent Agent is started on each node provided. Enter a single node per line, or choose Import to import a text file with a list of nodes containing Oracle databases that was created for you. Click Next to continue.



Figure 2-21: Click Next on the Discovery Wizard introductory screen to start the discovery process.



Figure 2-22: Provide the names of nodes you want to discover on the Specify Nodes screen.

7. The Discovery Wizard resolves the name of the nodes you specified and contacts the Oracle Intelligent Agent to provide it a list of databases and services on the node. When the process is complete, a checkmark appears next to the node discovered indicating success, as shown in Figure 2-23.

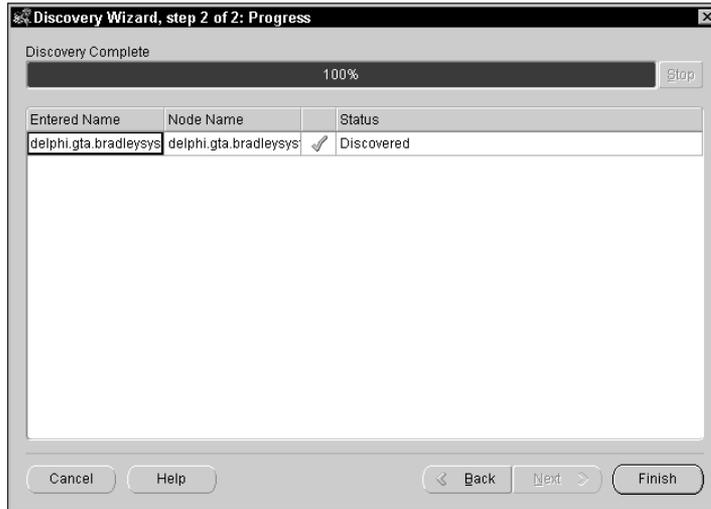


Figure 2-23: When discovery is complete, click Finish to exit the Discovery Wizard.

If you receive an error and a red X indicating that the node could not be discovered, determine the cause of the error, fix it, and try again.

8. On the Discovery Results dialog box, shown in Figure 2-24, click OK to acknowledge the results presented.

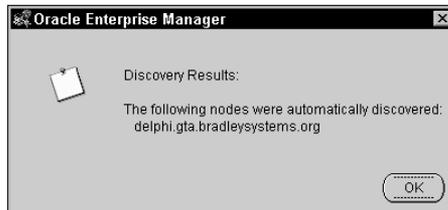


Figure 2-24: Click OK on the Discovery Results dialog box to complete the process.

9. The Oracle Enterprise Manager navigator pane will be updated with the results of discovery, as shown in Figure 2-25. Expand the nodes or databases to get more information on what can be administered.

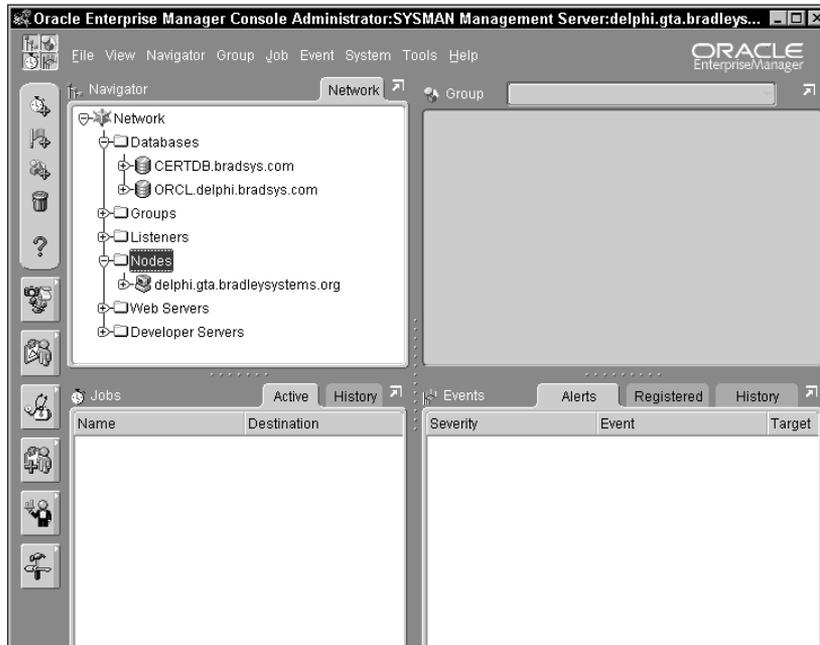


Figure 2-25: The OEM Console navigator pane adds the discovered nodes and databases to the list.



The Oracle8i: Architecture and Administration exam tests your basic knowledge of how to configure and use Oracle Enterprise Manager. You probably won't be asked detailed questions on how OEM works or how to create jobs and events, but rather questions on the capabilities of OEM.

Other graphical administration tools

Besides OEM, Oracle also provides a number of other administrative tools with a graphical user interface. These tend to be oriented to performing a specific task and include

- ♦ **Oracle Database Assistant**—The Oracle Database Assistant is a wizard-like program that guides you through the process of creating, modifying, and dropping databases.



How to use the Oracle Database Assistant to create a database is dealt with in Chapter 4.

- ♦ **DBA Studio**—DBA Studio is a tool very similar to Oracle Enterprise Manager, except that it enables you to connect directly to an instance or use the Oracle Management Server. When invoking DBA Studio to connect to a single instance, you are able to manage only that instance's objects; when connecting to OMS, you are able to manage multiple nodes and instances. The interface for DBA Studio is similar to the navigator pane of the OEM console, with a Windows-like detail pane on the right side, as shown in Figure 2-26.

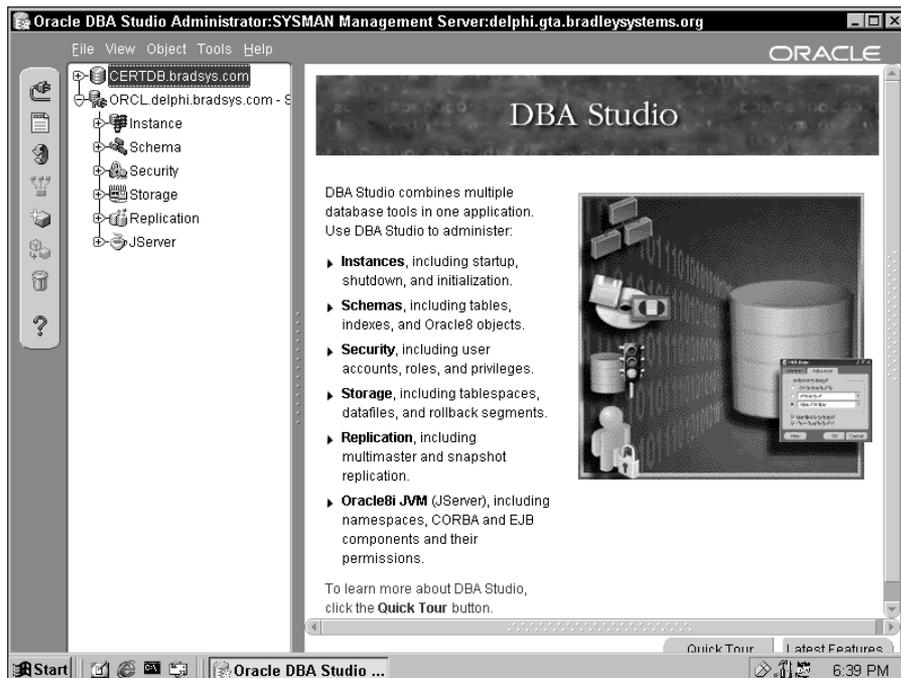


Figure 2-26: DBA Studio presents a Windows-style interface similar to the navigator pane in the OEM Console.

- ♦ **Net8 Assistant**—The Net8 Assistant is used to configure network settings for your nodes, including listeners, Oracle Names servers, and other elements. Its operation is beyond the scope of this book and the Oracle8i: Architecture and Administration exam.
- ♦ **Oracle Data Migration Assistant**—This wizard-like utility is helpful in the migration of databases from previous versions of Oracle to Oracle8i.



The Oracle Data Migration Assistant should not be confused with the Migration Assistant under the Enterprise Manager program group, which can be used to migrate previous versions of the OEM repository to the most current release.

While there are still other tools that Oracle provides to manage spatial and text data, configure replication, deal with enterprise-wide directory services, and so on, the reality is that these have a very particular purpose beyond the scope of this book—and you won't be tested on them when you write the Oracle8i: Architecture and Administration exam.

Key Point Summary

In preparing for the Oracle8i DBA: Architecture and Administration exam, please keep these points in mind regarding administrative tools and configuring privileged users:

- ♦ The Oracle Universal Installer is a graphical tool common to most Oracle platforms that can be used to install, remove, and change the configuration of installed Oracle products.
- ♦ The Oracle Universal Installer can be invoked interactively or from the command line and can run in silent mode without any user interaction.
- ♦ When installing Oracle software and creating databases, the Oracle Universal Installer adheres to Optimal Flexible Architecture, or OFA. OFA is a structure that was developed by Oracle consulting services for installing Oracle software and creating a file and directory structure to improve performance and ease management of Oracle databases.
- ♦ In order to create a database, or start and stop an Oracle instance, you must be authenticated as a privileged user.
- ♦ Privileged users can be authenticated using either the operating system or a password file.
- ♦ Configuring privileged users to be authenticated by the operating system depends upon the O/S being used, but usually involves an O/S user being a member of an O/S group that has been granted special privileges by the Oracle Universal Installer during the installation of the Oracle software. The `REMOTE_LOGIN_PASSWORDFILE=NONE` Oracle initialization parameter also needs to be set for the instance that supports O/S authentication of privileged users.
- ♦ The `ORAPWD` utility is used to create a password file and specify the number of entries it will support. It also sets the password for the special user `INTERNAL`, which exists for backward compatibility and should not be used as it will be desupported in future releases.

- ♦ If using password file authentication, you need to specify a value of EXCLUSIVE or SHARED for the REMOTE_LOGIN_PASSWORDFILE Oracle initialization parameter.
- ♦ Two special roles determine what a privileged user can do — SYSOPER (OSOPER when using O/S authentication) and SYSDBA (OSDBA when using O/S authentication). SYSDBA is the role with more privileges, because it also enables the user holding it to create the database and perform incomplete recovery.
- ♦ Oracle has a number of command-line tools for administration, including Server Manager line mode, SQL*Plus, and SQL*Plus Worksheet. Other tools, such as Import, Export, SQL*Loader, and so on also can be used to perform specific tasks.
- ♦ Oracle Enterprise Manager is the primary graphical tool for administering Oracle in an enterprise-wide environment. It has a three-tier structure consisting of a client (OEM Console), middle tier (the Oracle Management Server and Repository), and server (each Oracle node and database being managed).
- ♦ In order to use the OEM console, you must run the Configuration Assistant to create a repository and configure the management server. You then need to discover nodes and databases that you want to administer.
- ♦ The Oracle Management Server consists of a set of services for creating and managing security (Security Service), jobs and their execution (Job Scheduling Service), events and their monitoring (Event Service), and the discovery of nodes and services with cooperation from the Oracle Intelligent Agent on the node (Discovery Service).
- ♦ Other graphical tools are also provided, including DBA Studio (with similar functionality to the OEM console but a different interface), Database Configuration Assistant (for creating, altering, and dropping databases), Oracle Data Migration Assistant (for migrating from previous Oracle versions), and many others.



STUDY GUIDE

This chapter introduced new concepts to help you get started with Oracle8i Server. Now you should test your understanding by reviewing the assessment questions and performing the exercises below

Assessment Questions

1. You want to allow a Windows NT user called ToddR to be authenticated as a privileged user and be able to start up and shut down an instance called ORCL. Which two Windows NT local groups could you add ToddR to so that he may be able to perform these actions? (Choose two correct answers.)
 - A. ORA_PWFIL
 - B. ORA_ORCL_DBA
 - C. ORA_ORCL_OPER
 - D. ORA_ORCL_OSDBA
 - E. ORA_ORCL_OSOPER
2. You have configured operating system authentication for a database whose SID is ORCL, but it does not appear to be working. You suspect that it may be a problem with the Oracle initialization file. The file contains the following lines that you feel may be suspect:
 1. DB_NAME=ORCL
 2. INSTANCE_NAME=ORCL
 3. REMOTE_LOGIN_PASSWORDFILE=SHARED
 4. OS_AUTHENT_PREFIX=OPS\$
 5. REMOTE_OS_AUTHENT=TRUE

Which line in the Oracle initialization file is the most likely to be causing problems? (Choose the best answer.)

- A. 1
- B. 2
- C. 3
- D. 4
- E. 5

3. What steps must you perform before you can invoke the Oracle Enterprise Manager console successfully? (Choose all correct answers.)
 - A. Start the Oracle Intelligent Agent on the Management Server computer.
 - B. Run the Configuration Assistant to create a repository.
 - C. Create a database in your organization.
 - D. Configure privileged users.
 - E. Create additional administrators on the Oracle Management Server.

4. What are the services that make up the Oracle Management Server? (Choose all correct answers.)
 - A. The Security Service
 - B. The Oracle Intelligent Agent Service
 - C. The Schema Service
 - D. The Event Service
 - E. The Job Scheduling Service

5. If you have configured operating system authentication properly for your database, and are currently logged in as a user who has been granted all privileged operations except the ability to create a database, which of the following connect strings can you use in SQL*Plus to successfully connect to the instance? (Choose the best answer.)
 - A. CONNECT / AS SYSDBA
 - B. CONNECT / AS OSDBA
 - C. CONNECT / AS INTERNAL
 - D. CONNECT / AS SYSOPER
 - E. CONNECT / AS OSOPER

6. Which of the following is the default superuser account for an Oracle Management Server? (Choose the best answer.)
 - A. SYSTEM
 - B. SYS
 - C. SYSMAN
 - D. INTERNAL
 - E. MANAGER

7. In order for jobs to successfully run on a node that has been discovered and added to the repository, what needs to be started and running on the node? (Choose the best answer.)
- A. The database
 - B. The Management Server
 - C. The instance
 - D. The Intelligent Agent
 - E. The Discovery Service
8. Which of the following OEM components is provided by default when you install Oracle Enterprise Manager? (Choose the best answer.)
- A. The Database Management Pack
 - B. The Tuning Pack
 - C. The Management Server Pack
 - D. The Diagnostics Pack
 - E. The Change Management Pack
9. Which language is Oracle Enterprise Manager and the Oracle Universal Installer written in for ease of portability between platforms? (Choose the best answer.)
- A. C++
 - B. Visual Basic
 - C. Java
 - D. Assembler
 - E. PL/SQL
10. Which of the following can be used to create a password file? (Choose all correct answers.)
- A. ORAPWD Utility
 - B. Oracle Enterprise Manager
 - C. Database Configuration Assistant
 - D. ORADIM Utility
 - E. Server Manager line mode

Scenarios

1. You need to allow several of your fellow DBAs the ability to administer instances on your Windows NT–based Oracle server. The instances that are on the server have SIDs of ORCL, CERTDB, ORDER, ACCNTG, CUSTSRV, and TEST. All of the DBAs should be privileged users of the TEST and CERTDB databases, but only a select group should be able to perform privileged operations on the ORDER, ACCTG, and CUSTSRV databases — with a different group of admins for each one. No one, other than yourself, should have the ability to perform privileged operations on all of the databases on the Windows NT server.

How would you configure operating system authentication to support these requirements? Outline your strategy.

2. You need to deploy Oracle on ten UNIX and Windows 2000–based servers. The installation on each of the UNIX and Windows 2000 computers should be identical and automatic without user intervention. Once Oracle has been installed on each computer, you also want to configure the ability to use Oracle Enterprise Manager to centrally administer all of the computers running Oracle. You already have a repository and Oracle Management Server configured on an existing Windows NT–based computer.

How would you perform the deployment of Oracle to the ten computers?

How would you ensure that each of the new servers could be added to the repository, and how would you add them to the repository?

Lab Exercises

Lab 2-1 Installing Oracle

1. Insert the CD-ROM for your platform version of Oracle in the CD-ROM drive.
2. If you have AutoPlay configured, the Oracle Universal Installer should start automatically. If not, navigate to the root of the CD-ROM drive and invoke the setup program for your platform (SETUP.EXE for Windows-based platforms).
3. Select Install/Deinstall Products from the menu initially presented to launch the Oracle Universal Installer.
4. Follow the prompts to perform a Typical installation.
5. When prompted if you want to install a starter database, make sure you agree to the installation of a starter database, and when prompted for a SID choose the name ORCL.
6. When the installation completes, reboot your computer (optional, but highly recommended).

Lab 2-2 Configuring the Oracle Management Server

1. Make sure that you have installed Oracle8i on your computer and created the starter ORCL database in Lab 2-1. Make sure that you have also selected to install Oracle Enterprise Manager as part of the installation.
2. From the Start Menu, navigate to your Oracle home and then select Enterprise Manager, and then Configuration Assistant.
3. On the main Configuration Assistant screen, select “Create a new repository” and click Next.
4. On the next screen, you enter **system** for the username, **manager** for the password, and **ORCL** as the Service, and click Next.
5. Enter **OEMREP** as the name of the user that will be the owner of the repository and **oracle** for the password, then confirm password and click Next.
6. When prompted for the default and temporary tablespaces to be assigned to the user, accept the selections presented and click Next.
7. Review the information presented on the Create Repository Summary screen, and click Finish to start the process of creating the repository and configuring the Oracle Management Server.
8. After the “Processing Complete” message indicated 100 percent of the work is complete is displayed, click Close. You have now created the repository and configured the local OMS to use it.

Lab 2-3 Starting OEM for the First Time

1. From the Enterprise Manager program group, select Console to invoke the OEM console.
2. When prompted for a username and password, and management server to connect to, enter **sysman** for the username, **oem_temp** for the password, and select your computer name as the management server, or click the Browse button to locate the management server.
3. As this is the first time that you have connected to the OMS, select a new password for the *sysman* superuser and confirm it, and then click Change to continue.
4. When presented with the main OEM console screen, explore the different panes and menu options. Do not save any of your changes as this may disrupt future labs.
5. If you want to register additional nodes and databases for administration, select Discover Nodes from the Navigator menu to invoke the Discovery Wizard. Follow the prompts to discover additional nodes.
6. When you are done, exit the OEM Console.

Answers to Chapter Questions

Chapter Pre-Test

1. Oracle supports authentication of privileged users using either operating system authentication, where an operating system user is added to an O/S group that has been granted special privileges on Oracle, or by using password file authentication, where the list of Oracle users with SYSDBA or SYSOPER privileges is stored in a password file on disk. Password file authentication can be used for performing privileged operations remotely, whereas operating system authentication typically only works on the computer where the Oracle software resides, unless a secure channel has been established.
2. A user that has been granted the SYSDBA (or OSDBA) role for privileged operations can perform all of the same tasks that a user that has been granted the SYSOPER (or OSOPER) role can. SYSDBA users can also issue the CREATE DATABASE command or RECOVER DATABASE UNTIL command, which SYSOPER users are not permitted to execute.
3. Optimal Flexible Architecture is designed to facilitate a structure on your server that makes it easy to administer and facilitate the growth of databases. It includes provisions for hosting multiple databases on the same server, locating the datafiles for each database easily, and minimizing I/O contention by properly placing files on disks in such a way that files with high probability of contention are physically separated. The OFA also takes into account the addition of drives to more smoothly even out the I/O on the system.
4. You can use either Server Manager line mode or SQL*Plus to start and stop an instance. In versions of Oracle prior to 8i, Server Manager line mode was the preferred tool, but today you should use SQL*Plus. The reason for the switch is that all of the functionality that existed only in Server Manager line mode is now also available in SQL*Plus, and Server Manager line mode will not be supported in future releases of Oracle.
5. Oracle Enterprise Manager has a three-tier architecture with the client software (OEM console, for example) running on a database administrator's computer, and the Oracle Management Server and the OEM Repository running on another server computer. The Oracle Management Server connects to many Oracle database servers that run the Oracle Intelligent Agent to facilitate administration. This architecture enables you to add Management Servers to distribute the workload, and scale administration to very high levels to suit any organizational requirement.
6. Oracle graphical administrative tools, with the exception of the OEM console, can connect directly to an instance and bypass the Oracle Management Server. Doing so will not allow you to configure and manage jobs or events, but it lets you perform most other administrative tasks. DBA Studio, Security

Manager, Schema Manager, Storage Manager, SQL*Plus Worksheet, and Instance Manager can also bypass the Oracle Management Server. Of the available tools, DBA Studio provides the closest set of features to the OEM console, even when not connected to a Management Server.

7. In order to change the password of the user INTERNAL on a UNIX-based computer running Oracle8i, you need to run the ORAPWD utility to re-create the password file for the database. In the process of doing so, you will erase any entries that may exist in the current password file, so you should determine which users have been granted SYSDBA and SYSOPER privileges by querying the V\$PWFIL_E_USERS data dictionary view.
8. In order to completely configure operating system authentication of privileged users, you need to perform the necessary steps for your platform, and modify the Oracle initialization file for the affected instance to include the parameter and value REMOTE_LOGIN_PASSWORDFILE=NONE. If you do not set the value of this parameter to NONE, operating system authentication won't work.
9. Oracle Universal Installer enables you to perform a fully scripted installation of Oracle by passing it the name of a response file that you have created with all the necessary answers for the installation, as well as another parameter to run silently. The invocation of Oracle Universal Installer to perform a scripted silent installation on a Windows NT/2000 computer can be initiated with the following command (assuming D: is your CD-ROM drive):

```
D:\setup.exe -responsefile c:\myfile -silent
```
10. When you create a database, Oracle automatically creates a user called SYS with a password of CHANGE_ON_INSTALL. SYS owns the data dictionary and all objects therein. Another user called SYSTEM with a password of MANAGER is also created and granted full privileges on the data dictionary. Of the two, only the user SYS is added to the password file and can be authenticated as a privileged user if you have configured password file authentication for the database.

Assessment Questions

1. **B, C.** In a Windows NT/2000 environment you need to create local groups of the format ORA_*SID*_DBA or ORA_*SID*_OPER to which you would add an operating system user to grant the SYSDBA (OSDBA) or SYSOPER (OSOPER) privileges, respectively. Because you want ToddR to be able to start up and shut down the ORCL instance, you would create two local groups on the Windows NT computer, called ORA_ORCL_DBA and ORA_ORCL_OPER, and put ToddR's user account into one or both of the groups.
2. **C.** In order for operating system authentication to work properly, you need to ensure that the REMOTE_LOGIN_PASSWORDFILE parameter has a value of NONE configured in the Oracle initialization file. The file in this scenario had the value set to SHARED, which means password file authentication was taking place and not operating system authentication, thereby causing the problem.

3. **B, C.** In order to be able to successfully invoke the OEM console, you must first create a database in the organization that will host the OEM Repository, and run the Configuration Assistant to create the OEM Repository and configure the Oracle Management Server. In order to discover nodes, you also need to ensure that the Oracle Intelligent Agent is started on each node, include the OMS node, but this is not required to launch the OEM Console for the first time.
4. **A, D, E.** The Oracle Management Server is composed of four services: the Security Service, the Job Scheduling Service, the Event Service, and the Discovery Service. While the various services of OMS rely upon the Oracle Intelligent Agent, it is a separate component from the Oracle Management Server.
5. **D.** If you have been granted all privileged operations except the ability to create a database, you have been granted the equivalent of the SYSOPER role: OSOPER. This means that at the OS level your user account has been added to a group mapped to the role and you can connect as a privileged user to the instance using SQL*Plus by issuing the command “CONNECT / AS SYSOPER”.
6. **C.** The default superuser account on an Oracle Management Server is SYSMAN with an initial password of OEM_TEMP.
7. **D.** The Oracle Intelligent Agent is used by the Oracle Management Server to execute jobs and monitor events on a node that has been discovered.
8. **A.** The Database Management Pack is the only component that is provided free of charge when you purchase Oracle8i. You can install the other packs available (Tuning Pack, Diagnostics Pack, or Change Management Pack) by purchasing them separately.
9. **C.** Oracle Universal Installer and the Oracle Enterprise Manager, as well as the majority of the Oracle graphical tools, are written in Java to make portability easier and the interface uniform on most platforms where Oracle is available.
10. **A, C, D.** You can use the ORAPWD utility to create a password file, as well as to re-create a password file. A password file is also created when you create a database using the Database Configuration Assistant, or when you create a service for an Oracle instance using the ORADIM utility.

Scenarios

1. In order to configure operating system authentication to support the requirements outlined, you would perform the following steps:

- i. Create Windows NT local groups on the computer, called ORA_DBA, ORA_CERTDB_DBA, ORA_CERTDB_OPER, ORA_ORDER_DBA, ORA_ORDER_OPER, ORA_ACCNTG_DBA, ORA_ACCNTG_OPER, ORA_CUSTSRV_DBA, ORA_CUSTSRV_OPER, ORA_TEST_DBA and ORA_TEST_OPER.
- ii. Add your user account to the ORA_DBA local group, which enables you to administer any database and instance on the computer.
- iii. Add each of the other DBAs' operating system user accounts to the appropriate local groups to satisfy their administrative requirements.
- iv. Modify the INIT.ORA file for each instance to include the following parameter and value:

```
REMOTE_LOGIN_PASSWORDFILE=NONE
```
- v. Stop and restart each instance.

2. In order to deploy Oracle on the ten computers, you would need to create a response file for each operating system in question — one for UNIX and one for Windows 2000. You would then invoke the Oracle Universal Installer from each computer, passing it the name of the response file with the `-responsefile` parameter, and including the `-silent` parameter to run without providing feedback. If you like, you can copy the Oracle installation files to a central network share and invoke the Oracle Universal Installer by connecting to a share.

Once you have installed Oracle on each of the ten computers, you need to make sure that the Oracle Intelligent Agent is configured to start automatically on each node whenever the computer is restarted. This enables the Discovery Service of the Oracle Management Server to discover all databases on each node when needed.

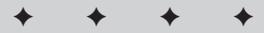
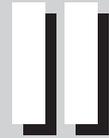
To add each database on each computer to the repository, you would invoke the OEM console and, from the Navigator menu, select Discover Nodes. When prompted, enter the names or IP addresses of the computers whose databases you want to discover. If the Intelligent Agent is running on each node, discovery should complete successfully.

Creating and Administering an Oracle Instance and Database

After receiving the 40,000-foot view of Oracle in Part I of this book, Part II takes you to the next logical progression — creating a database and learning more about what each of the database components is used for and how to administer them. This part of the book is divided into six chapters with emphasis on the physical structure of the database.

Chapter 3 shows startup and shutdown of an Oracle instance. You will learn that the `STARTUP` command takes parameters and that there are several modes of Oracle startup, each with a particular purpose. You will then learn how the `SHUTDOWN` command works and why Oracle has provided different shutdown modes to enable you to accomplish different tasks. To make all this work, you also need to learn how to create and configure a parameter file, and which parameters are key to the proper operation of an instance.

In Chapter 4, you will learn how to create an Oracle database. This is not necessarily as simple as it sounds since proper planning of operating system and file placement, as well as other issues is crucial. This chapter will point out some of the things you may need to do to ensure that the database you create performs well.



In This Part

Chapter 3

Managing an Oracle Instance

Chapter 4

Creating a Database

Chapter 5

Creating Data Dictionary Views and Standard Packages

Chapter 6

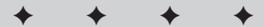
Maintaining the Control File

Chapter 7

Maintaining Redo Log Files

Chapter 8

Managing Tablespaces and Datafiles



Issuing a CREATE DATABASE statement is not the end of the database creation process. As you will learn in Chapter 5, additional steps need to be taken to ensure that the necessary features are properly installed, and database objects that assist in administration are created. This chapter will also introduce you to some key database objects that you, as a database administrator, may not deal with on a daily basis but need to be aware of.

The next three chapters (6, 7, and 8) provide a discussion on how to administer and maintain certain database files. In Chapter 6, you will be provided information on how a control file plays a key role in the database, and how to protect it so that your database does not suffer prolonged failure. Chapter 7 will discuss the best way to configure redo log files and the role these files play in the smooth operation and good recoverability of a database. Chapter 8 completes this part of the book by showing you how the datafiles of your database map to logical elements called tablespaces. You will be provided an overview of the Oracle database storage architecture and shown how to create and manage tablespaces and datafiles.

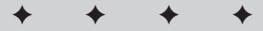
Managing an Oracle Instance

EXAM OBJECTIVES

- ◆ Managing an Oracle Instance
 - Create the parameter file
 - Start up an instance and open the database
 - Close a database and shut down the instance
 - Get and set parameter values
 - Manage sessions
 - Monitor the ALERT file and the trace files

3

CHAPTER



CHAPTER PRE-TEST

1. What is the syntax for the parameters in the INIT.ORA file?
2. Where are the stored configurations for an instance saved?
3. At what stage of opening the database is the control file read?
4. What command can you use in order to move the database from NOMOUNT state to MOUNT?
5. Which SHUTDOWN option enables you to immediately log users out when they complete their transactions?
6. What view can you query in order to find all dynamic performance views available in MOUNT state?
7. What are the two ways to display the current values for all initialization parameters?
8. How can you tell whether a dynamic initialization parameter needs the DEFERRED option in order to be changed with an ALTER SYSTEM command?
9. If you need to prevent end users from logging into the database, what can you enable?
10. What two numbers do you need to know in order to kill a session?

In this chapter, you learn how to work with the Oracle instance. Since the instance is your “door” to the database, it is extremely important to understand its components, operation, and tuning. You have already learned the definition of the Oracle Instance in Chapter 1, “Oracle8i Architecture.” You also were introduced to the major components of an instance—the SGA and the background processes. In this chapter, we look at how to configure the instance, how to start it and open the database, and how to shut it down. We also discuss the parameter file and changing the values of parameters, as well as managing user sessions (well, killing them, anyway). At the end of the chapter, we look at the ALERT log and diagnostic trace files created by the instance. In the lab at the end of the chapter you will practice getting and setting parameter values, managing sessions, and starting up and shutting down Oracle.

Overview of Starting and Stopping an Oracle Instance

Perhaps the first thing a DBA needs to know is how to start and shut down the database. While the process itself is far from complicated if everything goes well, you need to understand the fine points of what goes on behind the scenes—for the exam as well as in real life.

Starting up a database includes these steps:

1. First, Oracle reads the parameter file (INIT.ORA) and builds the instance. This is called the NOMOUNT stage.
2. Once the instance is built, Oracle reads the control file. Because the control file contains the physical structure of the database, Oracle now knows where to find the data and log files. This stage is called MOUNT.
3. Once the database is mounted, Oracle actually opens the data and log files. This is the third and final stage of the startup. Not surprisingly, it is called OPEN.

When a database is shut down, the steps are reversed. First, the database is closed, then dismounted, and then the instance is shut down. Before the database is closed, the user sessions must end. Also, the database can be left in a consistent or inconsistent state, depending on whether Oracle performs a checkpoint before closing it. Here are the four ways to shut Oracle down:

NORMAL	Oracle waits for all users to log out and performs a checkpoint before closing the database. Once the database is closed, the instance is shut down.
TRANSACTIONAL	Oracle logs all users out as soon as their transactions are completed. A checkpoint is performed, and the database is closed. Then, the instance is shut down.

IMMEDIATE	Oracle does not wait for current transactions to end and immediately ends all user sessions, rolling back all transactions in progress. Then, a checkpoint is performed and the database is closed. Finally, the instance is shut down.
ABORT	Oracle does not really close the database as such—it simply shuts down the instance. As a result, all user connections are terminated, and no checkpoint is performed. Active transactions are not rolled back, the database is left in an inconsistent state and requires recovery on the next startup. The recovery is automatic, but may take a while because Oracle needs to roll forward all the changes since the last checkpoint and roll back all uncommitted transactions.

Parameter File (INIT.ORA)

Objective

Create the parameter file

The file you need in order to start the instance is the parameter file. Assuming that Oracle has been installed, this is the only file you really need. Oracle uses the parameter file in order to determine how much memory to allocate for the SGA and its parts, and which background processes to start. Because the instance is nothing more than SGA and the background processes, you can use the INIT.ORA file to completely control all aspects of the instance.

The parameters in the INIT.ORA can be specified in any order. The syntax is simple:

```
keyword = value
```

You can also specify multiple values for the same parameter, by putting the values in brackets and separating them with commas, like this:

```
keyword = (value1, value2, value3)
```

For example, you should use multiple control files, so the parameter should look like this:

```
control_files = (c:\certdb\disk1\control01.con, c:\certdb\disk2\control02.con)
```

Be careful not to specify the same parameter twice: with very few exceptions, Oracle will only use the last value. Unfortunately, humans read top to bottom, so you are not likely to notice that there is another line with the same parameter at the end of the file when you are trying to figure out where Oracle is getting a specific value from.

If you want to place a comment into the parameter file, just start it with a “#” sign. Everything after the “#” sign is treated as a comment and not processed.

Here is an example of a parameter file:

```
db_name = "orcl"
db_domain = computersapiens.com
instance_name = orcl
service_names = orcl.computersapiens.com
compatible = 8.1.0

db_files = 30

control_files = ("D:\Oracle\oradata\orcl\control01.ctl",
                 "E:\Oracle\oradata\orcl\control02.ctl")

db_block_size = 4096
db_file_multiblock_read_count = 8

db_block_buffers = 200
shared_pool_size = 4194304
large_pool_size = 614400
java_pool_size = 0
log_buffer = 32768

log_checkpoint_interval = 10000
log_checkpoint_timeout = 1800

processes = 50

parallel_max_servers = 5

#audit_trail = true # if you want auditing
#timed_statistics = true # if you want timed statistics
max_dump_file_size = 10240 # limit trace file size to 5M each

# Uncommenting the line below will cause automatic archiving if archiving has
# been enabled using ALTER DATABASE ARCHIVELOG.
# log_archive_start = true
# log_archive_dest_1 = "location=C:\Oracle\oradata\orcl\archive"
# log_archive_format = %%ORACLE_SID%%T%TS%S.ARC

rollback_segments = ( RBS0, RBS1, RBS2, RBS3, RBS4, RBS5)

background_dump_dest = D:\Oracle\admin\orcl\bdump
user_dump_dest = D:\Oracle\admin\orcl\udump

remote_login_passwordfile = exclusive

sort_area_size = 65536
sort_area_retained_size = 65536
```

Required INIT.ORA parameters

Technically, there is no such thing as a required INIT.ORA parameter because each has a default value, which is often operating system dependent. You can create and run a database with an empty parameter file if you like. However, some of the defaults are less than optimal for most environments. Therefore, Table 3-1 lists some of the parameters that are always specified.

Table 3-1
Required INIT.ORA Parameters

<i>Parameter Name</i>	<i>Parameter Description</i>	<i>Default Value</i>
DB_NAME (String)	Used to specify the name of the database instance the instance is opening. Usually the same as the instance SID, but does not have to be. During creation of the database, must match the name specified in the CREATE DATABASE statement.	During creation of the database, same as the instance SID. After creation of the database, must be present for the database to start.
CONTROL_FILES (String)	Used to specify the names and locations of the control files. You should have at least two control files on different disks in order to prevent loss of all copies of control files.	Operating system dependent. On NT, %ORACLE_HOME%\DATABASE\CTL%\ORACLE_SID%.ORA.
DB_BLOCK_SIZE (Integer, specified in bytes, must be a multiple of the operating system	Used to specify the size of the database blocks for constructing the SGA on startup. During creation of the database, this is the size of the blocks used to build the allocation unit size) database. During startup, database blocks are cached in the SGA, so if the DB_BLOCK_SIZE parameter value does not match the actual block size used in the database, you get an error. Because the database block size cannot be changed without re-creating the database, this parameter is never adjusted after the creation of the database. You should consider using a block size of 4,096 bytes for an OLTP database, 8,192 bytes for a hybrid, and 16,384 and higher, if possible, for a data warehouse. The range is operating system dependent.	2,048 bytes.



Creation of the database is discussed in Chapter 4; multiplexing control files is discussed in Chapter 6; and optimal values for `DB_BLOCK_SIZE` is discussed in Chapter 9.

Other key INIT.ORA parameters

In addition to the parameters in Table 3-1, there are others that are most often used by DBAs in order to configure the instance. Some are used in order to size the SGA structures, some to start and configure background processes, and some to point the instance at directories used for various purposes. The list of parameters is long (over 180 documented, plus hidden ones), but here are the most common ones.

- ♦ `DB_BLOCK_BUFFERS` (Integer, 4 to operating system dependent maximum).
 - A very important parameter that is used to size the buffer cache in the SGA. The parameter specifies the number of database blocks cached. The bigger the cache, the more data you can cache for faster future access. Remember that the total size of the buffer cache is $DB_BLOCK_SIZE \times DB_BLOCK_BUFFERS$, and the server has a limited amount of memory.
 - Default: 50,331,648 bytes ÷ `DB_BLOCK_SIZE` (this means that the default size of the buffer cache is 48MB).
- ♦ `LARGE_POOL_SIZE` (Integer, specified in bytes. Minimum: 600K. Maximum: Operating system dependent, but at least 2GB. Can be set to 0 if not required.)
 - The large pool is the memory allocated to sessions in a multithreaded server, as well as for parallel processing. For example, it may be used by RMAN during backups.
 - Default: 0 unless parallel processing is enabled by setting relevant parameters. Then, the default is calculated using a complex formula.
- ♦ `SHARED_POOL_SIZE` (Specified in bytes. Minimum: 300 bytes. Maximum: Operating system dependent.)
 - As you already know, the shared pool is an important area in the SGA that holds the library cache and the data dictionary cache. By using this parameter, you can change the amount of memory allocated to this structure. Oracle actually distributes the memory between the library cache, the data dictionary cache, and other parts of the shared pool, “favoring” the data dictionary cache. By using a larger shared pool, you enable more statements and procedures to remain cached, improving the efficiency of your server.
 - Default: Operating system dependent. Can be 8, 16, or 64MB.

- ♦ **SORT_AREA_SIZE** (Specified in bytes. Minimum: $1 \times \text{DB_BLOCK_SIZE}$. Maximum: Operating system dependent.)
 - When users build an index or perform a query that includes an ORDER BY clause, Oracle may need to sort the data. This requires a considerable amount of memory, depending on the amount of data to be sorted. The memory is allocated in the PGA, up to the maximum specified by this parameter. Be careful: In a worst-case scenario, each user session can get this much memory allocated simultaneously. After the sort is complete, the memory is released, except for the amount specified by the SORT_AREA_RETAINED_SIZE parameter.
 - Default: Typically, 65,536 (64KB)
- ♦ **LOG_BUFFER** (Specified in bytes)
 - You know that the log buffer is part of the SGA that is used to hold log entries before they are written out to the log file by the LGWR process. This parameter is used to specify the size of the log buffer. A bigger buffer may help improve performance of a system that processes large transactions (batch jobs) by accumulating more changes before actually writing them out to disk. Oracle recommends setting it to 64K or higher.
 - Default: Operating system dependent. Also depends on the number of CPUs on the system. Usually, 500K or $128K \times \text{CPU_COUNT}$, whichever is greater.
- ♦ **DB_FILES** (Integer. Minimum: The actual number of datafiles in the database. Maximum: Operating system dependent.)
 - Used to specify how many datafiles the instance can keep open at any given time. You should keep the value identical to the value of MAX-**DATAFILES** specified during creation of the database or the control file.



Control files are discussed in Chapter 6; creating the database is covered in Chapter 4.

- Default: Operating system dependent. On NT: 200.

- ♦ **LOG_CHECKPOINT_TIMEOUT** (Specified in seconds)
 - This is one of the parameters that specify the frequency of checkpoints. The checkpoint target will be this many seconds behind the current log position. It also means that no buffer remains dirty (unsynchronized) for more than LOG_CHECKPOINT_TIMEOUT seconds.
 - Default: Oracle8i: 900 seconds (15 minutes). Enterprise Edition: 1,800 seconds (30 minutes).
- ♦ **LOG_CHECKPOINT_INTERVAL** (Specified in operating system blocks)
 - Another parameter used to specify the checkpoint frequency. The checkpoint target is this many operating blocks behind the current log position.

**Caution**

Setting this parameter to 0 in Oracle versions prior to 8i results in constant checkpoints and negatively affects performance.

- Default: Operating system dependent. On NT: 0 (disabled).

♦ **FAST_START_IO_TARGET** (Integer. Minimum: 1,000. Maximum: Equal to `DB_BLOCK_BUFFERS`. 0 means disabled.)

- Used to dynamically control the checkpoint interval. This parameter specifies how many blocks you want to be processed during instance recovery in the unlikely event of a crash. The smaller the value, the shorter the recovery. Only available in the Enterprise Edition.

**Cross-Reference**

Checkpoints are discussed in detail in Chapter 7.

- Default: All the buffers in the cache (equal to `DB_BLOCK_BUFFERS`).

♦ **LOG_ARCHIVE_DEST** (Path)

- This is the parameter that sets the archiving destination for redo logs.
- Default: Officially, none. However, if this parameter is not set, logs are archived to `?\RDBMS`. (The “?” in the path means `ORACLE_HOME`, especially when used for scripts.)

♦ **LOG_ARCHIVE_FORMAT** (String)

- You can use this one to configure the names of the archived logs. By using placeholders such as `%S`, you can have Oracle include the sequence number in the filename and make it easier to find the right log. The file will be saved in the `LOG_ARCHIVE_DEST` directory.
- Default: Operating system dependent. On NT: `ARC%S.%T` (`%S` is sequence number, `%T` is thread number).

♦ **LOG_ARCHIVE_START** (Boolean)

- By setting this to `TRUE`, you can start the Archiver process (`ARCn`) that automatically archives redo logs before they are overwritten. This is done when the database is in archivelog mode only.

**Cross-Reference**

Archiving logs and the three parameters that deal with it are discussed in more detail in Chapter 7.

- Default: `FALSE`

♦ **ROLLBACK_SEGMENTS** (String. Contains names of private rollback segments listed in `DBA_ROLLBACK_SEGS` view).

- You can configure Oracle to automatically “acquire” some rollback segments on startup. If the rollback segments were created as `PRIVATE` (the default), they need to be listed in this parameter in order to be brought online automatically when the database opens. Be careful when setting this parameter: If there is even one segment listed that does not exist or cannot be enabled for any reason, the database won’t open.



Rollback segments are covered in Chapter 10.

- Default: None. This means that the only rollback segments brought online automatically are SYSTEM and some of the PUBLIC rollback segments.
- ♦ TRANSACTIONS_PER_ROLLBACK_SEGMENT (Integer)
 - When Oracle starts, it calculates how many rollback segments it requires by using the following formula: $\text{TRANSACTIONS} \div \text{TRANSACTIONS_PER_ROLLBACK_SEGMENT}$. The default setting for transactions parameter is based on the value for sessions or processes. By setting TRANSACTIONS_PER_ROLLBACK_SEGMENT, you can set the optimal number of PUBLIC rollback segments acquired by the instance at startup.
 - Default: 5
- ♦ PROCESSES (Integer)
 - This is the maximum number of processes (background and server processes) spawned by Oracle. It in effect limits the number of sessions for the instance. When configuring this parameter, keep in mind that default values for other parameters, such as sessions and transactions, are based on the value of processes. The default value is usually too low, so you should consider increasing it to a number that allows for all the background processes as well as all the server processes you need in order to support the planned number of users.
 - Default: Derived, but with all other parameters at default values, 30.
- ♦ BACKGROUND_DUMP_DEST (Path)
 - Used to specify the directory in which the ALERT log and the trace files created by the background processes are saved. We discuss these files in the “Managing the ALERT File and Trace Files” section of this chapter.
 - Default: Operating system dependent. On NT: ?\rdbms\log.
- ♦ USER_DUMP_DEST (Path)
 - Used to specify the destination for trace files created by the server processes.
 - Default: Operating system dependent. On NT: %ORACLE_HOME%\RDBMS\TRACE.
- ♦ MAX_DUMP_FILE_SIZE (Specified in bytes, or UNLIMITED)
 - Used to limit the maximum size of trace files (except the ALERT log).
 - Default: UNLIMITED
- ♦ SQL_TRACE (Boolean)

- If you set this to TRUE, you are enabling tracing for every statement by every user for the entire instance, so it is not recommended. Instead, you can enable diagnostic tracing for a specific session by using an ALTER SESSION command or the DBMS_SESSION package. You should not set this parameter in the parameter file on a production database unless instructed to do so by Oracle Technical Support.
 - Default: FALSE
- ♦ REMOTE_LOGIN_PASSWORDFILE (Possible values: NONE, EXCLUSIVE, SHARED)
- This parameter configures whether and how the instance will use the password file. If you set it to NONE, only the operating system authenticates the privileged users. If you set this parameter to SHARED, only two users, SYS and INTERNAL, are allowed in the password file, but multiple databases will be able to share the file. If you set it to EXCLUSIVE, you are able to grant SYSDBA and SYSOPER privileges to users other than SYS and INTERNAL. This is the usual value for the parameter.
 - Default: NONE
- ♦ OS_AUTHENT_PREFIX (String)
- When using operating system authentication, this is the string that gets prepended to the operating system username in order to generate the database username. If you want the database user to be the same as the operating system user, set OS_AUTHENT_PREFIX to "" (an empty string).
 - Default: Operating system dependent, but typically "OPS\$"
- ♦ RESOURCE_LIMIT (Boolean)
- By default, Oracle does not track resource usage by user sessions, such as the CPU time consumed or the amount of data read. If you are going to enforce resource limitations with profiles, this parameter needs to be set to TRUE.



User profiles are discussed in Chapter 16.

- Default: FALSE
- ♦ DB_FILE_MULTIBLOCK_READ_COUNT (Integer)
- When Oracle is scanning an entire table, it is more efficient to read a number of blocks at once rather than reading them one by one. This parameter configures how many blocks will be read at once during a full table scan.
 - Default: 8
- ♦ COMPATIBLE (Version, range 8.0.0 to current release)
- Used to specify the desired level of backward compatibility. If you set this parameter to a version older than the current one, some of the features may be restricted.
 - Default: 8.0.0

♦ IFILE (Path)

- This is basically a “pointer” to another parameter file. By putting this parameter into the beginning of your parameter file, you can use a common parameter file for multiple databases. Use parameters after IFILE to set parameters that are specific to this instance and override the parameters in the common parameter file.
- Default: None

Oracle documentation lists 183 parameters. In addition, there are “undocumented” parameters, which usually start with the “_” character. Although they were designed for Oracle’s internal use, they are sometimes used by experienced DBAs in order to tweak their servers. As a general rule of thumb, these parameters should not be used except as directed by Oracle Support. They are very powerful and are capable of drastically affecting your server’s reliability and data integrity. Oracle won’t support you if you destroy your database by testing these parameters.

Storing saved configurations using Instance Manager

As an alternative to using a parameter file, you can take advantage of the stored configurations in the Instance Manager component of Oracle Enterprise Manager. In earlier releases of OEM, the stored configurations were saved in the registry of the client computer running the console. As you can imagine, this created problems where the configuration of the instance depended primarily on which console it was started from. In the current release, these configurations are stored in the shared repository on the management server and are accessible to all administrators.

In order to use stored configurations, you need to create one first. To create a stored configuration that reflects the current values of all parameters, follow these steps:

STEP BY STEP: Creating a Stored Configuration

1. Start the Oracle Enterprise Manager and connect to the instance you are working with. Although you can create a configuration without connecting as SYSDBA, you won’t be able to test it because you need to be a privileged user in order to start the instance.
2. Expand the Instance node.
3. Right-click the Database node and click Edit. You will be presented with the instance management screen. In order to make it more useful in the future, you can select the Show All States checkbox at the top. From here, you can start the instance and shut it down. If the database is not open, click Open and then the Apply button at the bottom of the screen. Then, click the All Initialization Parameters button at the bottom.

Notice that if you click the Description button, another pane opens up at the bottom and shows a brief description of the highlighted parameter.

4. Click on the Save As button at the bottom, and type in the name this configuration should be saved as. You may also type in a brief description. Click OK on the Saved Successfully box, click OK to close the parameter list screen, and OK again to close the Instance Manager screen. It will ask whether you want to save the current configuration—because we already did, and we did not make any changes since, you can safely say No. Now, in the main screen of the Enterprise Manager, you can see the new configuration.

From now on, you can start the instance by using a stored configuration instead of a parameter file. In order to use the stored configurations, you have to use the Instance Manager to start the instance. If you are using commands, you still need to use a parameter file, so keep it up-to-date.

Starting Up and Shutting Down an Oracle Instance

One of the most important skills for the exam, and for real life, is being able to start the instance and open the database, as well as shut it down properly. Although it seems like a simple task, there are a number of things to consider.

First, we need to know about the states of the instance and the database—SHUTDOWN, NOMOUNT, MOUNT, and OPEN. We need to know which views are available in each of them and which operations can be performed. We also must be able to transition from one state to another.

Then, we need to be able to shut the database down using any one of the four methods—NORMAL, TRANSACTIONAL, IMMEDIATE, and ABORT—and to understand the implications of each.

Startup/shutdown stages

On its way “up” and “down,” Oracle goes through four stages.

NOMOUNT

In NOMOUNT, only the instance is running. This means that the SGA is allocated and the background processes are running, but the control file and the data and log files have not been read yet. As a matter of fact, the database does not even have to exist. This is the state during which you can issue the CREATE DATABASE or the CREATE CONTROLFILE statements. Other than for issuing these two statements, this state is not used very often. When the database is being started, this is the first state it goes through.

MOUNT

After NOMOUNT, the next step is to MOUNT the database. Because there is a CONTROL_FILES parameter in the INIT.ORA file, Oracle knows where to look for the control files of the database. If all the control files mentioned in the CONTROL_FILES parameter are present and valid, the database is MOUNTed. This means that the control file is read and processed. The control file, as you know, contains the structure of the database—the locations of the data and log files. It also contains the synchronization information for the database. Given this, Oracle goes and reads the headers of the data and log files to make sure that they are present and current, but does not open the files themselves. The result is a maintenance state, where you can relocate data and log files, restore backups and recover them by applying archived log files, and query a large number of V\$ views.

OPEN

The next state is the final one, OPEN. When Oracle opens the database, it reads the data and log files, performs instance recovery if necessary, and allows users to access data in the database.

SHUTDOWN

The SHUTDOWN state is when the instance is not running. There is very little interesting about this state, but we need to shut down the database from time to time. For example, we may want to take a cold backup of all files, or we need to restart the database after changing some values in the parameter file. On startup, the parameter file will be read again, and the new values will take effect.

The STARTUP command

Objective

Start up an instance and open the database

In order to start the instance and (optionally) open the database when the database is shut down, we need the STARTUP command. It is issued from SQL*Plus, SQL*Plus Worksheet, or the Server Manager prompt and requires SYSDBA or SYSOPER privileges. This command enables us to start the instance only, or mount the database, or open the database in one simple statement. There are a few options that you need to be aware of:

- ♦ STARTUP NOMOUNT starts the instance only.
- ♦ STARTUP MOUNT starts the instance and mounts the database—that is, reads the control file.
- ♦ STARTUP OPEN or simply STARTUP starts the instance first, then mounts and opens the database. This is the usual form of this command.

You may need to point the instance at a specific parameter file to use. For example, you decided that the default location of the parameter file is not where you want to

keep it. In this case, you need to use the PFILE parameter to specify the location of the parameter file to be used. Your command could look something like this:

```
SVRMGR> STARTUP PFILE=D:\ORADATA\ORCL\INITORCL.ORA
```

Some less common options are FORCE and RESTRICT.

- ♦ **STARTUP RESTRICT** starts the instance and opens the database, but does not allow users to connect unless they have the RESTRICTED SESSION privilege. Because most of the time only the DBAs have this privilege (it is granted to the DBA role by default), this effectively allows only the DBAs to connect to and work with the database. You can allow other users to access the database later by using the ALTER SYSTEM DISABLE RESTRICTED SESSION command. We will discuss the ALTER SYSTEM commands in the “Dynamically Changing Parameter Values” section of this chapter.
- ♦ **STARTUP FORCE** is not something you need to use often. This command aborts the instance first, then starts it and opens the database. You use it when the database is “hung” between states. In effect, it is equivalent to SHUTDOWN ABORT immediately followed by STARTUP.



Although in theory Oracle can only be in one of the four states, in practice it can end up between them. For example, if a database fails to mount, you may end up between NOMOUNT and MOUNT states. You will need to shut down the instance, correct the problem, and try again.

The options can be combined if it makes sense to do so. For example, you may issue the following command when beginning to create a database:

```
SVRMGR> STARTUP NOMOUNT PFILE=C:\CERTDB\INITCERT.ORA
```

Also, the options can be specified in a different order. For example, the following is a valid command:

```
SVRMGR> STARTUP PFILE=C:\CERTDB\INITCERT.ORA MOUNT
```

Overall, here is the syntax of the command:

```
STARTUP [NOMOUNT|MOUNT|OPEN] [RESTRICT] [FORCE]
[PFILE=parameter_file]
```

Be sure to know all of the options for the exam and for real-life situations.

Changing instance modes

From time to time, you will need to start the instance only, then manually mount and open the instance. Or, you can start in MOUNT, then recover the database and open it manually. For that, you need the ALTER DATABASE command.

ALTER DATABASE command

This command has many uses, from relocating log files to resizing datafiles. The options we are looking at here are used to change the state of the database.

If you are in NOMOUNT state, you can mount the database by using the following command:

```
SVRMGR> ALTER DATABASE MOUNT;
```

In MOUNT state, you can open the database by issuing the ALTER DATABASE OPEN command, or you can dismount the database and only leave the instance running by using the (you guessed it!) ALTER DATABASE DISMOUNT command.

In theory, there is an ALTER DATABASE CLOSE command that takes a database from OPEN down to MOUNT, but it is virtually useless. It is only successful when there are no sessions other than your own connected to the database, and it lacks the options that would enable you to automatically disconnect all sessions. In practice, you will virtually always use the SHUTDOWN command followed by the STARTUP MOUNT to achieve this result.

The ALTER DATABASE command can only take you up or down one step at a time. This means that you cannot use ALTER DATABASE OPEN when you are in NOMOUNT, for example—you will get an error.

Automating instance startup

You can have Oracle start as soon as you start the server. The details are, of course, operating system dependent, but on Windows NT, you need to modify the settings of the Oracle service responsible for the instance. You can do so in a number of different ways, including editing the Registry or using the ORADIM.EXE utility.

First, you need to configure the service itself to start automatically. In order to do this, you need to go to the Control Panel (in Windows 2000, you need to go into the Administrative Tools) and double-click Services. Find the OracleServiceSID service and click the Startup button on the right (in Windows 2000, right-click the service and select Properties). Now select the startup type. Since you want it to start automatically, select Automatic. From now on, whenever the server is started, the service will also be started.

Next, you need to configure the service to automatically start the instance when the service starts. In order to do that, you start the Registry Editor by clicking Start ⇨ Run and entering **REGEDT32** in the text box.



Perhaps a little warning: If you are not comfortable with editing the Registry, please don't. There are no safeguards when you are directly editing the Registry, and you can easily make your system unbootable by typing in the wrong thing. Another method is available and is introduced in a couple of paragraphs.

Once you start the Registry Editor, you see five windows. Click the one that says `HKEY_LOCAL_MACHINE` in the title bar. Double-click `SOFTWARE`, then scroll down to `ORACLE` and double-click that. Click `HOME0`. On the right, you will see a lot of entries. One of them should be `ORA_SID_AUTOSTART`. If you cannot see this entry, you may be looking at the wrong `ORACLE_HOME`. On the left, you should have `HOME1` and possibly even `HOME2`. Check those—your machine may have multiple `ORACLE_HOMES`, and you need to find the right one.

Once you have located the `ORA_SID_AUTOSTART` entry, check the value. If it says `TRUE`, the service automatically starts the instance on startup. If it says `FALSE`, the value needs to be changed. Double-click the `ORA_SID_AUTOSTART` entry and change the value to `TRUE`, then click `OK`.

In order to start the instance and open the database, the service needs to know where the parameter file is. Check the `ORA_SID_PFILE` entry—it should contain the correct path to the `INIT.ORA` file. If not, it needs to be changed, too.

Sounds complicated? Well, all of this can be done in one simple step with the `ORADIM` utility. The trouble is,—you cannot use it to configure the service to start automatically and not to start the instance. It is either all automatic or all manual.

To make the service start automatically, start the instance, and open the database, use the following command from the command prompt (not the `SQL*Plus` or `Server Manager` prompt):

```
C:\>oradim -EDIT -SID SID -STARTMODE auto -PFILE parameter_file
```

This configures the service to start when the server is booted up and to start the instance and open the database using the specified parameter file.



The syntax of the `ORADIM` utility is covered in more detail in Chapter 4.

Opening a database in read-only mode

A database can be opened in read-only mode. This is a new feature in Oracle8i. You may want to do this if the database is used for queries only and there is no need to allow `DML` or `DDL` on it. When the database is in read-only mode, the only writes allowed are to the temporary tablespaces, for sorting purposes.

There is no `READ ONLY` option for the `STARTUP` command, so in order to open a database in read-only mode, you need to follow these two steps:

STEP BY STEP: Making a Database Read-Only

1. Mount the database by using STARTUP MOUNT:

```
SVRMGR> STARTUP MOUNT PFILE=C:\CERTDB\INITCERT.ORA
```

2. Issue the ALTER DATABASE OPEN READ ONLY command:

```
SVRMGR> ALTER DATABASE OPEN READ ONLY;
```

That's it — you have a read-only database. If you try to issue a DML or DDL statement, you will receive an error.

You cannot alter a read-only database to read-write by using the ALTER DATABASE command. In order to do that, you need to shut it down first, then start and open it normally:

```
SVRMGR> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR> STARTUP PFILE=C:\CERTDB\INITCERT.ORA
ORACLE instance started.
Total System Global Area                4818188 bytes
Fixed Size                               70924 bytes
Variable Size                           3850240 bytes
Database Buffers                        819200 bytes
Redo Buffers                             77824 bytes
Database mounted.
Database opened.
```

For example, you may have a reporting database. Every weekend, you load data into it, but for the rest of the week it is used for queries only. You may want to shut it down on Friday and start it in read-write mode. Then, you perform the load. After the load is done, you shut it down, mount it, and open it READ ONLY.

The SHUTDOWN command

Objective

Close a database and shut down the instance

We talked about shutting a database down already — after all, how can you start it if you do not shut it down first? Now, it is time to discuss the command in a little more detail.

The SHUTDOWN command can be used with one of four options: NORMAL (default), IMMEDIATE, TRANSACTIONAL, or ABORT. These options control whether

Oracle closes the database nicely and what happens to the existing user sessions. Here is the overall syntax of the command:

```
SHUTDOWN [NORMAL|IMMEDIATE|TRANSACTIONAL|ABORT]
```

NORMAL mode

When you use the SHUTDOWN NORMAL command (or just SHUTDOWN), you are shutting down the database in normal mode. This is the “too nice to work” option. First, Oracle prevents any new logins. Then, it waits for all the users to disconnect. Considering that some users don’t like logging out, this may take a while. Also, the Intelligent Agent — the server-side component of the Oracle Enterprise Manager — never logs out. This means that the database simply becomes unavailable for new logons.

When and if all the users do log out, Oracle performs a checkpoint. This means that all of the dirty buffers get written out to the datafiles, and the database is in a consistent state. Once that is done, the database is closed and dismounted, and then the instance is shut down.

Overall, this is not a very useful option for most environments. Unfortunately, it is the default, so if you forget to use another option, here is what happens: You issue the SHUTDOWN command and just get a blinking cursor on a new line. No users can log in, and your session is busy waiting for the shutdown to be processed. Since one of the users is the Agent who will never log out (username DBSNMP), this will take a while. You can do one of three things:

- ♦ First, if the Agent is the only user who refuses to log out, you can stop the Oracle `ORACLE_HOME` Agent service or kill the operating system process. This is not very nice, and besides, you may have other users on the system. However, if the Agent is the only user who has not logged out, it may be your best option because it will enable Oracle to perform a checkpoint before closing the database.
- ♦ The second option is to stop the Oracle service itself (or kill the operating system process). This is also not very nice, especially considering that the checkpoint won’t get performed.
- ♦ The last option is to start another session to Oracle. You need to connect as a privileged user — for example, INTERNAL. Killing all of the user sessions is not an option — you won’t be able to query the `V$SESSION` view to find the necessary information. We will discuss killing sessions later in this chapter — for now, just trust me: It will not work. The only thing you can really do is issue the SHUTDOWN ABORT command and abort the instance — any other command will just result in the “Oracle not available” error. No checkpoint, not nice.

Well, these are the options. Basically, don’t forget to specify an option with the SHUTDOWN command.

IMMEDIATE mode

This is the usual way to close the database and shut down the instance. The most important thing is to perform a checkpoint and leave the database in a consistent state, and this option does that. What it does not do is wait for current sessions to end, or even for active transactions to finish. The active transactions are rolled back, the users are forcibly logged out, and a checkpoint is performed. Then the database is closed and dismounted, and the instance is shut down nicely.

The statement is simple:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

It does take Oracle some time to perform all of those tasks, so do not take “IMMEDIATE” literally. It may take a few minutes to roll back the transactions, log users out, and especially perform the checkpoint.

TRANSACTIONAL mode

This is a compromise for those of us who do not believe in NORMAL and feel that IMMEDIATE is unnecessarily rough on users. The TRANSACTIONAL mode does log users out, but not until they finish (COMMIT or ROLLBACK) their current transactions. Then, a checkpoint is performed and the database is closed and dismounted in a consistent state. Last step—Oracle shuts down the instance.

Here is the syntax:

```
SVRMGR> SHUTDOWN TRANSACTIONAL
```

There is a TIMEOUT option for the transactional shutdown, and you can see it in the older versions of Oracle Enterprise Manager. In theory, it is supposed to wait for users to complete their transactions for a specified time, then roll those transactions back. However, it never worked, and Oracle no longer mentions it in their documentation or has it in the Instance Manager. If you are working with an older version, just ignore it.

ABORT mode

Aborting the instance is usually not your first choice. The reason is that all other shutdown modes perform a checkpoint and leave the database in a consistent state. ABORT does not. As a matter of fact, it never closes the database—it just kills the operating system processes and releases the memory that used to be allocated to the SGA. You can see that by watching the feedback from Oracle when you issue the command. Compare these two readouts:

```
SVRMGR> shutdown immediate;  
Database closed.  
Database dismounted.  
ORACLE instance shut down.
```

```
SVRMGR> shutdown abort;
ORACLE instance shut down.
```

As you can see, the IMMEDIATE option closes and dismounts the database first, whereas ABORT does not. The checkpoint is not performed, committed data will need to be recovered from redo logs, and any uncommitted transactions will need to be rolled back on next startup. This is known as instance recovery, or crash recovery. It is automatic, but it will delay the next startup. Also, if you were shutting down the database in order to perform a backup, you should start it up and shut it down again, nicely this time.

The only time the database should be shut down using the ABORT option is when you have no other choice, or when its state really does not matter anyway (for example, you are about to restore the whole thing from backup tapes).

Table 3-2 gives a summary of the four SHUTDOWN options.

Table 3-2 Summary of SHUTDOWN Options			
	<i>Wait for Users to Log Out</i>	<i>Wait for Transactions to Complete</i>	<i>Perform Checkpoint</i>
NORMAL	Yes	Yes	Yes
TRANSACTIONAL	No	Yes	Yes
IMMEDIATE	No	No	Yes
ABORT	No	No	No

Dynamic performance views

Oracle provides us with a multitude of views that enable us to monitor the activity in the instance and the database, as well as see the memory allocations and database structure. There are two types of views the DBAs frequently use: the dynamic performance views and the data dictionary views. The data dictionary views are based on the base tables in the data dictionary and often start with USER_, ALL_, or DBA_ prefixes. They are covered in detail in Chapter 5. In this chapter, we will discuss the dynamic performance views.

Most of the dynamic performance views that apply to a single instance start with the V\$ prefix. There are 183 of them listed in the V\$FIXED_TABLE view, although there are a few not listed there. There is no need to memorize all of them, but some you should know quite intimately — both for the exam and for real-life situations.

The V\$ views, as they are frequently called (“dynamic performance views” is too long and formal), are getting their data in real time from the SGA and the control file of the database. Because of this, most of them are available when the database is only mounted, and some can be queried when only the instance is running (NOMOUNT state). They give you up-to-date information on processes running, memory allocated, and different kinds of activity — rollback and sorting are two of the examples.

Although you do not have to memorize all of the views, you do need to know how to find the one you need. For most of the common ones, you can guess the name with fairly high accuracy by using a simple rule: V\$*whatever you are looking for, singular*. For example, if you need information on processes running, you could try V\$PROCESS. For SGA allocations, V\$SGA works, and for parameter names and values, V\$PARAMETER is a reasonable (and correct) guess.

If guessing is not good enough, you can query the V\$FIXED_TABLE view. It lists all the V\$ views that are available in NOMOUNT or MOUNT. Some of the V\$ views can only be queried when the database is open, and they will not show up in V\$FIXED_TABLE. For example, if you need to find all views that have to do with data and are available when the database is not open, try this query:

```
SVRMGR> SELECT * FROM V$FIXED_TABLE WHERE name LIKE '%DATA%';
NAME                                OBJECT_ID  TYPE      TABLE_NUM
-----
GV$DATABASE                          4294951316 VIEW          65537
V$DATABASE                          4294951047 VIEW          65537
GV$DATAFILE                          4294951319 VIEW          65537
V$DATAFILE                          4294951050 VIEW          65537
GV$SQL_BIND_METADATA                 4294951355 VIEW          65537
V$SQL_BIND_METADATA                 4294951202 VIEW          65537
GV$SQL_BIND_DATA                    4294951356 VIEW          65537
V$SQL_BIND_DATA                    4294951203 VIEW          65537
GV$DATAFILE_COPY                    4294951361 VIEW          65537
V$DATAFILE_COPY                    4294951221 VIEW          65537
GV$BACKUP_DATAFILE                  4294951365 VIEW          65537
V$BACKUP_DATAFILE                  4294951225 VIEW          65537
GV$DATAFILE_HEADER                  4294951370 VIEW          65537
V$DATAFILE_HEADER                  4294951253 VIEW          65537
GV$PROXY_DATAFILE                   4294951558 VIEW          65537
V$PROXY_DATAFILE                   4294951559 VIEW          65537
16 rows selected.
```

The GV\$ views are global and are only meaningful in a Parallel Server environment. Of the eight V\$ views listed, you can see that V\$DATAFILE is probably the one you need.

The next step is to find out which columns you are interested in. Some of these views have quite a few, and queries should select only the ones you are interested in. So, before actually querying a view like this, use the DESCRIBE command (can be abbreviated to DESC), like this:

```
SVRMGR> DESC V$DATAFILE;
```

This gives you the list of columns available. Don't worry if you do not understand the meanings of all of them — very few people do. Just use the ones you understand and don't be afraid to look them up in the documentation.

As we already discussed, these views are based on the information in the control file or directly on the memory structures in the SGA. It makes sense that the former are only available when the control file is read — that is, in MOUNT. The latter can be queried in any state except SHUTDOWN.

Views available in NOMOUNT state

The views that are available in NOMOUNT are the ones that are based directly on the SGA or have to do with the background processes that are part of the instance. Some of these views are listed in Table 3-3.

Table 3-3
Dynamic Performance Views Available in NOMOUNT State

<i>View Name</i>	<i>View Description</i>
V\$SGA	Displays summary information on the SGA.
V\$INSTANCE	Shows status and name of the instance.
V\$PROCESS	Shows the list and details on the Oracle processes running as part of the instance.
V\$PARAMETER	Shows the names and current values of initialization parameters.
V\$SESSION	Shows details on sessions currently connected to the instance.
V\$ACCESS	Shows sessions accessing objects in the database. This includes fixed tables, so this view is available in NOMOUNT.
V\$FIXED_TABLE	Lists the dynamic performance views available even when the database is not open.
V\$BGPROCESS	Has a row for each possible background process, even ones not started.
V\$MYSTAT	Shows statistics on the current session. Very useful for finding the session ID for your own session.
V\$PWFILE_USERS	Shows a list of users in the password file and the privileges (SYSDBA or SYSOPER) granted to them.
V\$VERSION	Lists version numbers for core components of Oracle.
V\$OPTION	Lists TRUE or FALSE for each option available with the current version of Oracle. You can see which options are installed and which are not.

There are other views, but these are the more commonly used ones. As you keep going through the book, you will find more detailed information on many of them.

Views available in MOUNT state

The rest of the V\$ views get their information from the control file of the database. Because the control file is only read when the database is mounted, they are not available in NOMOUNT. It is fairly easy to figure out which views these are simply by thinking of the contents of the control file. It keeps the structural information (data and log filenames and locations), synchronization information (sequence numbers and SCNs for redo and archived logs), and backup information. Table 3-4 lists some of the views only available in MOUNT:

Table 3-4
Dynamic Performance Views Available in MOUNT State

<i>View Name</i>	<i>View Description</i>
V\$CONTROLFILE	Lists the name and status of the control files for the database.
V\$DATABASE	Lists information on the database: name, status, mode (archivelog or noarchivelog), checkpoint number, etc.
V\$DATAFILE	Lists the names and details of the datafiles listed in the control file.
V\$DATAFILE_HEADER	Lists information found in the headers of the datafiles. Useful in recovery scenarios.
V\$LOG	Lists the details on the log file groups in the database: size, number of members, current sequence, whether the group has been archived.
V\$LOGFILE	Lists the names of the actual physical log files and the groups they belong to.
V\$TABLESPACE	Lists the names and numbers of the tablespaces.
V\$LOG_HISTORY	Shows which change was recorded in which log sequence.
V\$ARCHIVED_LOG	Shows logs that have been archived by the instance and the details of each.
V\$CONTROLFILE_RECORD_SECTION	Shows how many records have been allocated in the control file for log files, datafiles, log history, etc. Also shows the size of each record.
V\$BACKUP_SET	This and other backup-related views are only useful if you use RMAN (Recovery Manager) for performing backups. Recovery Manager records successful backups in the control file of the database.



Oracle has another exam that deals with backup and recovery issues.

Being familiar with V\$ views is an important part of being an efficient (and certified!) DBA. Make sure that you spend some time running queries against these views to get comfortable with their structure and the type of information they provide.

Managing Oracle Instance Settings

Objective

Get and set parameter values

OK, now we can start and shut down the instance and change the state of the database. However, there are a few more aspects of managing the instance that we need to discuss.

First, you should realize that if you change a parameter value in the parameter file, you need to restart the database for the change to take effect. Most of the time this is something to avoid. Not only is the database unavailable while it is being restarted, but there is another reason to avoid unnecessary restarts. When you shut down and restart the database, the data that was cached in the SGA is no longer cached. For a while the performance is going to be quite poor, until the cache is repopulated with relevant data.

So, we try to change as many parameters as possible dynamically. In order to do that, we need to know which parameters we can do this with, how we can do it, and what their values are right now.

Also, we need to be able to manage sessions. Sometimes a session can become a source of problems, and it needs to be terminated. At other times, we need to prevent most users from logging in altogether. In this last part of the chapter, we discuss some session management techniques.

Displaying parameter values

In order to display the current values of parameters, it is not enough to look at the parameter file itself. First, not all parameters are always explicitly specified there (most are not). Second, some may have been changed for the current session or the entire instance after the database has been started up.

There are two ways to display the current values of parameters. The simplest is the `SHOW PARAMETER` command. It gives you the current value for all documented parameters that match the search string you specify.

For example, say you need to know the value of sorting-related parameters. Issue the following command:

```
SHOW PARAMETER SORT
```

The result will look something like this:

```
SVRMGR> SHOW PARAMETER SORT
NAME                                TYPE      VALUE
-----
nls_sort                             string
sort_area_retained_size              integer 0
sort_area_size                       integer 65536
sort_multiblock_read_count           integer 2
```

As you can see, Oracle returns the names and current values of all parameters that have the string “SORT” in the name. The string is not case-sensitive, so you can enter it in any case.

The other way to find out what a parameter is set at is to query the V\$PARAMETER view. This view has a lot more information than the SHOW PARAMETER command provides you with. For example, it tells you whether a given parameter can be modified dynamically, what the scope of this modification is (SYSTEM or SESSION), and whether you need to use any particular options to modify it.

In order to simply view the current values of sort-related parameters, issue the following query:

```
SVRMGR> SELECT name, value FROM V$PARAMETER
        2> WHERE name LIKE '%sort%';
NAME                                VALUE
-----
nls_sort
sort_area_size                      65536
sort_area_retained_size              0
sort_multiblock_read_count           2
4 rows selected.
```

First, please note that the parameter names are case-sensitive and are stored in lowercase. Other than that, we have pretty much the same information from this query that we had from the SHOW PARAMETER command. The big difference is in the columns we did not select—most of them have to do with dynamic changes to parameter values.

Dynamically changing parameter values

Some of the parameters can have their values changed “on the fly,” while the database is running.

You are already familiar with the concept of a session. A session is a user connection to a database and has its own properties. As a matter of fact, a lot of the INIT.ORA parameters are just defaults for sessions and can be changed for each session. For example, the SORT_AREA_SIZE parameter can be changed for the current session by using the ALTER SESSION command, like this:

```
SVRMGR> ALTER SESSION SET sort_area_size = 131072;
```

Some other parameters can be changed dynamically for the entire instance by using the ALTER SYSTEM command. For example, the LOG_CHECKPOINT_TIMEOUT parameter can be changed for the instance by using the following command:

```
SVRMGR> ALTER SYSTEM SET log_checkpoint_timeout = 900;
```

Now what if you want to change the maximum amount of memory allocated to every session in the instance? If you try to issue the following command, you will get an error:

```
SVRMGR> ALTER SYSTEM SET sort_area_size=131072;  
ALTER SYSTEM SET sort_area_size=131072  
*
```

```
ORA-02096: specified initialization parameter is not modifiable with this option
```

What Oracle is trying to tell you is that you cannot use ALTER SYSTEM to modify the value of this parameter for the current sessions — only for the future ones. In order to do that, you need to use the DEFERRED option:

```
SVRMGR> ALTER SYSTEM SET sort_area_size = 131072 DEFERRED;  
Statement processed.
```

This will not affect the current sessions, but all the future ones will have up to 128K of memory allocated for sorting, if necessary.

**Caution**

After dynamically modifying a parameter value, make sure that you change its value in the parameter file, as well. If you don't, the changes will be lost after you restart the instance.

In order to change a parameter dynamically, you need to have sufficient privileges. Obviously, we would not want users to be able to change the values of initialization parameters for the entire system. It may also not be a good idea to even let them change settings for their own session — who is going to get called when things stop working? So, Oracle has two privileges that can be used to control who can change the session and system settings. These privileges are ALTER SYSTEM and ALTER SESSION. The DBA has both of them, of course.

**Cross-Reference**

These and other privileges are covered in more detail in Chapter 18.

Determining if a parameter can be changed

Guessing whether any given parameter can be changed dynamically is difficult; memorizing each of them is boring. Trying to remember which ones can be modified using the ALTER SESSION command or the ALTER SYSTEM command, and whether you need to use the DEFERRED option, can get the best of us committed to a psychiatric institution. Thankfully, Oracle provides us with the V\$PARAMETER view, which tells us exactly how we can modify each of the parameters, whether it can be done at all, and whether it has already been done after the last startup. The most relevant columns of the view are listed in Table 3-5.

Table 3-5
The Columns of the V\$PARAMETER View

<i>Column Name</i>	<i>Description</i>
NAME	The name of the parameter, in lowercase.
DESCRIPTION	A brief description of what the parameter does. A sort of built-in help for those times when the documentation is not available.
TYPE	The datatype of the parameter: 1 means Boolean, 2 means string, 3 means integer. There is only one parameter type 4—that is the IFILE parameter (pointer to another parameter file).
VALUE	The current value for the parameter—for the current session.
ISDEFAULT	TRUE if the current value of the parameter was set by default. If you explicitly set the parameter, this column will show FALSE, even if the current value is the same as default.
ISSES_MODIFIABLE	TRUE if the parameter can be changed by using the ALTER SESSION command; FALSE if it cannot.
ISSYS_MODIFIABLE	IMMEDIATE if the parameter can be modified by using the ALTER SYSTEM command without the DEFERRED option; DEFERRED if it requires the DEFERRED option; FALSE if it cannot be modified with the ALTER SYSTEM command.
ISMODIFIED	If the parameter has not been modified from the value set in the parameter file or by default, this column will show FALSE. If an ALTER SESSION command was used to modify it, you will see MODIFIED. If the parameter has been modified with an ALTER SYSTEM command, this column will show SYS_MODIFIED.

So, in order to find out how a parameter can be modified, issue the following command:

```

SVRMGR> SELECT name, isses_modifiable, issys_modifiable
        2> FROM V$PARAMETER
        3> WHERE name = 'sort_area_size';
NAME                                     ISSES  ISSYS_MOD
-----
sort_area_size                           TRUE   DEFERRED
1 row selected.

```

This means that you can change the value of this parameter by using the ALTER SESSION command or the ALTER SYSTEM ... DEFERRED command.

Managing User Sessions

Objective

Manage sessions

Any database exists in order to serve users. However, there are times when the database needs to be “user-free.” For example, you need to restore it from backup and test that the restoration was successful, or you need to perform a large, consistent export of data and cannot have anyone modify the database during it.

At such times, you need to clear the database of users and prevent others from logging in. The first thing to do is to make sure that only a few users, primarily DBAs, can connect to the database.

Restricting who can connect

You can restrict access to the database by enabling RESTRICTED SESSION. This means that only those users who have the RESTRICTED SESSION privilege can log in. By default, it is only granted to the DBA role, so no end users should have it. Effectively, you are putting the database into a DBA-only mode.

You can do this in one of two ways. The first is to use the ALTER SYSTEM command, like this:

```

SVRMGR> ALTER SYSTEM ENABLE RESTRICTED SESSION;
Statement processed.

```

This places the system into RESTRICTED SESSION mode and prevents most users from logging in. However, it does not get rid of the sessions already in progress. Those need to be killed manually, and you will learn to do that later in this chapter.

You already saw the second method—you can use the STARTUP RESTRICT command to start the database and immediately enable RESTRICTED SESSION. So, to get rid of all current sessions and place the database into restricted mode, you could shut down IMMEDIATE or TRANSACTIONAL and then use STARTUP RESTRICT.

In order to allow users to start logging in again, issue the following command:

```
SVRMGR> ALTER SYSTEM DISABLE RESTRICTED SESSION;  
Statement processed.
```

In order to see whether the RESTRICTED SESSION is in effect, query the V\$INSTANCE view:

```
SVRMGR> SELECT logins FROM V$INSTANCE;  
LOGINS  
-----  
RESTRICTED  
1 row selected.
```

When the RESTRICTED SESSION has been disabled, the value in the LOGINS column changes to ALLOWED.

Terminating sessions

Once you enable RESTRICTED SESSION, you need to forcibly log out the users currently connected to the database. Or, there is a session that is blocking a rollback segment, and it needs to be terminated.



Rollback segments and troubleshooting of this and other problems is discussed in Chapter 10.

What you need to do is find the right session and then use the ALTER SYSTEM command to “kill” it. The command has the following syntax:

```
ALTER SYSTEM KILL SESSION 'SID,SERIAL#'
```

So, we need to find the SID and serial number for the session we need to get rid of.

The SID is the session ID that is unique for each session currently connected to the instance. However, if a user disconnects, the SID may be reused for a new session, so just the SID itself is not sufficient to uniquely identify the session to kill. The combination of the SID and the serial number, however, is unique.

We get the SID and serial number from the V\$SESSION view. The session that is causing problems can be identified by a number of methods, depending on the problems it is causing. In Chapter 10, you will see how to track down a session blocking a rollback segment by using the V\$ROLLSTAT, V\$TRANSACTION, and V\$SESSION views. For now, just to keep things simple, we will assume that we simply want to get rid of all nonadministrative users still logged in.

In order to identify those users, we only need to query the V\$SESSION view:

```

SVRMGR> SELECT username, sid, serial#, status FROM V$SESSION
        2> WHERE username IS NOT NULL;
USERNAME          SID          SERIAL#     STATUS
-----
SCOTT              8            5 INACTIVE
SYS                7            1 ACTIVE
2 rows selected.

```

Excluding NULL simply excludes the rows that belong to Oracle's own internal sessions.

You can see that there are only two “real” sessions: yourself and Scott. Scott's session is listed as `INACTIVE`, meaning that it is not actively executing a command at this time.

If there are many user sessions, (there usually are), you can find your own session SID by using the `V$MYSTAT` view like this:

```

SVRMGR> SELECT DISTINCT SID FROM V$MYSTAT;
SID
-----
      7
1 row selected.

```

So, now we have confirmed that the second session shown in the `V$SESSION` view is our own, because the SID is the same as the SID in the `V$MYSTAT` view. Now all that is left is to actually kill Scott's session:

```

SVRMGR> ALTER SYSTEM KILL SESSION '8,5';
Statement processed.

```

If you requery the `V$SESSION` view now, Scott's session may still be there, with “KILLED” in the `STATUS` column. In order for the session to actually disappear, the user needs to be notified that the session has been killed. If the session is active when it is killed, this notification happens right away. If the session is not active, the user will be notified the next time they try to execute anything:

```

SVRMGR> SELECT * FROM EMP;
SELECT * FROM EMP
*
ORA-00028: your session has been killed

```

Now the session will disappear from the `V$SESSION` view altogether.

Note that this does not prevent Scott from immediately logging in again. In order to do that, you should enable `RESTRICTED SESSION` first, and then start killing existing sessions. If logins are restricted, Scott won't be able to reconnect.

The ALTER SYSTEM command disconnects the session from the database, but it does not kill the operating system process associated with it. In order to kill both of these birds with one stone, use one the following commands:

```
SVRMGR> ALTER SYSTEM DISCONNECT SESSION '8,5' IMMEDIATE;  
Statement processed.
```

Or:

```
SVRMGR> ALTER SYSTEM DISCONNECT SESSION '8,5' POST_TRANSACTION;  
Statement processed.
```

The first one, with the IMMEDIATE option, was not available prior to Oracle 8.1.6. Neither is available prior to 8i. The meaning of these keywords is fairly obvious: the POST_TRANSACTION option allows the user to complete the current transaction, while the IMMEDIATE option kills the session immediately.

Once the logins have been restricted and all the user sessions have been killed, you have a “user-free” database. Hurry up and perform the maintenance, because time is money!

Managing the ALERT File and Trace Files

Objective

Monitor the ALERT file and the trace files

Oracle does not run silently. Even when everything is OK, it records a log of significant operations for the DBA. When problems happen, it attempts to record relevant information to help us troubleshoot the problem. The first place to look is in the *SIDALRT.LOG* file, the ALERT file.

The ALERT file, or ALERT log, as it is commonly known, is the main log of activity for the instance. It records startups, shutdowns, log switches, and parameters explicitly specified in the parameter file. It also contains a record of CREATE, ALTER, or DROP statements for DATABASE, TABLESPACE, and ROLLBACK SEGMENT, whether issued by the DBA or as part of internal functions (for example, during startup). Finally, it contains a log of major errors — for example, block corruption or ORA-00600 errors. The ORA-00600 is a generic internal error recorded when an Oracle process encounters an unexpected low-level condition. You can think of it as Oracle’s version of a GPF. (Remember Windows 3.1?)

Here is an example of an ALERT log — lines have been edited to show significant events, and comments are placed inline to explain the events.

```
Thu Feb 22 05:55:59 2001  
ORACLE V8.1.6.0.0 - Production vsnsta=0  
vsnsql=e vsnxtr=3
```

```
Windows 2000 Version 5.0 , CPU type 586
Starting up ORACLE RDBMS Version: 8.1.6.0.0.
System parameters with non-default values:
  processes                = 20
  shared_pool_size         = 4194304
  large_pool_size          = 614400
  java_pool_size           = 32768
...
PMON started with pid=2
DBW0 started with pid=3
LGWR started with pid=4
CKPT started with pid=5
SMON started with pid=6
RECO started with pid=7
Thu Feb 22 05:56:01 2001
alter database mount exclusive
Thu Feb 22 05:56:06 2001
Successful mount of redo thread 1, with mount id 950117110.
Thu Feb 22 05:56:06 2001
Database mounted in Exclusive Mode.
Completed: alter database mount exclusive
Thu Feb 22 05:56:06 2001
alter database open
Thu Feb 22 05:56:07 2001
Thread 1 opened at log sequence 788
  Current log# 2 seq# 788 mem# 0: D:\ORACLE\ORADATA\ORCL\REDO02.LOG
Successful open of redo thread 1.
Thu Feb 22 05:56:07 2001
SMON: enabling cache recovery
SMON: enabling tx recovery
Thu Feb 22 05:56:11 2001
Completed: alter database open
```

This is a startup. You can see which parameters were explicitly set, which processes started as part of the instance, which log was the current log, and a step-by-step of mounting, then opening the database. There is no mention of a crash recovery, so the database was in a consistent state after the previous shutdown. Here is what a crash recovery looks like:

```
Thu Feb 22 10:48:08 2001
alter database open
Beginning crash recovery of 1 threads
Thu Feb 22 10:48:09 2001
Thread recovery: start rolling forward thread 1
Recovery of Online Redo Log: Thread 1 Group 2 Seq 788 Reading mem 0
  Mem# 0 errs 0: D:\ORACLE\ORADATA\ORCL\REDO02.LOG
Thu Feb 22 10:48:10 2001
Thread recovery: finish rolling forward thread 1
Thread recovery: 18 data blocks read, 18 data blocks written, 29 redo blocks
read
Crash recovery completed successfully
```

This indicates that the previous shutdown was not NORMAL, IMMEDIATE, or TRANSACTIONAL—the server may have lost power, or a SHUTDOWN ABORT may have been issued, or a background process failed and the instance was automatically aborted. You need to investigate further.

Here is a rollback segment problem:

```
Thu Feb 22 16:16:49 2001
ORA-1628: max # extents 3 reached for rollback segment RBS01
Failure to extend rollback segment 8 because of 1628 condition
```

You will discuss how to fix it in Chapter 10—for now, you just need to recognize an error when you see one. It helps to search the file for the “ORA-” string to find errors in it.

Sometimes you have a major error reported by a background process. It might look something like this in the ALERT log:

```
Sun Apr 29 08:52:28 2001
Errors in file D:\Oracle\admin\orcl\bdump\orclLGWR.TRC:
ORA-00313: open failed for members of log group 1 of thread 1
ORA-00312: online log 1 thread 1: 'D:\ORACLE\ORADATA\ORCL\REDO01.LOG'
ORA-27041: unable to open file
OSD-04002: unable to open file
O/S-Error: (OS 2) The system cannot find the file specified.
```

```
Sun Apr 29 08:52:28 2001
Errors in file D:\Oracle\admin\orcl\bdump\orclLGWR.TRC:
ORA-00321: log 1 of thread 1, cannot update log file header
ORA-00312: online log 1 thread 1: 'D:\ORACLE\ORADATA\ORCL\REDO01.LOG'
```

```
Sun Apr 29 08:52:28 2001
Errors in file D:\Oracle\admin\orcl\bdump\orclLGWR.TRC:
ORA-00313: open failed for members of log group 1 of thread 1
```

```
Thread 1 advanced to log sequence 1054
Current log# 1 seq# 1054 mem# 1: D:\ORACLE\ORADATA\ORCL\REDO1C.RDO
```

As you can see, one of the members of a log group (Group 1) is missing. The other member is fine, because you can see that the switch did happen (“Thread 1 advanced...” message close to the end). The other important part of this snippet is the pointer to the trace file orclLGWR.TRC. Let’s take a look:

```
*** SESSION ID:(3.1) 2001-04-29 08:52:18.579
ORA-00313: open failed for members of log group 1 of thread 1
ORA-00312: online log 1 thread 1: 'D:\ORACLE\ORADATA\ORCL\REDO01.LOG'
ORA-27041: unable to open file
OSD-04002: unable to open file
O/S-Error: (OS 2) The system cannot find the file specified.
ORA-00321: log 1 of thread 1, cannot update log file header
ORA-00312: online log 1 thread 1: 'D:\ORACLE\ORADATA\ORCL\REDO01.LOG'
ORA-00313: open failed for members of log group 1 of thread 1
```

```
ORA-00313: open failed for members of log group 1 of thread 1
ORA-00312: online log 1 thread 1: 'D:\ORACLE\ORADATA\ORCL\RED001.LOG'
ORA-27041: unable to open file
OSD-04002: unable to open file
O/S-Error: (OS 2) The system cannot find the file specified.
ORA-00321: log 1 of thread 1, cannot update log file header
ORA-00312: online log 1 thread 1: 'D:\ORACLE\ORADATA\ORCL\RED001.LOG'
ORA-00313: open failed for members of log group 1 of thread 1
```

As you can see, the message is more detailed (although in this case, not that much more informative—the ALERT log gave us all the information we needed to figure out the cause). Sometimes, you need this extra information.

The ALERT log and the trace files created by background processes are located in the directory specified by the `BACKGROUND_DUMP_DEST` parameter in the parameter file. It is extremely important that the DBA spends some time every day checking the ALERT log looking for potential problems, even if the instance seems to be operating normally. In our example, one of the two members of a group was missing, so only one was left. In effect, that group was not multiplexed—and Oracle was operating normally. The error we saw was the only indication that something was very wrong and required immediate attention.

This brings us to a very important question—how often do we check the ALERT log? Well, how long would you want a missing log file to remain unnoticed? How about corruption in the datafiles? Oracle recommends that the ALERT log be checked daily, but it should be more frequent than that. Also, don't forget to archive and clear the ALERT log from time to time—it can grow indefinitely.

A server process can create another type of a trace file. These trace files include the ones created because the `SQL_TRACE` parameter has been enabled and files created by the `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command. These traces are saved in the directory specified by the `USER_DUMP_DEST` parameter.



Creating a trace file by using the `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command is discussed in Chapter 6.

Key Point Summary

- ♦ In order to start, an Oracle Instance needs the parameter file (`INIT.ORA`). This file contains settings that control space allocation in the SGA, background process configuration, and locations of the control files. There are almost two hundred parameters, but each has a default value that is usable for most databases.
- ♦ The Oracle Instance and database have four possible states: `SHUTDOWN`, `NOMOUNT`, `MOUNT`, and `OPEN`. You should be very familiar with these states and which files are read to put the database into each one.

- ♦ The `STARTUP` command is used to start a database that is currently shut down. You can use this command in order to put the database into `NOMOUNT`, `MOUNT`, or `OPEN` (default) state. If the database is not shut down, you can use the `ALTER DATABASE` command to move between states one step at a time.
- ♦ In order to shut the database down, you use the `SHUTDOWN` command. There are four modes of `SHUTDOWN`: `NORMAL` (the default), `TRANSACTIONAL`, `IMMEDIATE`, and `ABORT`. You should be very comfortable with each one and what it does and does not do. The main difference is that `ABORT` does not perform a checkpoint while others do.
- ♦ In order to get information on the current state of the instance and database, you can use dynamic performance views (`V$ Views`). The `V$` views that gather information from memory are available in `NOMOUNT` state; the ones based on the control file are only available in `MOUNT` or `OPEN`.
- ♦ Some of the parameters cannot be modified while the instance is running; others can — with an `ALTER SESSION` or `ALTER SYSTEM` command. In order to find out which parameters can be modified and the scope of possible modifications (`SESSION` or `SYSTEM`), use the `V$PARAMETER` view. The same view can be used in order to see the current values of the parameters, or you can use the simplified `SHOW PARAMETER` command.
- ♦ Sometimes you need to prevent end users from logging into the database. In order to do that, you use the `ALTER SYSTEM ENABLE RESTRICTED SESSION` command. To disconnect existing sessions, use the `ALTER SYSTEM KILL SESSION 'SID,Serial#'` command.
- ♦ In order to monitor the “health” of your Oracle instance, you need to check the `ALERT` file frequently. The `ALERT` file contains information on major events, such as startups, shutdowns, and major errors. Some additional diagnostic information may be present in the trace files created by background processes. Both the `ALERT` log and the background traces are recorded in the directory specified by the `BACKGROUND_DUMP_DEST` parameter.



STUDY GUIDE

In this part of the chapter, you will get a chance to practice the skills covered in the previous pages. First, you will try your hand at some exam-style questions. Then, you will look at two scenarios that require you to choose the appropriate course of action. After that, you will get a few lab exercises that enable you to get some hands-on practice with managing an instance. Finally, you will see the answers to the Pre-Test, the Quick Assessment, and the Scenario sections, and detailed steps required to complete the lab exercises.

Assessment Questions

1. Which of the following database states does not require that the control file be available?
 - A. NOMOUNT
 - B. MOUNT
 - C. OPEN
 - D. STARTUP RESTRICT
2. Which of the following dynamic performance views is available when the database is in the NOMOUNT state?
 - A. V\$DATABASE
 - B. V\$TABLESPACE
 - C. V\$SGA
 - D. V\$DATAFILE
3. Which two of the following must you specify when terminating a session? (Choose two.)
 - A. Username
 - B. Session ID
 - C. Session serial number
 - D. Instance name
 - E. Session address

4. How can you make a database read-only?
 - A. ALTER DATABASE READ ONLY
 - B. STARTUP RESTRICT
 - C. ALTER DATABASE OPEN READ ONLY
 - D. STARTUP READ ONLY

5. Which of the following shutdown modes performs a checkpoint before closing the database but does not wait for users to complete their transactions?
 - A. NORMAL
 - B. IMMEDIATE
 - C. TRANSACTIONAL
 - D. ABORT

6. Which of the following can be used in order to find out whether a parameter can be changed dynamically?
 - A. The SHOW PARAMETER command
 - B. The V\$INSTANCE view
 - C. The V\$SESSION view
 - D. The V\$PARAMETER view

7. What does the ALTER SYSTEM ENABLE RESTRICTED SESSION statement do?
 - A. It prevents new connections from users who do not have the RESTRICTED SESSION privilege and terminates existing sessions of such users.
 - B. It prevents new connections from users who do not have the RESTRICTED SESSION privilege but allows all existing sessions to continue.
 - C. It prevents new connections from users who do not have the RESTRICTED SESSION privilege but allows all existing sessions to continue in read-only mode.
 - D. It shuts down the database and restarts it with the RESTRICT option.

8. Which initialization parameter configures the size of the buffer cache in the SGA?
 - A. DB_BLOCK_BUFFERS
 - B. LOG_BLOCK_BUFFERS
 - C. DB_BUFFER_SIZE
 - D. LOG_BUFFER

9. Where in the parameter file should the IFILE parameter be placed?
 - A. In the end
 - B. In the middle
 - C. In the beginning
 - D. Does not matter
10. After you dynamically modify a parameter using the ALTER SYSTEM ... DEFERRED command, which sessions will get the new value?
 - A. Current sessions only
 - B. New sessions only
 - C. Both current and new sessions
 - D. DEFERRED is not a valid option

Scenarios

1. As the new DBA for Humungous Online Company Inc., you are now responsible for the maintenance of the 24/7 database supporting their Web site. In the past, all changes to the parameter values have been performed by changing the parameter file, shutting down, and then restarting the database. The management has asked you to provide them with a plan to improve the uptime of the database.
 - A. Does the database need to be shut down and restarted every time you need to change the value of a parameter? If not, how can it be done?
 - B. How can you find out if a given parameter can be changed dynamically?
 - C. One of the requests of the management is that when you do need to close the database and shut down the instance, you do so without losing the current transactions. How can this be achieved?
 - D. Your assistant needs to be able to dynamically change the parameter values for the entire system. What privilege does your assistant need?
2. You are a DBA at Honest Marketing Research Ltd. Their database is supposed to be used for queries only, with weekly loads on weekends. Unfortunately, sometimes users make mistakes and change the data. Also, some users are workaholics and log into the system while you are performing the load on Saturdays. You need to prevent all changes to data other than the loads, and you need to prevent users from logging in during the load itself. How can you achieve these goals?

Lab Exercises

You will create your practice database in the labs following the next chapter. For now, you can use the sample ORCL database that was created as part of a typical Oracle installation. If it was not created, you can easily add it by running the Oracle Database Configuration Assistant. Accept all default values — you will only need this database for these labs. Alternatively, you can wait to perform these labs until after you complete the labs in Chapters 4 and 5.

Unless specifically stated otherwise, the lab assumes that you are logged into the database as a user with SYSDBA privileges.

Lab 3-1 Modifying the Parameter File

1. Locate the parameter file for the ORCL instance on your computer. Make a copy of it, just to be on the safe side.
2. Using a text editor, open the file and make the following changes:

- a. Change the buffer cache size to 2,427,600 bytes



Remember that this parameter is set in database blocks, not bytes! You need to calculate the number of blocks that need to be cached in order to cache 2,427,600 bytes.

- b. Change the log buffer size to 131,072 bytes.

- c. Make sure that each session is allocated up to 131,072 bytes for sorting in memory.

3. Make the changes take effect.
4. Using Server Manager line mode (svrmgrl), confirm that the new values are now in effect.

If the instance did not start properly, confirm that the changes you made are correct and the values are valid. You may need to issue SHUTDOWN ABORT and/or restart the service (NT and Windows 2000 only) in order to be able to start the instance again after correcting any mistakes.

Lab 3-2 Dynamically Modifying Initialization Parameters

1. Of the parameters you modified in the previous lab, which can be modified again without restarting the database? Use Server Manager line mode (svrmgrl) to find out.
2. Using Server Manager line mode, modify that parameter for the current session to half of its current value.
3. Confirm that the change has taken effect.
4. Using Server Manager line mode, dynamically change the value of this parameter system-wide to 8,096 bytes. What option do you need to use?

5. Check whether this parameter got changed for the current session. Did it change? Why or why not?
6. Start a new Server Manager session to the server and log in as user SYSTEM, with password MANAGER. Check the value of the parameter you changed in step 4. Does this session have the current value? Log out of this session.
7. Restart the database. What is the value of the parameter after restarting?
8. Restore the old values for the three parameters you changed in step 2 of Lab 3-1 and restart the database.

Lab 3-3 Managing Sessions

1. From the session still running from the previous lab, make the database inaccessible for end users.
2. Using another session, log in as user SYSTEM, with password MANAGER. Could you log in? Why or why not? Leave the session running.
3. How can you confirm that end users are restricted from logging in?
4. From the first session, kill the second one (established in step 2).
5. What is the status of the killed session?
6. From the second (killed) session, try to query the V\$SESSION view. What is the result?
7. From the first session, check that the second session no longer exists.

Answers to Chapter Questions

Chapter Pre-Test

1. The syntax is *parameter_name=parameter_value* if there is only one value; *parameter_name=(parameter_value_1,parameter_value_2,...parameter_value_n)* if there is more than one value. Comments are marked with the “#” sign.
2. The stored configurations are stored in the Repository on the Management Server. In the previous releases of Oracle, they were stored in the Registry.
3. The control file is read when the database is MOUNTed. In NOMOUNT state, only the parameter file is read; when the database is OPEN, the datafiles and log files are opened.
4. In order to change the database from NOMOUNT state to MOUNT state, you need to use the ALTER DATABASE MOUNT command.
5. The SHUTDOWN option that waits for users to complete their transactions and then logs them off is TRANSACTIONAL. NORMAL does not log users off automatically; IMMEDIATE does not let users complete their transactions before logging them off; ABORT just kills the instance.

6. In order to display the names of the dynamic performance views available in MOUNT state, you should query the V\$FIXED_TABLE view.
7. The two ways to display the current values of initialization parameters are the SHOW PARAMETER command and the V\$PARAMETER view. The SHOW PARAMETER command does not show whether the parameter can be or has been modified dynamically; the V\$PARAMETER view does.
8. You can tell whether you need to use the DEFERRED option with the ALTER SYSTEM command in order to dynamically modify a parameter by looking at the ISSYS_MODIFIABLE column of the V\$PARAMETER view. If the value is FALSE, you cannot use the ALTER SYSTEM command to modify this parameter. If the value is IMMEDIATE, the DEFERRED option is not necessary. If the value is DEFERRED, you need to use the DEFERRED option.
9. You need to enable RESTRICTED SESSION in order to prevent end users from logging into the database. You can also, of course, simply disconnect the network cable from the server (not the right answer on the exam). When you enable RESTRICTED SESSION, only the users who have the RESTRICTED SESSION privilege can connect to the database, and this privilege is only granted to the DBA role by default.
10. In order to kill a session by using either the ALTER SYSTEM KILL SESSION or the ALTER SYSTEM DISCONNECT SESSION commands, you need the session ID (the SID column in the V\$SESSION view) and the serial number (the SERIAL# column in the V\$SESSION view).

Assessment Questions

1. A. The NOMOUNT state means that the instance has been started, but it has not attempted to open or mount the database. For the instance to start, the only file that is necessary is the parameter file. The control file is read when the database goes from the NOMOUNT state to MOUNT, so answer B is incorrect. The OPEN state (answer C) requires the database to be MOUNTed first, so the control file must be present. STARTUP RESTRICT (answer D) is a command, not a state. It opens the database and immediately enables RESTRICTED SESSION. For more information, see “Startup/shutdown stages” in this chapter.
2. C. When the database is in NOMOUNT state, the only views you can query are the ones that get their information from memory or have to do with the background processes. Of the four views listed, the only one that fits this description is the V\$SGA view, which shows the amount of memory allocated to the SGA. The other three views get their information from the control file. The control file is not read until the database goes into MOUNT state, so these views are not available in NOMOUNT. For more information, see “Dynamic performance views” in this chapter.

3. **B and C.** The syntax for the command necessary to kill a session is `ALTER SYSTEM KILL SESSION ('SID,Serial#')`. The `ALTER SYSTEM DISCONNECT SESSION` command requires the same parameters. The username and session address may be necessary for properly identifying the session to be killed, but they are not needed to kill it. For more information, see “Terminating sessions” in this chapter.
4. **C.** In order to make a database read-only, you need to `MOUNT` it first. Then, you can issue the `ALTER DATABASE OPEN READ ONLY` command. If the database is already open, it needs to be shut down first, so answer A is incorrect. Answer B is incorrect because `STARTUP RESTRICT` starts the instance and opens the database in `RESTRICTED SESSION`, not `READ ONLY` mode. Answer D is incorrect because you cannot start the instance and open the database in read-only mode in one step. For more information, see “Opening a database in read-only mode” in this chapter.
5. **B.** The `IMMEDIATE` shutdown logs users off automatically, whether or not they have completed their transactions. Any transactions that were in progress are rolled back. The `NORMAL` shutdown (answer A) waits for users to not only complete their transactions, but to voluntarily log out. The `TRANSACTIONAL` shutdown (answer C) allows users to complete their current transactions before logging them off. The `SHUTDOWN ABORT` command (answer D) does not wait for users to complete their transactions, but neither does it perform a checkpoint (all other options do). For more information, see “The `SHUTDOWN` command” in this chapter.
6. **D.** The `ISSES_MODIFIABLE` and `ISSYS_MODIFIABLE` columns of the `V$PARAMETER` view tell you whether a parameter can be dynamically modified by using the `ALTER SESSION` or `ALTER SYSTEM` command, or both. The `SHOW PARAMETER` command (answer A) only shows the name and value for each parameter, not whether it can be modified dynamically. The `V$SESSION` view (answer C) shows details on all current user sessions and has nothing to do with initialization parameters; the `V$INSTANCE` view (Answer B) does not show any information on initialization parameters, either.
7. **B.** When you execute the `ALTER SYSTEM ENABLE RESTRICTED SESSION` command, users who do not have the `RESTRICTED SESSION` privilege are unable to connect to the database. However, existing sessions are not affected and may need to be killed. This makes answer A incorrect. Answer C is incorrect because there is no such thing as a read-only session. Shutting down the database and restarting it with the `RESTRICT` option (answer D) is an alternative to enabling `RESTRICTED SESSION` and killing all current user sessions, but it does not happen automatically. For more information, see “Restricting who can connect” in this chapter.
8. **A.** The `DB_BLOCK_BUFFERS` parameter configures how many blocks will be cached in the buffer cache. Multiplied by the `DB_BLOCK_SIZE`, it gives you the amount of memory allocated to the buffer cache in the SGA. The parameters in answers B and C do not exist. The `LOG_BUFFER` parameter (answer D) controls the amount of memory allocated to caching log entries. For more information, see “Parameter File (INIT.ORA)” in this chapter.

9. **C.** The IFILE parameter should be placed in the beginning of the parameter file. This way, you can override the parameters in the included generic file with parameters specifically set later in the parameter file for the instance. Otherwise, the parameters set in the file pointed at by the IFILE parameter can override the parameters you set for this instance. For more information, see “Parameter File (INIT.ORA)” in this chapter.
10. **B.** If you need to use the DEFERRED option in order to modify a parameter with the ALTER SYSTEM command, it means that Oracle cannot modify the parameter for the current sessions, but the new sessions will get the new setting. For more information, see “Dynamically changing parameter values” in this chapter.

Scenarios

1. The previous DBA for Humungous Online Company Inc. was obviously unaware of the dynamic parameters in Oracle. By shutting down the database every time a change to a parameter value was needed, he or she made the database unavailable for a few minutes and invalidated the cache, reducing the performance of the database. This is not the way to run a 24/7 database.
 - A. The database does not need to be shut down every time a parameter needs to be changed—you can use the ALTER SYSTEM command for a large number of them. The ALTER SESSION command is not very useful here—it will only change the value for your own session, not for all other users. In order to make the change persist even after you shut down and restart the database eventually, you should update the parameter file immediately after making a change with the ALTER SYSTEM command.
 - B. In order to find out whether a parameter can be changed dynamically, you can query the ISSYS_MODIFIABLE column in the V\$PARAMETER view.
 - C. Assuming that the application is written properly (it does not start a transaction and leave it open while waiting for user input), SHUTDOWN TRANSACTIONAL is a good solution since it will allow all current transactions to complete before closing the database.
 - D. Your assistant needs the ALTER SYSTEM privilege. Although granting the DBA role allows your assistant to change the parameters, too, it is not the best solution since it allows him or her to perform any action in the database (too much of a good thing).
2. In order to prevent changes to data, you can run the database in read-only mode for most of the week. In order to perform the load, you will need to make it read-write, but you can use STARTUP RESTRICT to enable RESTRICTED SESSION as soon as the database is open and prevent end-user connections. Once the load is done, shut down the database, mount it, and then use the ALTER DATABASE OPEN READ ONLY command to open it for queries only.

Lab Exercises

Lab 3-1

1. a. On Windows NT or Windows 2000, use Start ⇨ Find or Search to find the `initORCL.ora` file.

On UNIX-based systems, use the `find` command to find the same file.

1. b. On Windows-based systems, use Windows Explorer to copy the file to `initORCL.bak`.

On UNIX-based systems, use the following command:

```
cp initORCL.ora initORCL.bak
```

2. Using Notepad (Windows) or `vi` or your favorite text editor (UNIX), make the following changes:

- a. Locate the `DB_BLOCK_SIZE` parameter. Calculate how many blocks need to be cached in order to have 2,427,600 bytes in the buffer cache. The formula is

$$2,427,600 \div \text{DB_BLOCK_SIZE}$$

For a database with 2KB blocks, the result is 1,200. For 4KB blocks, you need 600. For 8KB blocks, it works out to 300 blocks. Assuming 4KB blocks, the parameter should look like this:

```
db_block_buffers=600
```

- b. Edit the `LOG_BUFFER` parameter:

```
log_buffer=131072
```

- c. Edit the `SORT_AREA_SIZE` parameter:

```
sort_area_size=131072
```

3. To make the changes take effect, restart the database:

```
SVRMGR> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE shut down.
SVRMGR> STARTUP PFILE=location_of_parameter_file
```

4. To check that the changes took effect, you can use the `SHOW PARAMETER` command or query the `V$PARAMETER` view:

```
SVRMGR> SHOW PARAMETER sort_area_size
NAME                                TYPE      VALUE
-----
sort_area_size                       integer  131072
SVRMGR> SHOW PARAMETER log_buffer
NAME                                TYPE      VALUE
```

```

-----
log_buffer                               integer 131072
SVRMGR> SHOW PARAMETER db_block_buffers
NAME                                     TYPE      VALUE
-----
db_block_buffers                         integer 600

```

Or:

```

SVRMGR> SELECT name, value FROM V$PARAMETER WHERE name IN
2> ('sort_area_size', 'log_buffer', 'db_block_buffers');
NAME                                     VALUE
-----
db_block_buffers                         600
log_buffer                               131072
sort_area_size                           131072
3 rows selected.

```

Lab 3-2

1. You can use the following query:

```

SVRMGR> SELECT name, issys_modifiable, isses_modifiable
2> FROM V$PARAMETER WHERE name in
3> ('sort_area_size', 'log_buffer', 'db_block_buffers');
NAME                                     ISSYS_MOD ISSES
-----
db_block_buffers                         FALSE     FALSE
log_buffer                               FALSE     FALSE
sort_area_size                           DEFERRED  TRUE
3 rows selected.

```

The only parameter that can be modified dynamically is `SORT_AREA_SIZE`. It can be modified for the current session by using the `ALTER SESSION` command or system-wide by using the `ALTER SYSTEM ... DEFERRED` command.

2. Use the following statement:

```

SVRMGR> ALTER SESSION SET sort_area_size=65536;
Statement processed.

```

3. You can use the following command:

```

SVRMGR> SHOW PARAMETER sort_area_size;
NAME                                     TYPE      VALUE
-----
sort_area_size                           integer 65536

```

The result shows that the setting has been changed for the current session.

4. Use the following command:

```

SVRMGR> ALTER SYSTEM SET sort_area_size=8096 DEFERRED;
Statement processed.

```

You need to use the DEFERRED option (see the query in step 1).

5. Use the SHOW PARAMETER command:

```
SVRMGR> SHOW PARAMETER sort_area_size;
NAME                                TYPE      VALUE
-----                                -
sort_area_size                       integer 65536
```

The value did not change, because the ALTER SYSTEM ... DEFERRED command only affects new sessions, not the current ones.

6. Using another session, connect as SYSTEM and use the SHOW PARAMETER command to check the value of SORT_AREA_SIZE for this session:

```
SVRMGR> CONNECT SYSTEM/MANAGER
Connected.
SVRMGR> SHOW PARAMETER sort_area_size
NAME                                TYPE      VALUE
-----                                -
sort_area_size                       integer 8096
SVRMGR> EXIT
Server Manager complete.
```

You can see that for this new session the value is the new value.

7. After restarting the database, the value returns to the one specified in the parameter file and all dynamic changes you made are lost, because the parameter file was not updated.
8. This is where that backup copy of the parameter file comes in handy.

Lab 3-3

1. You need to enable RESTRICTED SESSION:

```
SVRMGR> ALTER SYSTEM ENABLE RESTRICTED SESSION;
Statement processed.
```

2. The connection succeeds because SYSTEM has the DBA role, which has been granted the RESTRICTED SESSION privilege:

```
SVRMGR> CONNECT SYSTEM/MANAGER
Connected.
```

3. You can query the LOGINS column of the V\$INSTANCE view:

```
SVRMGR> SELECT logins FROM V$INSTANCE;
LOGINS
-----
RESTRICTED
1 row selected.
```

You can see that RESTRICTED SESSION is enabled.

4. a. First, you need to find that other session. Query the V\$SESSION view:

```
SVRMGR> SELECT username, sid, serial# FROM V$SESSION
        2> WHERE USERNAME IS NOT NULL;
USERNAME          SID          SERIAL#
-----
SYS                7             1
SYSTEM            8            40
2 rows selected.
```

4. b. Now you can issue the ALTER SYSTEM command:

```
SVRMGR> ALTER SYSTEM KILL SESSION '8,40';
Statement processed.
```

5. To find out the status of the session you killed, query the V\$SESSION view:

```
SVRMGR> SELECT username, sid, serial#, status FROM V$SESSION
        2> WHERE USERNAME IS NOT NULL;
USERNAME          SID          SERIAL#    STATUS
-----
SYS                7             1 ACTIVE
SYSTEM            8            40 KILLED
2 rows selected.
```

The status is “KILLED.” The user has not been notified yet.

6. You can use any query, actually. The result will be the same:

```
SVRMGR> SELECT * FROM V$SESSION;
SELECT * FROM V$SESSION
*
ORA-00028: your session has been killed
```

7. Requery the V\$SESSION view:

```
SVRMGR> SELECT username, sid, serial#, status FROM V$SESSION
        2> WHERE USERNAME IS NOT NULL;
USERNAME          SID          SERIAL#    STATUS
-----
SYS                7             1 ACTIVE
1 row selected.
```

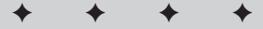
Creating a Database

EXAM OBJECTIVES

- ◆ Creating a Database
 - Prepare the operating system
 - Prepare the parameter file
 - Create the database

4

CHAPTER



CHAPTER PRE-TEST

1. What file must be configured before you can create a database?
2. What are the required elements of the Oracle CREATE DATABASE statement?
3. What privilege is required to create a database?
4. What command must be run on a Windows NT/2000 environment as part of the database creation process?
5. What mode must the Oracle instance corresponding to the database be in for the CREATE DATABASE command to execute successfully?
6. What is Optimal Flexible Architecture?
7. How many redo log files must be created when you create a database?
8. How do you create a control file?
9. What are two ways that you can create a database in Oracle?
10. What database block size will Oracle's datafiles have when the database is created? How do you change the database block size?

In this chapter, you will learn how to create an Oracle database using both the Oracle Database Configuration Assistant and the CREATE DATABASE command. First, you will learn what must be performed at the operating system to ensure that database creation takes place successfully. You will then find out about the importance of placing different files that are created in appropriate locations on the hard disks of the computer that will hold the database. This chapter also discusses the permissions that users need to create a database. Finally, the chapter discusses and presents solutions to some common problems that arise during database creation. In the lab at the end of the chapter, you will create the CERTDB database that you will use throughout the rest of the book.

Overview of Creating and Managing a Database

The first step in working with Oracle is to create a database for storing your data. Although installing Oracle will give you the opportunity to create a starter database, you will need to create additional databases to support your applications and to isolate certain types of data from others. Also, the characteristics of a database designed to handle online transaction processing (OLTP) data, such as order entry systems and accounting data, are very different from those used to store data for online analytical processing (OLAP) and decision-support systems (DSS), such as data warehouses or data marts. For this reason, you need to create many different databases.

You may need to create databases to replace existing ones or to migrate data from earlier versions of Oracle. You may also need a database to support a new application that has been purchased by the enterprise, such as an enterprise resource planning (ERP) application, or to support analysis of existing data so that you can make more informed decisions on future direction. Whatever the reason, as a DBA you will most certainly have to create a few databases in your career.

A single server running Oracle can have many databases on it. Each database can be administered by different DBAs or the same group of DBAs can be responsible for all of them. The ideal situation, if budget were no object, would be to have only one database on each server and in this way ensure the optimal performance for the database by not having other databases competing for system resources. Part of the database creation process also involves properly configuring each database's resource requirements to reflect the estimated demands for its data, all the while keeping in mind what impact those resource requirements may have on other applications running on the same server. Monitoring resources on the server is typically done by the system administrator but needs to be a consideration for the DBA as well.

The process of creating an Oracle database consists of the following steps:

1. Preparing the operating system for the database. Doing so may involve creating directories to hold the data, ensuring that the required services exist (on Windows NT/2000), and verifying that you have the proper permissions to access the resources on the server that will be required by the database creation process.
2. Planning the size and location of the different files that make up the database. These files include the datafiles, redo log files, control files, the parameter file, and archived redo log files. The Oracle Optimal Flexible Architecture provides some guidelines on how you should do this.
3. Creating a parameter (INIT.ORA) file for the database with the correct parameters for the database and its intended purpose. This includes properly configuring the DB_BLOCK_SIZE parameter and the name and location of control files; it also includes ensuring that Oracle takes appropriate resources (such as memory) to run the database.
4. Creating the database. You can use the Oracle Database Configuration Assistant or issue the CREATE DATABASE command to create the database.

Creation Prerequisites

Before you can successfully create a database, you need to ensure that you have the required permissions both within Oracle and at the operating system. For the database creation to be a success, you need to create the folders where the database files are stored. You also need to create the Oracle parameter file for the database and estimate the number and storage requirements of the datafiles that will be required to support the data to be stored in the database.

Operating system considerations

Objective

Prepare the Operating System

When it comes to preparing the operating system for a database, there really is not that much to do. The Oracle executables, if properly configured when the software was installed on the computer, have already been assigned the proper privileges and have membership in the required groups. This is true in both Windows NT/2000 and UNIX environments. As a DBA, you may not actually have the option to change any of the settings for how the Oracle software runs on the machine because this is something that system administrators may do, especially in larger enterprises. The tasks that need to be performed here deal with ensuring that the proper file structure can be created when the CREATE DATABASE statement is executed, and that there is sufficient memory (RAM) on the computer that will house the database.

To ensure that the database creation process works properly, first make sure that you have sufficient free disk space on the hard disks where you intend to place your database files. The amount of disk space required and the number of disks needed will depend on the amount of data that the database will manage, the number of users accessing the data simultaneously, the type of activity on the database, and the number of hard disks available. For example, a database storing order information being accessed by five users will require less storage than one that is being used by five people to analyze sales trends by customers over the last three years. In general, you should estimate more rather than less disk space — data has an uncanny way of growing to fill it anyway.



You can find more information on where to place the different Oracle files later in this chapter, in the chapters dealing with the Oracle files (chapters 6, 7, 8, and 9), and in chapters dealing with database objects that require storage (chapters 10, 11, and 12).

You will need to create the physical folders or directories on those drives to prevent an error being generated when the `CREATE DATABASE` command is executed. Oracle does not create any paths in the hierarchy specified for each file to be initialized when the database is created. This means that if you specify that a datafile will be located in a folder called `/oradb/sales/datafiles`, you will need to ensure that the physical path already exists on the file system before you issue the `CREATE DATABASE` command.

Finally, on the file front, if you will be re-creating a database that previously failed, scan the directories where the different files were to be created for any files with the same name as those to be initialized on the hard drive. Unless you have specified the proper option for the `CREATE DATABASE` command, Oracle will not create a file if a file with the same name already exists. If Oracle finds one file that is the exact same name as the file to be created, Oracle will not overwrite the file but rather will exit with an error. It is a good idea, in any case, to scan the file system paths that the database will use for any files that may be problematic and to delete, rename, or move them before executing the `CREATE DATABASE` command.

After ensuring that the physical file structure is correct, you also need to ensure that the amount of memory that the database instance will require is available on the computer. Because Oracle will, in most cases, allocate its memory requirements at instance startup, you should confirm that sufficient free memory exists on the computer and that Oracle will not adversely affect other applications. To complete this task, you should contact your system administrator and analyze the requirements with the resources available to see whether the chosen server for the database is the right one. As with the amount of disk space required, you will determine the amount of memory needed by what the database will be used for, the number of simultaneous users, and the Oracle options that will be needed for this database.

Permissions

The CREATE DATABASE command is not one that just anyone can execute. To be able to execute it, you must be authenticated as a privileged user on the database—that is, someone who has been granted the SYSDBA or OSDBA roles. You can be authenticated by using the operating system or by password file authentication.



Chapter 2 covers creating a password file and granting individuals privileged user access on the database.

The way that operating system authentication works is different for each operating system; see Chapter 2 for more information. If you want to use password file authentication, you will need to create the password file first by using the ORAPWD utility. The exact name of the password file and its physical location are specific to each operating system that Oracle runs on, so check your operating system documentation to be sure. However, on most platforms, the password file is called `orapw<SID>`, where `<SID>` is the SID for the Oracle database that the password file protects.

For example, if you wanted to create a password file that would grant privileged access to ten users for a database and instance called CERTDB and set the password for the INTERNAL account to be oracle, you would issue the following command:

```
orapwd file=orapwCERTDB password=oracle entries=10
```

After you create the password file, make sure that you place it in the proper folder of the hard drive so that Oracle will find it. On Windows NT/2000 and most UNIX platforms, this is the DBS folder in the Oracle home directory (ORAHOME).

The final step is to add the Oracle user accounts that will issue the create database command to this password file. To do this, you will need to start the instance. Because you have not created a parameter file yet, you will need to wait until you have performed that step to update the password file.

Planning database file locations

When creating a database, part of the role of the DBA is to ensure that files are placed to provide for good recoverability as well as good performance. This means that, where possible, the DBA should place multiple copies of key critical database files on several physical disks. This also means that the DBA needs to place files that may have a high probability of contention on separate physical disks to avoid introducing a point of contention. In deciding the placement of files, you need to be concerned about control files, redo log files, and the Oracle datafiles before creating the database.

Control file placement

As you will find out in Chapter 6, the Oracle control file is a critical file for any database. It contains information about the location of all other files that the database uses. In fact, you can't start an Oracle instance if the control file is lost or damaged in any way. For this reason, Oracle recommends that you create multiple control files when you configure the parameter file.



For information on how to add more control files after the database has already been created, please refer to Chapter 6.

During the database creation process, Oracle reads the parameter file to determine the locations of the control file to be created. If the parameter file specifies more than one control file, Oracle will create both at this time. To ensure that database recoverability remains high, you should specify at least two control files in the parameter file with the `CONTROL_FILES` parameter, as follows:

```
CONTROL_FILES=(/certdb/disk1/CONTROL01.CON,  
/certdb/disk2/CONTROL02.CON)
```

When you issue the `CREATE DATABASE` command, Oracle will create both files and provide for fault tolerance should you lose a disk.



Chapter 6 provides basic information on how to further protect the control file through backups and other means. Please consult it for further information on the control file.

Redo log file placement

The Oracle redo log files are critical to the good recoverability of the database. As with the control files, the redo log files are initialized when the database is created and maintained thereafter. The redo log files contain information on the chronological changes made to the data in the database, and you can use them to recover all the data in the case of failure.



Chapter 7 covers redo log files.

Oracle places redo log files in redo log file groups, which are a collection of one or more individual redo log files, called members, that will have the same information written to them. Because all members of a redo log file group have identical contents, if a member of a group is not available when a write needs to be made, as long as another member of the same group exists, Oracle can perform the write and thereby ensure that the database can be recovered. However, if no members of a redo log file group are available—because only one was created and no others were added, for example—Oracle cannot perform the write. The unpleasant side effect of this is that you can't make any changes to the database, and it simply stops; it allows no further modifications unless the problem is corrected.

The way redo log files work is quite simple. When you make a change to the database, such as deleting, updating, or adding a record in a table or creating, updating, or dropping database objects, these changes are first recorded in the redo log files. Before a transaction can be committed, a write must be made to the redo log file to ensure that a physical record of the change exists. This means that redo log files are the most write-intensive files in an Oracle database.

When deciding the placement and number of redo log files, you will need to keep both the recoverability aspects and the write-intensive nature of the files in mind. This means that you should create at least two redo log file groups (preferably three) with two members each on physical disks that are separate from the rest of the database. You should not place redo log files on the same physical disks as datafiles because doing so may have a negative impact on performance owing to the write-intensive nature of the files. Furthermore, because these files are so critical, having two members in each redo log file group, with each member on a separate physical disk, allows you to mirror the contents of the files and ensure that the database can be recovered in case of physical disk loss.

Data file placement

In deciding on the placement of datafiles for the database, at the time of database creation you should know the characteristics of the data to be contained in the database, as well as how it will be accessed and which tables will be the most read- or write-intensive. In other words, before creating a database, you should apply the KYD factor — Know Your Data. The KYD factor will help you determine how many datafiles you require and where they should be placed to minimize contention and optimize disk performance. In practice, if you are implementing a new database for an application that has never been deployed, this might be more difficult. If the application is being purchased from a third-party vendor, the vendor may be able to offer some assistance.



Many Oracle databases are used for the deployment of enterprise resource planning (ERP) or other systems being purchased from a third party. In this situation, the vendor must be able to offer some guidelines on how much data storage you will require and the optimal placement of datafiles, redo log files, and control files. A vendor's inability or unwillingness to provide this kind of information should raise a flag in your mind.

When deciding on the placement of datafiles, keep these guidelines in mind:

1. Place the SYSTEM tablespace and its datafiles on a physical disk separate from the data and index tablespaces.
2. Place datafiles for tablespaces containing tables, indexes, clusters, and partitions on separate disks from datafiles containing indexes. Doing so ensures that index IO operations and data IO operations do not contend with each other.
3. Create multiple tablespaces to house rollback segments and temporary segments and place their datafiles across several disks to balance out the IO across multiple drives.

4. Place datafiles for tablespaces with different characteristics on separate physical disks. This means that you should keep tablespaces with high fragmentation propensity separate from those with a low fragmentation propensity. Furthermore, if data has different life spans, such as temporary objects and application data, you should place these on different disks.
5. If you are partitioning tables, place the datafiles for each partition in different physical disks to minimize the possibility of IO contention. Doing so typically is a good idea in data warehouse situations in which a large fact table is partitioned across multiple tablespaces. Having the datafiles for each tablespace on a separate disk will allow for multiple simultaneous reads to take place and provide better performance.
6. If you are using hardware RAID, ensure that the redo log files are not on the RAID strip set but only the datafiles. Datafiles may be all right to place on a RAID array, but redo log files may negatively impact performance because of their write-intensive nature, and you should always place them on separate physical disks.



Chapter 8 provides more detail on key criteria for the placement of datafiles for your Oracle database. Chapter 9 provides additional information on the Oracle storage structures.

Oracle's Optimal Flexible Architecture

With the introduction of Oracle8, Oracle also provided a set of guidelines on where to place different files in a database. When creating a database with the Database Configuration Assistant, you use the Optimal Flexible Architecture (OFA) to decide where to put each file that makes up the database.

A team at Oracle responsible for tuning and installing UNIX-based Oracle databases wrote the Optimal Flexible Architecture. It is designed to facilitate a structure on your server that makes administering and facilitating the growth of databases easy. It includes provisions for hosting multiple databases on the same server, locating the datafiles for each database easily, and minimizing IO contention by properly placing files on disks in such a way that files with high probability of contention are physically separated. The OFA also takes into account the addition of drives to more smoothly even out the IO on the system.

When you install Oracle, the Oracle Universal Installer automatically uses OFA to decide on the best place to locate the Oracle executable and support files, as well as the location of database files if a started database is installed. When you run the Database Configuration Wizard, Oracle also uses the OFA structure to assist in the determination of the best location of your datafiles, redo log files, and control files.

The Optimal Flexible Architecture consists of the following structural elements, some of which are defined as environment variables on the target operating systems:

- ♦ **ORACLE_BASE**— This environment variable points to the root of the OFA directory tree. When you install Oracle8i, ORACLE_BASE is set to the name of the folder that will host your Oracle software: C:\ORACLE. This location is where software is installed and is the root for multiple ORACLE_HOME folders, in case multiple versions of Oracle are installed on the same machine.
- ♦ **ORACLE_HOME**— This environment variable points to the location of the Oracle software and (only indirectly) data. When you install Oracle on your server, you are prompted for the location of the ORACLE_HOME and, if it does not exist, Oracle creates it. Each ORACLE_HOME can have a different version of the Oracle software, and multiple ORACLE_HOMES can exist on the same server, thereby allowing you to run multiple versions of Oracle on the same machine, which might be necessary during a migration or an upgrade. Oracle provides a way to switch between multiple ORACLE_HOMES on the same machine to ensure that the proper version of the product is run.
- ♦ Individual product files for each version are stored in separate folders to avoid causing any potential conflicts. The same goes for database creation and other scripts, administrative utilities, parameter files, and other software used for each version of Oracle, each database, or both. For example:
 - A directory for storing Oracle server data (typically called oradata) on each of the physical disks present. OFA will make use of all available physical devices and will create a folder on each hard disk to hold the data for the different databases created. This organization makes locating Oracle data and other files easy when you're browsing the contents of the hard disk.
 - A separate physical directory on each disk under the oradata directory for each database's files. If you had a database called CERTDB, a folder called oradata/CERTDB would be created on each physical disk that OFA decides to use. This arrangement would allow the placement of data and other files on multiple physical disks and provide better performance as well as increase recoverability.

A typical structure on Windows NT/2000 might look like this:

```

C:\ORACLE          --- The ORACLE_BASE directory

C:\ORACLE\ADMIN    --- The admin scripts and product files for the databases
                   --- created. A separate directory is created under the
                   --- ADMIN directory with the same name as the SID of the
                   --- database to store its files, e.g. INIT.ORA.

C:\ORACLE\ORA81    --- The ORACLE_HOME directory for Oracle8i. This is the
                   --- location of the Oracle executable files and all support
                   --- files for the particular version installed. Multiple
                   --- directories can exist, one for each version installed.

C:\ORACLE\ORADATA  --- The location of the datafiles for each database
                   --- created. A separate physical directory exists with
                   --- the same name as the SID of the database under the
                   --- ORADATA directory.
  
```

```
D:\ORACLE\ORADATA --- If multiple hard disks are found on the system, OFA may
E:\ORACLE\ORADATA --- create additional directories on those drives to hold
F:\ORACLE\ORADATA --- the datafiles for the databases created. Under each
--- ORADATA directory, multiple directories with the same
--- name as the SID of the database may exist.
```

Although Oracle will create this structure for you, you are not bound by OFA and may choose your own locations for your data and other Oracle database files.

When determining the placement of files, you should keep the principles of OFA in mind to ensure that your database runs well.



When taking the Oracle exam, you will be expected to know the OFA directory structure and where different files are stored. Although you may not encounter too many questions dealing with OFA directly, other questions on file placement will require knowledge of it because it is the Oracle-recommended method.

Creating a Database

Objective

Create the database

You are ready to create your database after you have done the following: ensured that the Oracle files are properly placed to meet the needs of the database and to make optimal use of the existing physical disk structure; the parameter file has been created; the operating system has been prepared. You can create an Oracle database using the Database Configuration Assistant, a GUI tool that guides you through the process, or you can do it manually using the CREATE DATABASE command.

Using the Oracle Database Configuration Assistant

The Oracle Database Configuration Assistant is a GUI utility that prompts you for all the settings needed to create an Oracle database. It will create the parameter file and place the datafiles, control files, and redo log files in the appropriate locations defined by OFA. Prior to Oracle8i, it was available only on Windows NT/2000, but is now a standard part of every Oracle installation. You can also use it to modify settings or delete a database from the computer if necessary.

One of the main advantages to using the Oracle Database Configuration Assistant is its ability to update the LISTENER.ORA and TNSNAMES.ORA files on the computer where the database is being created. This, in essence, performs the basic configuration of the Oracle networking component, Net8, and allows you to access the database you create using GUI tools such as DBA Studio, Enterprise Manager, and SQL*Plus. The preconfiguration of Net8 takes place whether you choose to allow the Database Configuration Assistant to create the database for you or save the results of your selections in script files. This preconfiguration can also be a problem if you saved the scripts and later decide not to create the database: you will need to remove the network entries manually using Net8 Assistant or by editing the files.



Configuration of Net8 is beyond the scope of this book and is not covered on this exam. Oracle has another exam specifically dealing with network administration in an Oracle environment.

Running the Database Configuration Assistant

To start the Database Configuration Assistant on Windows NT/2000, select Start ⇨ Programs ⇨ Oracle with select your Oracle Home ⇨ Database Administration ⇨ select Database Configuration Assistant. The screen shown in Figure 4-1 will be displayed. The main startup screen allows you to specify whether you want to create a new database (the default option), modify an existing database (for example, change the password for the INTERNAL account), or delete a database. The Create option will create all the necessary files, as you will see. The Delete option will remove a database and all of its services (on Windows NT) as well as all the data and other files.

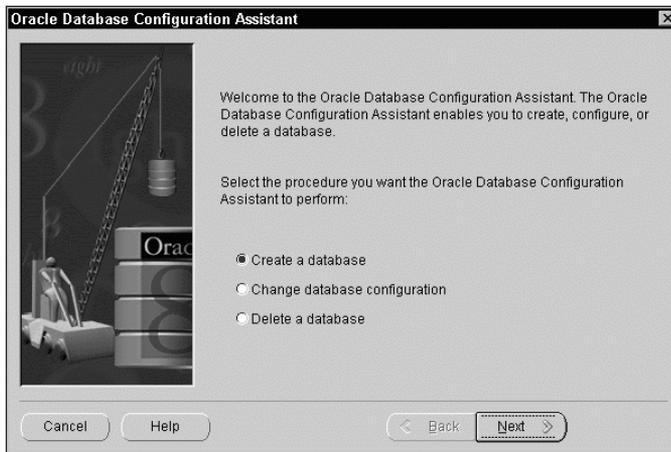


Figure 4-1: The Oracle Database Configuration Assistant startup screen allows you to select whether to create, modify, or delete a database.

If you choose to create a database, you will be presented with the screen shown in Figure 4-2, which asks whether you want to create a database using a Typical configuration or specify the various settings yourself using a Custom configuration. The Typical option is the quickest way to create a database, whereas Custom gives you more control over everything that the Assistant does. Furthermore, selecting Typical will not allow you to save the configuration in scripts for running later but will require that you agree to have the Database Configuration Assistant create the database when it completes.

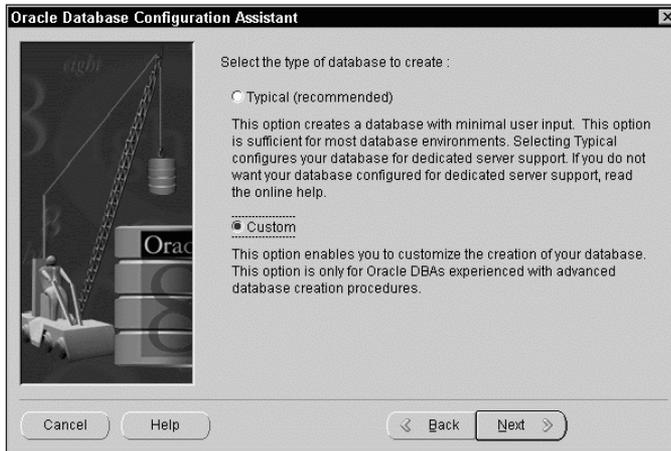


Figure 4-2: If you choose to create a database, you must select whether to create a typical database or perform a custom installation.

If you selected Typical, you are presented with the screen shown in Figure 4-3, which asks whether you want to select a CD-based creation, in which the Database Configuration Assistant copies files from the CD to create the database, or create the database files from scratch.

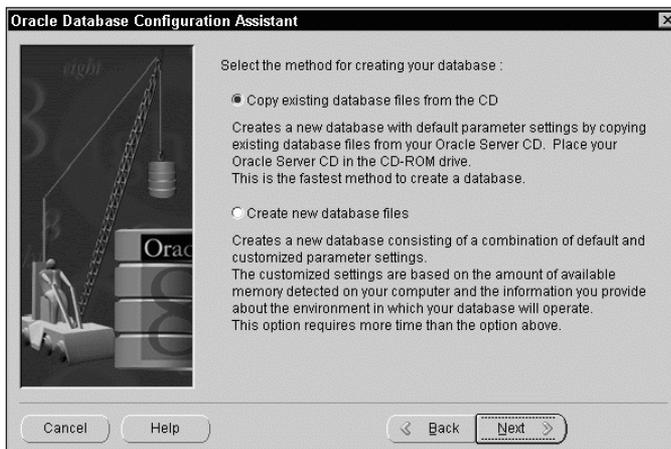


Figure 4-3: The Typical option allows you to perform a quick CD-based database creation or create new database files.

The CD-based option takes ready-made Oracle files and copies them to your hard drive to create the database. Essentially, it creates a clone of a typical database configuration and does not allow you to make any changes to these settings. Except

for testing or to create a database for education and learning, it is not recommended that you select this option. The configuration of this database may not match your specific application requirements and you may introduce inefficiencies at the outset.

The option to create new database files will provide some control in a Typical installation because later you will be prompted for your specific database characteristics, but a number of things will be assumed (the placement of data, control, and redo log files, for example) and will be created to conform to OFA. If you want to have the option to locate Oracle files in specific locations on your hard drives, you should choose a Custom install. You can do this by clicking Back in the Database Configuration Wizard and then selecting Custom.

If you chose a Custom database creation, or select Create New Database Files in a Typical database creation, you will be prompted for the characteristics of your database, as shown in Figure 4-4. Your choice here will have an impact on how the parameter file is configured as you step through the wizard; it will also affect how files will be placed by default later in the wizard. Each database environment makes certain assumptions, as described in Table 4-1.

Table 4-1
Oracle Database Environment Types

<i>Environment</i>	<i>Characteristics</i>
Online Transaction Processing (OLTP)	This environment assumes that your database will process large volumes of database modifications such as INSERTs, UPDATEs, and DELETEs. Users need to be able to retrieve individual or small groups of data quickly but typically will not be reading large volumes of data at one time. The data typically consists of small records (100–400 bytes). The emphasis is on fast retrieval of current data and the capability to support many changes to the database without creating a bottleneck. This database environment is very update intensive.
Decision Support System (DSS)	Decision-support systems, such as data warehouses or OLAP environments, are used to query large volumes of data typically from several different tables at the same time. Result sets in this environment can consist of thousands or even millions of rows. Complex queries may frequently be processed by the system and the database should be able to retrieve data as quickly as possible. Updates and modifications are done rarely or on a scheduled or batch basis, so there is no real need to optimize the database for writes.
Multipurpose (sometimes also called hybrid)	Hybrid systems are those that need to support both OLTP and DSS types of functionality. In a hybrid environment, settings provide a middle ground between optimal OLTP performance and optimal DSS performance. This is the default selection.

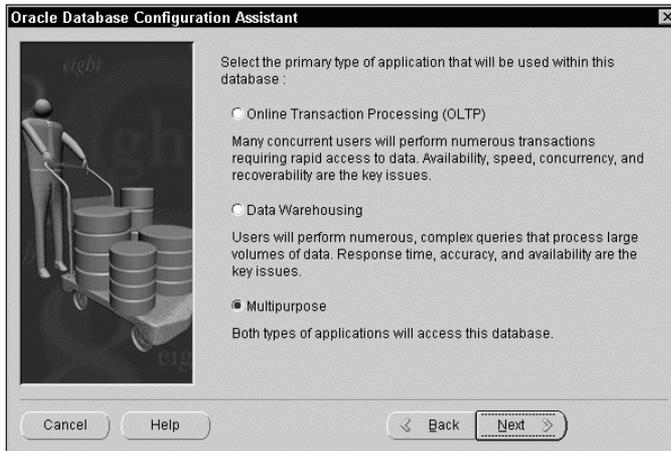


Figure 4-4: If you selected a Custom database creation or the Create New Database Files option of a Typical creation, you will need to choose the characteristics of the database to be created.

After making your choice regarding the type of environment that your database will be used to support, you will be prompted to specify the number of users who will be accessing your database simultaneously (see Figure 4-5). The default selection is 15 but may be changed to correctly indicate your requirements. This number has an impact on the parameters configured in the INIT.ORA file and the sizing and number of rollback segments and temporary tablespaces. Enter the number or click the up or down arrows and then click Next to continue.

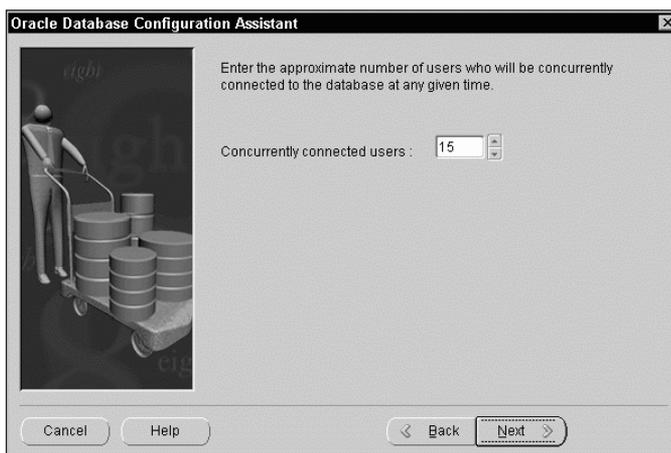


Figure 4-5: If you selected a Custom database creation or the Create New Database Files option of a Typical creation, you will need to specify the number of users that your database will support.

If you selected a Custom database creation, you will be prompted to decide whether your database will support dedicated server mode or multithreaded server (also known as Shared Server), as shown in Figure 4-6. In most cases, you should choose Dedicated Server mode. If you are anticipating a large number of users (400 or more) to be accessing this database, you may select Shared Server. Shared Server will make better use of system resources for large numbers of users by having users connected to the database share the Server Process. If you select Shared Server, Dedicated Server is also supported implicitly—that is, all Oracle databases support Dedicated Server mode.

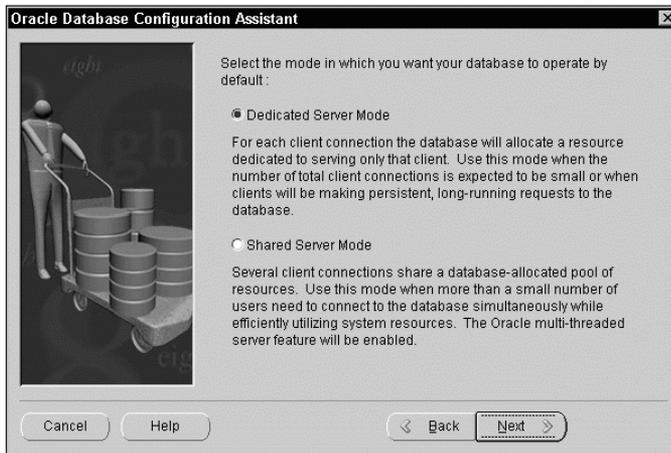


Figure 4-6: If you selected a Custom database creation, you will be prompted to specify whether your database will support Dedicated Server or Shared Server mode.

In Dedicated Server mode, when a user connects to the database instance, a Server Process will be launched to do work on that user's behalf. If the user is not accessing the database, memory and a process slot for the server are still being allocated, which may produce a lack of resources if too many users are connecting to the instance. In Shared Server mode, the Server processes are started as needed by Oracle and shared among all the users connecting. This means that the same hardware and database instance can support more users than in Dedicated mode, because users not performing any work on the database take up less memory and no CPU.

After clicking Next, you can select which Oracle options will be installed and configured for the database. The list presented, which is shown in Figure 4-7, will include only those options that have been purchased and installed on your hard drive when you installed the Oracle software. To find out more about each option, refer to the Oracle documentation. In general, you should install SQL*Plus Help and, if you plan to support Java in your database, the JServer option. Other options exist for more specific characteristics of your database, such as the ability to replicate data between Oracle databases or to use the database to store videos and images. This screen will appear in a Custom database creation or if you selected Create New Database Files for a Typical creation. After making your choices, click Next.

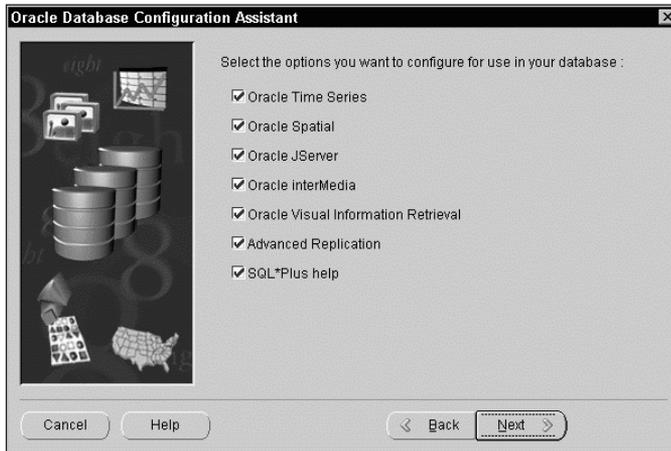


Figure 4-7: If you selected a Custom database creation or the Create New Database Files option of a Typical creation, you will be prompted to specify which Oracle options should be installed.

The next screen asks you to input unique identification information for your database. This includes the Global Name for the database in the format of *database.domain*, in which *database* is the name of the database and *domain* is a DNS-style domain name to which the database belongs. The SID identifies which instance will access the database and uniquely identifies each instance on the server running Oracle. In all environments, the SID cannot be larger than eight characters (prior to Oracle8i, the SID could not be more than four characters on Windows NT). Figure 4-8 shows how a Global Database Name and SID might be entered.

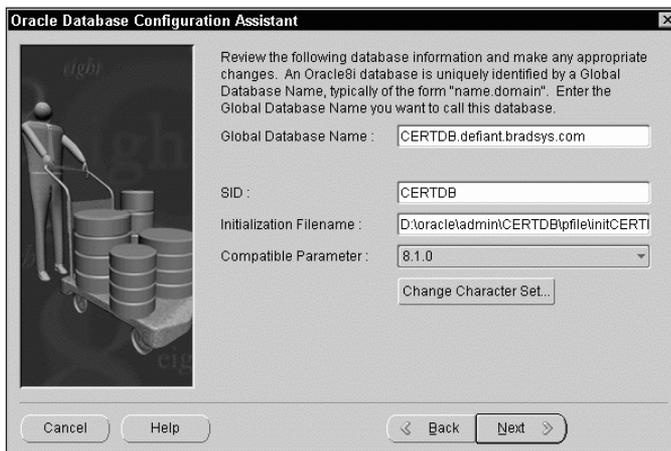


Figure 4-8: You must enter a SID and Global Database Name to uniquely identify your database to Oracle.

If you selected a Typical database creation, you will have only the option to input the Global Database Name and SID. If you are creating your database using a Custom option for the Database Configuration Assistant, you will also have the capability to specify the following: the location of the parameter, that is, the `INIT.ORA` file; a compatible value for the database; and the database character set and national character set.

The default for the compatible parameter is 8.1.0, which makes the database act as though it is an Oracle 8.1.0 database — the original release of Oracle8i. The compatible parameter determines the behavior of certain Oracle utilities as well as internal procedures when performing their designated tasks. You should accept the default choice unless your application requires a different setting or you have been advised to do so by a vendor or other party. Because changing the parameter value may affect how a database operates, it should be set with plenty of consultation regarding the impact of the change.

The database character set determines what types of data will physically be stored in character columns of your tables — that is, columns with datatypes of `CHAR`, `VARCHAR2`, `LONG`, and `CLOB`. The default presented to you will be the best one determined by Oracle based upon the configuration of your machine, including language and regional settings as well as the language of the version of Oracle you purchased. In other words, depending on where you are in the world, you may see a different selection. In most cases, you should accept the default unless your application requires specific settings. If you decide to change the character set or need to specify a different national language character set, click `Change Character Set` and you will see the dialog box shown in Figure 4-9.



Figure 4-9: In a Custom database creation, you have the option to change the character set and National Language character set of your database.

You now have the option to change either or both the character set and National Language character set of your database. These should, in most situations, be left at the default and be the same. For situations in which you want to store character data with different character sets in columns with a `NCHAR`, `NVARCHAR2`, or `NCLOB` datatype, you can specify a different National Language character set. An example of when doing so might be necessary is having your database used by individuals in your Hong Kong and U.S. offices. You may choose a character set that corresponds

to the U.S. office's requirements, but also choose a Chinese-language character set for the Hong Kong office. Your application would also need to ensure that tables had both national language and regular character columns, and it needs to be able to determine where the data is being input from to place it in the right columns.



In practice, most databases will contain data only for one character set because having more than one makes the development of applications more difficult. If another character set needs to be supported, most organizations create another database.

You should note that after you have selected your character set and National Language character set, you can't, in many situations, change them except by recreating the database. Oracle8i provides ALTER DATABASE CHARACTER SET and ALTER DATABASE NATIONAL CHARACTER SET commands to change the character set and National Language character set, but only if the one you are changing the database to is a strict superset of the current one. For this reason, you need to ensure that your choice will support the type of character data that will be resident in the database. If you are not sure, check with the vendor or your developers. After making your choice, click OK and then Next to continue with the Assistant.

If you are performing a Custom database creation, you will next be prompted to enter a password for the INTERNAL privileged account, as shown in Figure 4-10. This account has the SYSDBA role necessary to create a database and can be used when you need to perform privileged options. If you need to perform privileged operations, such as startup and shutdown of the database instance, you need to be connected as a privileged user. The INTERNAL account is the only one created by default, and its password needs to be specified. If you are performing a Typical database creation, the INTERNAL account password will be automatically set to oracle. In a Custom creation, you have the option to specify the password. You cannot leave the password blank.

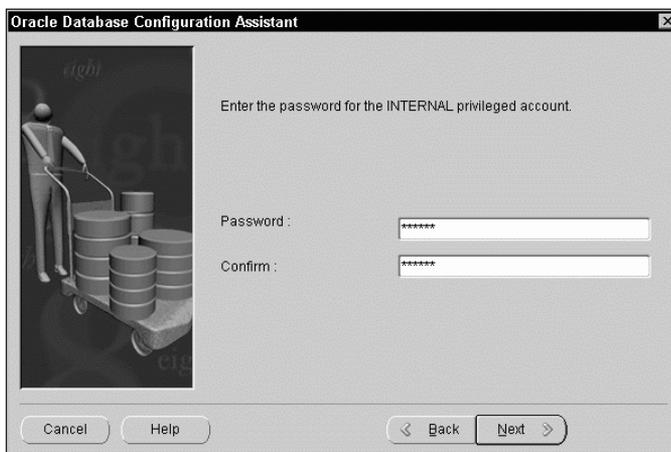


Figure 4-10: In a Custom database creation, you need to enter a password for the INTERNAL privileged account.

The next series of screens are available only if you are performing a Custom database creation with the Database Configuration Assistant. The first of these asks you to specify the location of the control files for your database and will present a recommended location, as shown in Figure 4-11. In most cases, you should accept the default presented. However, before doing so, ensure that your control files are located on at least two physical disks (if installed on the computer). This is strongly recommended by Oracle to ensure that your database does not become inaccessible if you lose all your control files.

On the same screen, you will also be asked to set some database limits. These are used to actually size the control file when the database is created. Ensure that you have a sufficient number of datafiles, redo log groups, and redo log members specified to support your application. Once set, these values cannot be changed except by re-creating the control file—not a pleasant experience. If you are not sure, choose larger rather than smaller values. One value to change is the Redo Log Members; the default of 2 provides for minimal redundancy in your redo log files and should be increased to at least 4 or 5. The defaults for datafiles and redo log groups are sufficient in most cases. After you have made your choices and verified the settings, click Next.

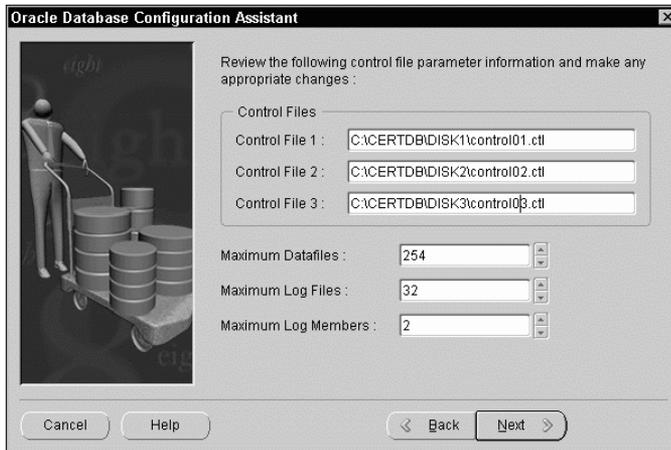


Figure 4-11: In a Custom database creation, you will be prompted to confirm the location of the control files for the database and specify the maximum number of datafiles, redo log groups, and redo log members.

In a Custom database creation, you will next be presented with a tabbed dialog box, as shown in Figure 4-12. Here, you will be presented with default choices, based upon answers to questions previously asked by the Database Configuration Assistant, for the size, location, and storage characteristics of the various tablespaces that make up your database. The Database Configuration Assistant requires you to follow the Optimal Flexible Architecture and you must specify these

values for all the tablespaces that it wants to create, including SYSTEM, USER, TOOLS, INDEX, ROLLBACK, and TEMPORARY. You should verify that the datafiles corresponding to the tablespaces are sized appropriately and distributed among your physical disks to minimize contention. Changing the size of the SYSTEM tablespace's datafiles is usually not a good idea because this value is calculated to hold the objects needed for the Oracle options that you previously selected to install. You can change the values of other tablespaces by selecting the tab for the appropriate tablespace and changing the value. After you have made appropriate choices, click Next to continue.



Please see Chapter 8 for a full discussion on the number of tablespaces your database should have and what they should be used for. The OFA-compliant dialog box presented by the Database Configuration Assistant in Custom mode assumes that you are familiar with the issues presented in Chapter 8.

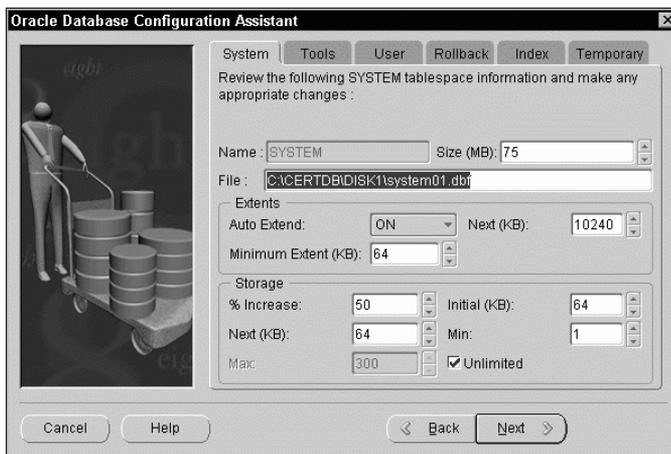


Figure 4-12: In a Custom database creation, you have the option to specify the placement and size of datafiles for your tablespaces.

The next screen (see Figure 4-13) in Custom database creation mode asks you to verify the names and locations of the redo log file groups and the first members of each group for the database. In most cases, the default number of redo log file groups and their sizes are sufficient. You should verify that all redo log file groups to be created are the same size and that their filenames allow for the easy creation of additional members if needed. For this reason, you will probably have to change the default names to something like redo01a.log from redo01.log. That way, if you need to add a second member to the redo log file group, you can name it redo01b.log and know by the filename that the log member belongs to the first redo log file group. Oracle requires that your database have at least two redo log file groups—but three is better, as you will find out in Chapter 7.

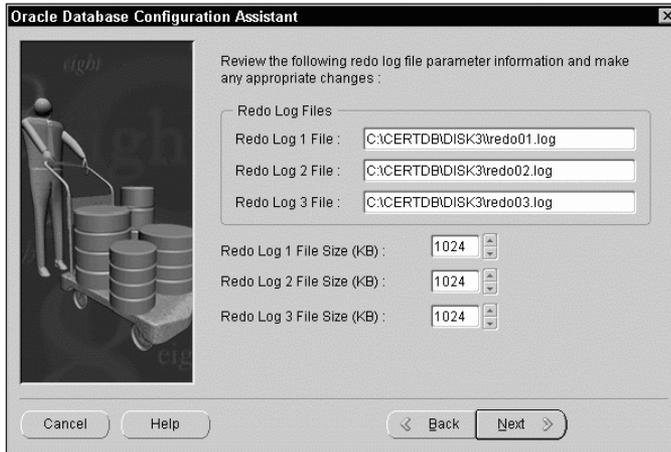


Figure 4-13: In a Custom database creation, you have the option to specify the placement and size of redo log files.

Clicking Next presents a dialog box, as shown in Figure 4-14, that allows you to specify values for the CHECKPOINT_INTERVAL and CHECKPOINT_TIMEOUT initialization parameters. The defaults presented for these should be sufficient in most cases.

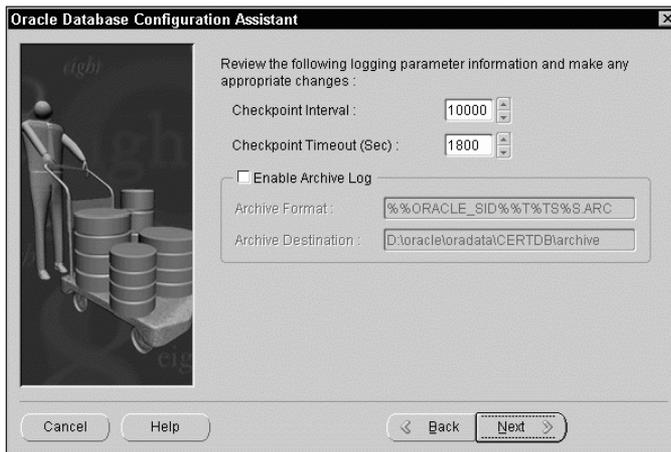


Figure 4-14: In a Custom database creation, you have the option to verify and change checkpointing parameters and turn on archiving.

Another option that you can select on this dialog box (this option is turned off by default) is whether to place the database in ARCHIVELOG mode upon creation and, if so, where the archived redo log files should be placed. Archiving is a process of copying redo log files to a specific location after a log file switch takes place. Archived redo log files allow you to perform advanced recovery options, such as point-in-time recovery, and is recommended for OLTP environments. However, it is not recommended that archiving be turned off until after the database is created and loaded with tables and data.



Cross-Reference

Chapter 7 covers checkpoints and archiving in greater detail. A full discussion on archiving is beyond the scope of this book.



Exam Tip

You will not be tested on recovery issues dealing with archiving on this exam. These issues are covered on the Oracle8i Backup and Recovery exam.

The screens shown in Figures 4-15 and 4-16 are shown to you only if you have chosen a Shared Server configuration for the database to be created. It allows you to set the various options required for a multithreaded server database environment, such as the initial and maximum number of dispatchers for each of the protocols you specify, the initial and maximum number of shared server processes, and other settings. After you have made your choices, click Next to continue.

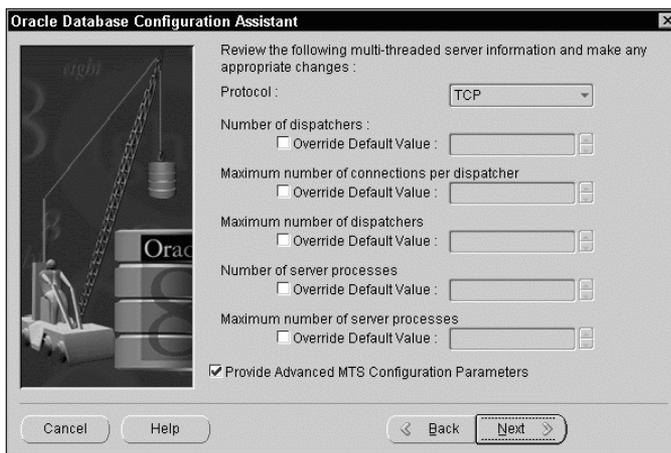


Figure 4-15: If you selected a Shared Server environment in a Custom database creation, you now have the opportunity to configure MTS settings.



Exam Tip

This exam does not cover how to configure and create a multithreaded server environment. Therefore, you will not be tested on your knowledge of the options presented by this screen.

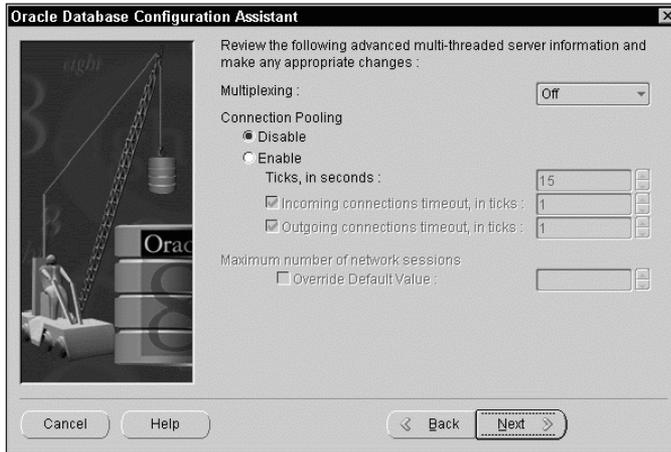


Figure 4-16: If you selected a Shared Server environment in a Custom database creation, you also have the option to indicate whether Multiplexing or Connection Pooling should be enabled.

The next screen (see Figure 4-17) allows you to configure and confirm the initialization parameter values for the instance, including the following:

- ♦ The size of the shared pool (SHARED_POOL_SIZE)
- ♦ The number of database buffers in the cache (DB_BLOCK_BUFFERS)
- ♦ The size of the large pool (LARGE_POOL_SIZE)
- ♦ The size of the redo log buffer (LOG_BUFFER)
- ♦ The maximum number of background and server processes (PROCESSES)
- ♦ The size of the database block to be used by the database (DB_BLOCK_SIZE)

These values should be compatible with the resources available on the computer, particularly memory, and the target database environment. The Database Configuration Assistant will request information from the operating system on available RAM and other resources and will take the database environment into consideration when presenting you with its determination. In many cases, letting the Wizard make these choices works well; however, if other applications are to be installed on the same server, you should verify that Oracle's settings won't place an undue strain on the computer. In any case, you can change these values later, with the exception of DB_BLOCK_SIZE, but not without shutting down the instance.



Figure 4-17: In a Custom database creation, you will be prompted to verify the values of several key initialization parameters.

The next screen prompts you to specify the location of user and background process trace files, that is, the values for `USER_DUMP_DEST` and `BACKGROUND_DUMP_DEST` initialization parameters, respectively. (See Figure 4-18.) You may want to redirect the `USER_DUMP_DEST` location to a mapped drive corresponding to the user's home directory to ensure that each user has his or her own set of trace files.

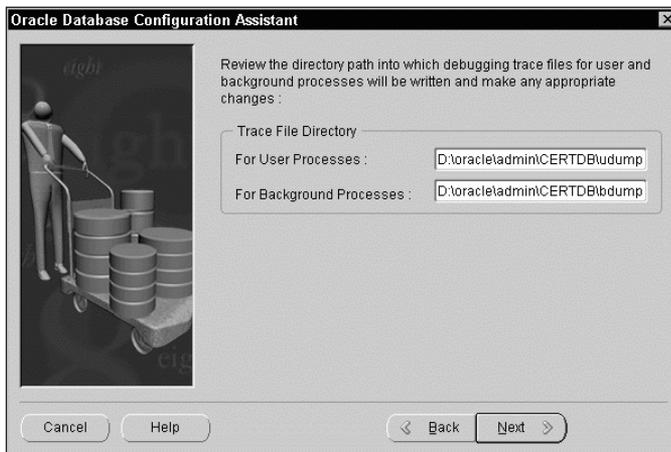


Figure 4-18: In a Custom database creation, you will be prompted to set and verify the location of background and user process trace files.

The final screen (see Figure 4-19) in a Custom installation, or in the Create New Database Files option of a Typical installation, asks whether you would like to create the database right away or save the scripts to create it on your hard drive for later execution. This screen is not available in a CD-based typical installation because the Assistant simply copies files from the CD-ROM, which must be inserted in the CD-ROM drive, onto the hard disk. Because the creation of a new database and new files can be a lengthy process, you are given the option here.

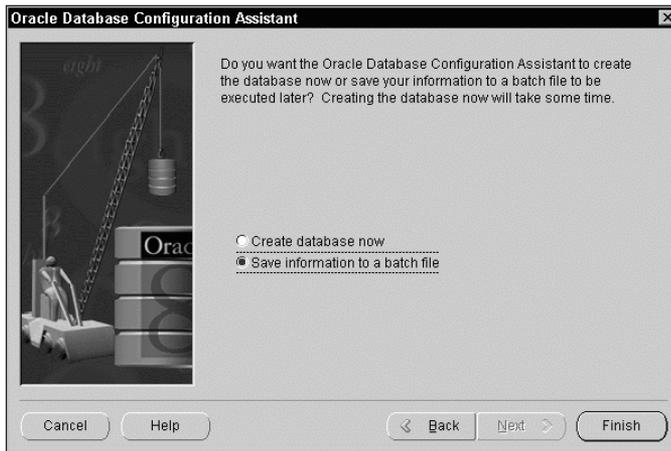


Figure 4-19: You have the option to create the database right away or create scripts to be run later.

In all cases, it is recommended that you save the information to a batch or shell script file and have the Database Configuration Assistant create the scripts. This way, you can review and make changes to the scripts before you run them in case some of the settings do not match what is required, as well as to have a way to re-create the database in case the process fails the first time. This is a lot easier than trying to remember what settings you entered in the Assistant.

When you decide to save the scripts, you will be prompted for a location. You can choose to place the scripts in the default location provided or, better still, place the files in one of the directories that will be used by the database—such as the one that holds the Oracle parameter (INIT.ORA) file for the database. When all the scripts have been saved, you will see a screen similar to that shown in Figure 4-20.

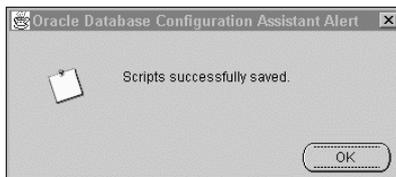


Figure 4-20: You have successfully run the Database Configuration Assistant.

The files that are created by the database configuration assistant will vary, depending upon the Oracle options that you have chosen to install. You will usually have the following files:

- ♦ `INIT<SID>.ORA` — The Oracle initialization parameter file for the database.
- ♦ `create<SID>.bat` — A Windows NT/2000 batch file to create the database and run all the necessary commands. The file will be used to call the various programs required and pass them the script names to create the database and install any selected options. On UNIX systems, the file will be a shell script rather than a BAT file.
- ♦ `<SID>run.sql` — An Oracle script that issues the `CREATE DATABASE` command to create the database.
- ♦ `<SID>run1.sql` — An Oracle script that creates additional tablespaces and the standard data dictionary views.
- ♦ `<SID>altertablespace.sql` — An Oracle script to alter the default and temporary tablespace for the user `SYSTEM`.

You may have additional files if you have decided to configure other Oracle options.

Reviewing the contents of these files before running them is a good idea so that you have an idea of what is happening. A final note: Creation of a database with the Database Configuration Assistant can take quite a while and place a very heavy burden on the CPU and hard disks of the computer after the process starts, so consider running the database creation scripts during off hours. Feel free to take lunch or a break after they start — they take 30–60 minutes on average to complete.

Creating a database manually using the `CREATE DATABASE` command

Objective

- Prepare the operating system
- Prepare the parameter file
- Create the database

Although the Oracle Database Configuration Assistant makes the job of creating your database a point-and-click operation, the actual steps that it goes through are the same ones you would need to accomplish when you create your database manually. Basically, you need to perform the following to create a database:

1. Determine a unique instance and database name; also select a database character set and National Language character set.
2. Configure the operating system and the environment to support starting the instance and creating the database.
3. Create and configure the Oracle parameter (`INIT.ORA`) file.

4. Create a password file for authentication of privileged users.
5. Start the instance.
6. Create the database.
7. Create the data dictionary views and standard packages; configure selected options for the database.

Of the preceding steps, the first six are covered in this chapter. The last step is covered in Chapter 5.

Identifying the database

The first step in the creation of a database is to select a name or system identifier (SID). Although Oracle places virtually no restrictions on what name you can use for your instance and database, you are limited to eight (8) characters. Furthermore, the name of the instance and database must start with a letter and cannot contain any reserved characters such as \$ or #. The Oracle community generally accepts that SIDs should contain only letters and numbers.

A SID must be unique on the computer on which Oracle is running. This means that if you have several different databases on a computer, each must have its own unique SID. Because the database will most likely be accessed across the network, the Global Database Name, of which the SID is a part, must also be unique across the network. You specify the Global Database Name when you create the parameter file.

To ensure that the SID for your database is unique, you can verify which other SIDs have been used on the target computer. In a Windows 2000 environment, you do this by using Administrative Tools/Services to get a list of all services that start with “OracleService” (the list of services on a Windows NT system is found by using Control Panel/Services). The characters after “OracleService” in the name of the service indicate the SID that is already in use and cannot be used, unless the service is removed from the computer. For example, as shown in Figure 4-21, you could not use a SID name of ORCL on this computer because a service called “OracleServiceORCL” already exists.

A UNIX system has a corresponding utility to determine which SIDs are being used. One way to determine which SIDs may potentially be unavailable is to use the UNIX *find* command to search the hard disk for all Oracle initialization files; from that you can discern which SIDs have already been allocated. The syntax to type and the UNIX prompt are as follows:

```
find / -name "init*.ora" - print | pg
```

This command searches the hard disk from the root directory for any files whose name matches the pattern `init*.ora` and then prints the results to the screen. The `| pg` portion tells UNIX to stop and allow you to scroll in case more than a screenful of information is returned.

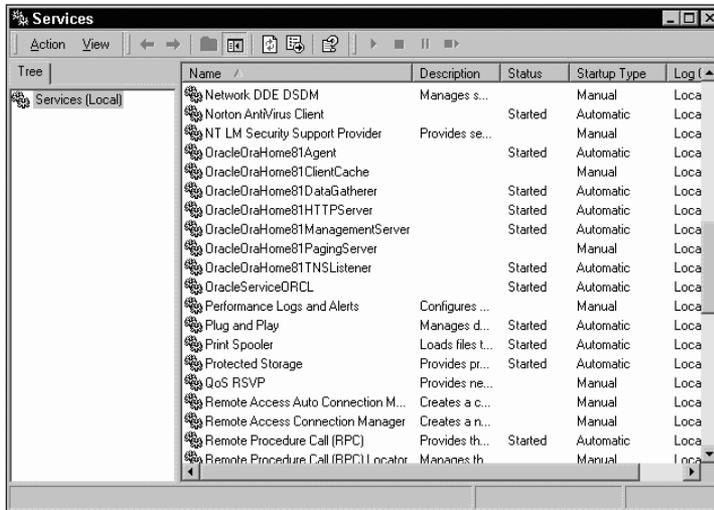


Figure 4-21: Use the Services MMC snap-in in Windows 2000 to get a list of SIDs that are currently in use.

After issuing this command, you can scan through the names of the files presented and take any portion after `init` and before `.ora` as a SID that may be used on the computer. These are not available to you and you must choose another name for the SID.

In both Windows NT/2000 and UNIX environments, if you adhered to the Optimal Flexible Architecture, you can also view the list of directories under the `ADMIN` directory in the location pointed to by the `ORACLE_BASE` environment variable. OFA creates a separate directory for each database created to store the `INIT.ORA` and other files. This will not work if databases were created outside OFA or their locations otherwise altered.

After you have decided on a SID, you should also determine what the database character set and National Language character set should be.

Preparing the operating system for database creation

Before creating a database on your server, you need to configure the operating system environment to successfully create the database. On a Windows NT/2000 computer, the Oracle Universal Installer does this when the Oracle software is installed. Unless you are running multiple versions of Oracle on the same computer, the configuration arrived at by Oracle Universal Installer will properly prepare the operating system environment for the creation of a database. However, on a UNIX system, your session may not have all the required environment variables set, so you will need to ensure that they have been passed on to you in your startup script, or you will need to configure them before you create the database.

Table 4-2 lists the operating system environment variables that are required or recommended to be set for a successful database creation.

Table 4-2
Environment Variables Required for Database Creation

<i>Variable</i>	<i>Description</i>
ORACLE_HOME	This variable must point to the location of the Oracle executable (binary) files, for example “/usr/app/oracle/product/8.1.7” on a UNIX system. This allows Oracle to properly locate the files needed for database creation. This variable is required.
ORACLE_SID	This variable specifies the name of the instance and database to be created. It must be set to properly start the instance and must be unique on the machine. This variable is required.
ORA_NLS33	If you are creating a database with a character set and National Language character set other than US7ASCII, you will need to set this environment variable so that Oracle can locate the character set specifications during the creation process. This variable is required in all cases except when creating a database to use the US7ASCII character set. It typically should point to the “ocommon/nls/admin/data” directory under ORACLE_HOME, the default installation location of the NLS files.
ORACLE_BASE	This variable is not required, but should be set if you are performing an OFA-compliant installation. It typically points to the base directory for all Oracle installations, for example, /usr/app/oracle.
PATH	The search path must include the Oracle “bin” directory under ORACLE_HOME where all the executable files are stored. On a UNIX system ensure that you modify the contents of the PATH environment variable to include it—this was already done on Windows NT/2000 when the Oracle software was installed.

To set these environment variables on a UNIX system running the Korn shell, issue command similar to the following:

```
$ ORACLE_SID=CERTDB; export ORACLE_SID
$ ORACLE_BASE=/usr/app/oracle; export ORACLE_BASE
$ ORACLE_HOME=/usr/app/oracle/product/8.1.7; export ORACLE_HOME
reverse order, most DBA's use
ORACLE_HOME=$ORACLE_BASE/product/8.1.7; export ORACLE_HOME
$ ORA_NLS33=$ORACLE_HOME/ocommon/nls/admin/data; export
ORA_NLS33
$ PATH=$PATH:$ORACLE_HOME/bin; export PATH
```

Please note that UNIX is a case-sensitive environment and the environment variable names **must** be in all uppercase. Ensure that when setting these environment variables you do not use mixed or lower case as subsequent commands will fail.

On a Windows NT/2000 system, you only need to set the ORACLE_SID environment variable for the Command Prompt window you will be using to create the database. To do so, issue a command similar to the following:

```
C:\> set ORACLE_SID=CERTDB
```

The ORADIM utility

One further consideration for Windows NT/2000 environments is the creation of the service for the database. This is done by running the ORADIM utility and passing it a number of parameters. The syntax of the ORADIM utility, as it pertains to the creation of a database, is as follows:

```
oradim -NEW -SID <SID> [-INTPWD <password>]
[-SRVC <servicename>] [-MAXUSERS n]
[-STARTMODE auto|manual] [-PFILE <parfile>]
```

Table 4-3 provides more information on the meaning of each of the parameters for the utility.

Table 4-3
ORADIM Utility Parameters

<i>Parameter</i>	<i>Description</i>
-NEW	Indicates that you are creating a new database and instance.
-SID <SID>	Specifies the name of the SID to create. <SID> cannot be more than 8 characters, must start with a letter, and cannot be the same as another <SID> on the system.
-INTPWD <password>	The password to be used for the INTERNAL privileged account after the service is created. If this parameter is not specified on the command line, the INTERNAL account will automatically be assigned a password of oracle.
-SRVC <servicename>	The name of the Windows NT/2000 service to be created. If this parameter is not specified, the service will automatically be given a name of OracleService<SID>, for example OracleServiceCERTDB.
-MAXUSERS n	The maximum number of users defined in the password file. The default is 5 and limits how many individuals can be granted SYSDBA or SYSOPER privileges.

Continued

Table 4-3 (continued)

<i>Parameter</i>	<i>Description</i>
-STARTMODE auto manual	Specifies the initial startup mode for the service. The choices are auto, where the service will start automatically whenever Windows NT/2000 starts, or manual, where the DBA or system administrator must manually start the service. For production systems, this should be set to "auto" after database creation. The default setting is manual. You can also change the startup mode of the Oracle service at any time by using Control Panel/Services in Windows NT or the Services MMC snap-in in Windows 2000.
-PFILE <parfile>	The full physical path to the Oracle initialization parameter file for the instance. This file must exist. You should always specify the full path and filename so that the service can start properly, as follows: C:\CERTDB\DISK1\INITCERTDB.ORA

To create Windows 2000 services for a database called CERTDB, you may issue a command similar to the following:

```
oradim -new -sid CERTDB -maxusers 30 -pfile c:\certdb\disk1\initcertdb.ora
```

Whenever the ORADIM utility is executed, it records the operations executed in a log file called ORADIM.LOG in the DATABASE directory under ORACLE_HOME. For example, if Oracle8i was installed on your C: drive, the file will typically be c:\oracle\ora81\database\oradim.log.

Preparing the parameter file

Objective

Prepare the parameter file

Before you can create a database, you need to create a parameter file, also called an INIT.ORA file, with the proper configuration parameters to start the instance. In order to create a database, you must start the instance for the database. In order to start the instance, you must have a parameter file with the proper configuration settings.

If you use the Database Configuration Assistant to create your database, the parameter file is created at the same time. In this case, you do not need to do anything. However, if you want to manually create the database using the CREATE DATABASE command, you will need to manually create the parameter file.



Although the Database Configuration Assistant makes life easy when creating a database, you will be tested on how to create a parameter file manually and the different elements that must be included for database creation to succeed.

To create a parameter file, use a text editor (that is, vi or Notepad) to create the file from scratch, or copy an existing parameter file to the name you require for the new database. When creating a parameter file, you **must** use a text editor, as the file must contain only ASCII data.

The parameter file should be named INIT<SID>.ORA, where <SID> is the SID of the instance that will access the database, and should be placed in the ORACLE_HOME/DBS directory after creation. This is because when starting the instance, Oracle will look for a file called INIT<SID>.ORA, and, if it does not find it, will return an error. You can name it something else, but then you will need to refer to it explicitly during instance startup.



On Windows NT/2000 environment, the name and location of the parameter file are tied to the database instance during the database creation process. If you want to move the parameter file after the services are created, you will need to modify the startup settings for the service in the registry.

The parameter file for the database to be created **must** have at least the parameters listed in Table 4-4 specified.

Table 4-4
Required INIT.ORA Parameters

<i>Parameter</i>	<i>Description</i>
DB_NAME	The name of the database to be created. This parameter is typically the same as the value for the environment variable ORACLE_SID and the <SID> portion of the INIT<SID>.ORA filename. Although DB_NAME does not need to match these values, Oracle recommends that it does for simplicity's sake. This parameter must match the database name specified when issuing the CREATE DATABASE command.
CONTROL_FILES	The name and location of the control files for the database. These files do not need to exist and will be created when the CREATE DATABASE command is executed. You should specify at least two control files on separate physical disks. If you're specifying more than one control file, enclose the control files in parentheses and separate them by commas, as shown in the example below.

Continued

Table 4-4 (continued)

<i>Parameter</i>	<i>Description</i>
DB_BLOCK_SIZE	Although this parameter defaults to 2048 and does not need to be specified, you should carefully plan and specify it here. DB_BLOCK_SIZE specified the size of each block in the datafiles and the size of each database buffer cache buffer in the SGA. The value of DB_BLOCK_SIZE cannot be changed except by re-creating the database, so make sure that you specify the correct value for your expected data type.



Optimal values for DB_BLOCK_SIZE will be discussed in Chapter 9.

You can specify other parameters in the parameter file as well, and many more will be discussed in later chapters. Their settings are important in determining the resources that the instance will consume when it is started and will also affect the performance of the database. Later chapters discuss these issues in more detail. Your parameter file may look something like this:

```

db_name = "orcl"
db_domain = mars.bradsys.com
instance_name = ORCL
service_names = orcl.mars.bradsys.com
db_files = 200
db_block_size = 8192

control_files = ("d:\oracle\oradata\orcl\control01.ctl",
"e:\oracle\oradata\orcl\control02.ctl", "f:\oracle\oradata\orcl\control03.ctl")

remote_login_passwordfile = exclusive
os_authent_prefix = ""

open_cursors = 300
max_enabled_roles = 30

db_file_multiblock_read_count = 8 # INITIAL
db_block_buffers = 15602 # INITIAL
shared_pool_size = 35000000
large_pool_size = 614400
java_pool_size = 20971520
log_checkpoint_interval = 10000
log_checkpoint_timeout = 1800

processes = 150 # INITIAL
parallel_max_servers = 5 # SMALL

log_buffer = 32768 # INITIAL

```

```
#audit_trail = true # if you want auditing
#timed_statistics = true # if you want timed statistics
max_dump_file_size = 10240 # limit trace file size to 5MB each

# Uncommenting the line below will cause automatic archiving if archiving has
# been enabled using ALTER DATABASE ARCHIVELOG.
# log_archive_start = true
# log_archive_dest_1 = "location=f:\oracle\oradata\orcl\archive"
# log_archive_format = %%ORACLE_SID%%T%TS%.ARC

rollback_segments = ( RBS0, RBS1, RBS2, RBS3, RBS4, RBS5, RBS6 )

background_dump_dest = f:\oracle\admin\orcl\bdump
user_dump_dest = f:\oracle\admin\orcl\udump

compatible = 8.1.0
sort_area_size = 65536
sort_area_retained_size = 65536
```

Creating a password file

If you are running Windows NT/2000, the ORADIM utility automatically creates a password file in the ORACLE_HOME/DBS directory and makes it a hidden file. In this environment, you don't need to do anything else to create the file. On UNIX systems, you will need to create the password file using the ORAPWD utility. Information on doing this is presented earlier in this chapter, as well as in Chapter 2.

The bottom line is that before you create a database, you need to be able to connect to the instance as a privileged user, using either operating system or password file authentication. If you don't establish proper authentication, you will not be able to issue the CREATE DATABASE command.

Starting the instance of the database to be created

The next step in the creation of a database is to start the instance that will be used to access the database. To do so, you need to be a privileged user either through operating system or password file authentication.

To start the instance, start Server Manager Line Mode and start the instance in a NOMOUNT state. The startup state **must** be NOMOUNT because no datafiles or control files exist to open. If you forget to specify the NOMOUNT state when you issue the STARTUP command in Server Manager Line Mode, Oracle will generate an error but stop at this startup state anyway. To start the instance for database creation, issue the following commands:

```
C:\> svrmgrl

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.
```

```

Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SVRMGR> connect internal/oracle;
Connected.
SVRMGR> startup nomount pfile="c:\certdb\disk1\initcertdb.ora";
ORACLE instance started.
Total System Global Area                198010908 bytes
Fixed Size                               75804 bytes
Variable Size                           70045696 bytes
Database Buffers                        127811584 bytes
Redo Buffers                             77824 bytes
SVRMGR>

```

When issuing the STARTUP command in Server Manager Line Mode, you should specify the PFILE parameter and pass it the full path of the parameter file to ensure that you are starting the proper instance. In Windows NT/2000, this should be the same file as that specified when running ORADIM.

You are now ready to issue the CREATE DATABASE command to create the database.

Issuing the CREATE DATABASE command

With all the preparatory work done, you are ready to issue the CREATE DATABASE command to create your database. The syntax for the CREATE DATABASE command is as follows:

```

CREATE DATABASE [dbname] [CONTROLFILE REUSE]
[DATAFILE filename SIZE n[K|M] [REUSE]
[AUTOEXTEND {OFF|ON [NEXT n[K|M]]
MAXSIZE {UNLIMITED|n[K|M]}]
}], ...]
[LOGFILE [GROUP n](filename, filename,...)SIZE
n[K|M][REUSE]
[GROUP n](filename, filename,...)SIZE n[K|M] [REUSE]...]
[MAXLOGFILES n]
[MAXLOGMEMBERS n]
[MAXLOGHISTORY n]
[MAXDATAFILES n]
[MAXINSTANCES n]
[ARCHIVELOG|NOARCHIVELOG]
[CHARACTER SET charset]
[NATIONAL CHARACTER SET charset]

```

Table 4-5 outlines the meaning of the various parameters for the command.

Table 4-5
CREATE DATABASE Command Parameters

<i>Parameter</i>	<i>Description</i>
<i>Dbname</i>	Specifies the name of the database to be created. If not specified, defaults to the value of the DB_NAME Oracle initialization parameter and must be equal to it. This parameter is specified in the CREATE DATABASE syntax more for readability than any other useful purpose.
CONTROLFILE REUSE	Tells Oracle not to generate an error if it finds a file on disk with the exact same name as that specified in the parameter file for the control file. This is typically specified if you are re-creating an existing database because the previous creation attempt did not complete. If Oracle finds a file on disk, it will abort the creation process unless this parameter is specified.
DATAFILE	<p>The DATAFILE parameter indicates the physical name and size as well as AUTOEXTEND characteristics of the datafiles for the SYSTEM tablespace. More than one datafile can be specified, but the CREATE DATABASE statement will associate all datafiles specified with the SYSTEM tablespace only. You need to provide the full physical path for the datafiles as well as the size. The size can be specified in bytes, kilobytes (K), or megabytes (M). If the file already exists on disk from a previous database creation attempt, you need to also specify the REUSE parameter to tell Oracle to overwrite the file.</p> <p>There is no default for the DATAFILE parameter and both the filename and size need to be specified. The path to the datafile must already exist for the creation to succeed.</p> <p>For each datafile, you may also specify AUTOEXTEND characteristics, that is, what should happen if the datafile runs out of disk space. The default is AUTOEXTEND OFF, but you may turn it on and tell Oracle to enlarge the file each time more room is needed by the value of NEXT, up to a maximum value of MAXSIZE. If you specify a NEXT incremental value but do not specify MAXSIZE, MAXSIZE defaults to UNLIMITED.</p>
LOGFILE GROUP <i>n</i>	Instructs Oracle to create a redo log file group with the associated members specified. You must specify at least two redo log file groups with one member each when creating a database. Additional redo log file groups and members can be added after the database is created. Each redo log file group should be the same size. All members of a redo log file group will always be the same size as that which the SIZE parameter applies to the group as a whole.

Continued

Table 4-5 (continued)

<i>Parameter</i>	<i>Description</i>
	If the files already exist on disk from a previous attempt to create the database, you can specify the REUSE parameter to overwrite the existing files.
MAXLOGFILES <i>n</i>	Instructs Oracle to create <i>n</i> slots in the control file to hold information about the redo log file groups. MAXLOGFILES specifies what the maximum number of redo log file groups that can be supported by this database should be set to. If you need to change this value, you must re-create the control files.
MAXLOGMEMBERS <i>n</i>	Instructs Oracle to allocate space in the control files for <i>n</i> members of each redo log file group. MAXLOGMEMBERS specifies what the maximum number of members supported for each redo log file can be for this incarnation of the database. If you want to change this value, you must re-create the control files.
MAXLOGHISTORY <i>n</i>	For an Oracle Parallel Server environment, specifies the maximum number of archived redo log files to be tracked in the control files for automatic media recovery. If you wanted to change this value, you would need to recreate the control files.
MAXDATAFILES <i>n</i>	Instructs Oracle to allocate <i>n</i> slots for datafile information in the control files. Essentially, this sets the maximum number of datafiles that are supported by this database, unless the control file is re-created to increase this value.
MAXINSTANCES <i>n</i>	Specifies the maximum number of Oracle instances that can simultaneously mount and open this database. The default for MAXINSTANCES is 1 and should be increased only if you plan to use Oracle Parallel Server.
ARCHIVELOG	Turns on ARCHIVELOG mode for the database immediately after creation. In ARCHIVELOG mode, Oracle makes a copy of each redo log file as it becomes full and does not reuse the redo log file unless it is archived. In order for your database to operate properly, if you decide to enable ARCHIVELOG mode, you should also configure the corresponding INIT.ORA parameters of LOG_ARCHIVE_START, LOG_ARCHIVE_DEST_ <i>n</i> , and LOG_ARCHIVE_FORMAT. Failure to do so may cause the database to stop when redo log files become full. It is generally not recommended that ARCHIVELOG mode be turned on at database creation time, but rather after it has been populated and prior to it becoming operational.
NOARCHIVELOG	Instructs Oracle to place the database into NOARCHIVELOG mode, which allows redo log files to be overwritten as they are needed. This is the default.

<i>Parameter</i>	<i>Description</i>
CHARACTER SET	Specifies the character set to be used for this database when storing data in CHAR, VARCHAR2, LONG, and CLOB columns. If not specified, this parameter defaults to US7ASCII, a 7-bit character set that supports the first 128 characters of the ASCII table. If you specify a character set other than the default, ensure that the ORA_NLS33 environment variable has been properly set.
NATIONAL CHARACTER SET	Specifies the National Language character set used to store data in NCHAR, NVARCHAR2, and NCLOB columns. If not specified, this also defaults to US7ASCII.

**Tip**

The CREATE DATABASE syntax is long, with many parameters. Rather than type it interactively in server manager, Oracle recommends that you create it using a text editor and then run it as a SQL script in Server Manager Line Mode.

After deciding on the various options for the CREATE DATABASE command, you can issue the command in Server Manager Line Mode, as in the following example:

```
SVRMGR> CREATE DATABASE CERTDB
 2> LOGFILE 'C:\CERTDB\DISK3\redo01a.log' SIZE 1024K,
 3>      'C:\CERTDB\DISK3\redo02a.log' SIZE 1024K,
 4>      'C:\CERTDB\DISK3\redo03a.log' SIZE 1024K
 5> MAXLOGFILES 32
 6> MAXLOGMEMBERS 4
 7> MAXLOGHISTORY 1
 8> DATAFILE 'C:\CERTDB\DISK1\system01.dbf' REUSE
 9>      SIZE 75M AUTOEXTEND ON NEXT 10240K
10> MAXDATAFILES 254
11> MAXINSTANCES 1
12> CHARACTER SET WE8ISO8859P1
13> NATIONAL CHARACTER SET WE8ISO8859P1;
Statement processed.
```

**Tip**

You may be interested to know that Oracle actually runs a script during the database creation process. The script that Oracle runs when you issue the CREATE DATABASE command is called SQL.BSQ and is located in the ORACLE_HOME\RDBMS\ADMIN directory.

**Caution**

Never modify the SQL.BSQ script file. Doing so will cause problems in later database creation attempts. Should the file become damaged, you may copy it from the CD or another installation of Oracle. The script is the same on all platforms running the same version of Oracle, such as 8.1.7 or 8.1.6.

You have now created the database at its most basic level. However, you cannot use the database in its present condition. You still need to create the data dictionary and the Oracle standard packages and procedures. This is discussed in Chapter 5.

Post-creation status

After the database is created, Oracle mounts and opens the database. The newly created database consists of the following elements:

- ♦ The SYSTEM tablespace and the datafiles that make it up. As mentioned earlier, creating the database with the CREATE DATABASE command creates only the SYSTEM tablespace. All other tablespaces need to be created manually.
- ♦ The control files in the location specified in the parameter file. Control files will be sized according to the parameters specified during database creation, including MAXLOGFILES, MAXLOGMEMBERS, MAXDATAFILES, MAXLOGHISTORY, and MAXINSTANCES.
- ♦ The online redo log files with members in the location specified in the CREATE DATABASE command. Each member of the same group will be the same size.
- ♦ Two Oracle users — SYS and SYSTEM. SYS — will be created with a default password of CHANGE_ON_INSTALL, and SYSTEM will be created with a default password of MANAGER. Oracle recommends, after you create the data dictionary views as outlined in Chapter 5, you change the password for these users using the ALTER USER command.
- ♦ A SYSTEM rollback segment located on the SYSTEM tablespace.
- ♦ Internal data dictionary tables. At this point no DBA_, ALL_, or USER_ data dictionary views exist in the database, but the base data dictionary tables are not created as of yet.
- ♦ Dynamic performance views. The V\$ dynamic performance views, such as V\$CONTROLFILE, V\$DATAFILE, V\$PARAMETER, and others can be queried at this point.

You may wish to query the dynamic performance views to ensure that things were created as expected. For example, to verify the number of redo log file groups and the location of your redo log members, you can issue the following commands:

```
SELECT * FROM V$LOG
SELECT * FROM V$LOGFILE
```

Troubleshooting Database Creation Problems

Although database creation is a smooth process in most cases, you may encounter some problems during the process. Some of the more common ones include:

- ♦ Syntax errors in the SQL script — Many of us may faithfully type the syntax of the CREATE DATABASE command in our scripts and be absolutely certain that it is correct, and then leave a stray comma or some other typo in the mix. These types of errors are sometimes referred to as “PIBCAS” or “Problem is between chair and system” errors. If one is found in your CREATE DATABASE

script, the creation process will fail. This is why it is a good idea to spool the output of the CREATE DATABASE command to a file by adding a line similar to the following prior to the CREATE DATABASE command:

```
spool credb.log
```

- ♦ File to be created already exists on the hard drive — This is a common problem when you are issuing the CREATE DATABASE command again because a previous attempt failed. Before doing so, ensure that all files are removed from the hard drive (remember to SHUTDOWN the instance first) by scanning the directories referenced in your CREATE DATABASE statement and deleting any redo log, data, or control files found.
- ♦ The path where database files need to be does not exist — If you did not create the directories referenced by your CREATE DATABASE command or where your control files are to be located, the process will fail. Ensure that all directories have been created before you execute the command.
- ♦ Permission problems — For the CREATE DATABASE to succeed, the Oracle software needs to have proper permissions to create files in all the directories you are referencing. Ensure that the owner of the Oracle software has these privileges. This is more of an issue with UNIX installations than those on Windows NT/2000 because the Oracle Universal Installer ensures that the Oracle software has Administrator privileges on the Windows NT/2000 computer.
- ♦ Insufficient space on the hard drive — Before creating a database, make sure you have sufficient free space to create data and redo log files of the sizes mentioned in your CREATE DATABASE statement. To correct the problem of insufficient space, move existing files from the target hard drive to another disk or move the location of the files to be created to a different location in the script. This problem may also arise if you have a disk quota configured on the hard disks. In the latter case, increasing the quota should solve the problem.

If any of these situations occurred when you executed your CREATE DATABASE statement, remove all files that may have been generated by the statement, SHUTDOWN the instance, and repeat the process.

Key Point Summary

- ♦ You can use the Database Configuration Assistant or the CREATE DATABASE command to create a database. The Database Configuration Assistant will create all the necessary directories and files according to the parameters you outline.
- ♦ Database Configuration Assistant will place files in locations that conform to the Optimal Flexible Architecture and will configure Oracle parameter file (INIT.ORA) parameters based upon the characteristics you specify and the resources available on the computer where it is run. You should review these settings to ensure that they are adequate for your requirements.

- ♦ When creating a database with the Database Configuration Assistant, you should let it create the scripts that you will later run to perform the database creation. This allows you to verify your configuration and make changes if necessary. It also makes it easier to repeat the database creation process should an error occur.
- ♦ The Optimal Flexible Architecture was designed to organize the file system to minimize contention, make adding files to the database easy, and create a directory structure that is easy to maintain. It is one solution that you can use to solve the problem of arranging Oracle database files for best performance.
- ♦ Planning your database file structure is a critical step to be performed prior to the creation of a new Oracle database. Before deciding on where to place files, you should have a good understanding of how the data will be accessed.
- ♦ When creating a database you should create at least two control files on separate physical disks.
- ♦ Only privileged users who have been granted the SYSDBA or OSDBA role have permission to create the database. Authentication of privileged users can take place through the operating system or by using a password file.
- ♦ When creating a database, you must create at least two redo log file groups — three is preferred. Each group should have at least two members, each on separate physical disks.
- ♦ If you want to use the CREATE DATABASE command, create a script to be run instead of typing in the command interactively.
- ♦ Before creating the database with the CREATE DATABASE command, ensure that all directories specified in your script already exist. If not, the creation process will fail.
- ♦ The parameter file for a new database must contain at least the DB_NAME and CONTROL_FILES parameters with the correct values. You should also specify a value for the DB_BLOCK_SIZE parameter if the default of 2048 does not meet your database characteristics.
- ♦ When using the CREATE DATABASE command, you must prepare the operating system by initializing the ORACLE_HOME, ORACLE_SID, ORA_NLS33, and PATH environment variables. This is required on UNIX systems, whereas only ORACLE_SID is required on Windows NT/2000 because the others are configured by the Universal Installer when the Oracle software is installed.
- ♦ On Windows NT/2000, the ORADIM utility must be run to create the service for the instance.
- ♦ To create a database, you must start the instance to a NOMOUNT state.
- ♦ Decide on an appropriate character set and National Language character set before creating the database because you can't change these character sets except by re-creating the database after exporting all the data. Do not use the default US7ASCII character set.

- ♦ After creation of your database, the data dictionary base tables are created and you can query the V\$ dynamic performance views. The database is not yet fully configured and additional scripts need to be run.
- ♦ After database creation, two users will exist in the database: SYSTEM with a password of MANAGER and SYS with a password of CHANGE_ON_INSTALL. You should change the passwords for these users by using the ALTER USER command.
- ♦ If your database creation fails for any reason, remove any files that were initialized, correct the problem, and repeat the process.



STUDY GUIDE

This portion of the chapter tests your understanding of the material covered by having you go through some assessment questions that are similar in style to those you will be asked on the Oracle exam. This section is followed first by scenarios for creating a database and then lab exercises that have you create a database to use for the remainder of the book. Finally, at the end of the chapter you can find answers and explanations for the chapter pre-test questions, assessment questions, scenarios, and the lab exercises.

Assessment Questions

1. You have received a script that is used to create a database for your newly purchased customer management system. When you execute the script, it generates an error. Which of the following lines from the script is causing the error? (Choose the best answer.)
 - A. `CREATE DATABASE CRMDATA`
 - B. `LOGFILE '/CRMDATA/redo01a.log' SIZE 1024K,
'/CRMDATA/redo02a.log' SIZE 1024K,
'/CRMDATA/redo03a.log' SIZE 1024K`
 - C. `DATAFILE '/CRMDATA/system01.dbf' SIZE 75M
AUTOEXTEND OFF NEXT 10240K`
 - D. `CHARACTER SET WE8ISO8859P1`
 - E. `NATIONAL CHARACTER SET WE8ISO8859P1;`
2. Which of the following is the Oracle Optimal Flexible Architecture not specifically designed to address? (Choose the best answer.)
 - A. Ease of administration
 - B. Improving performance of the database
 - C. Hosting multiple databases on the same server
 - D. Location of files to minimize IO contention
 - E. Providing a structure to easily locate files
3. After creating a database, which of the following views can be used to get information about different files that make up the database? (Choose two correct answers.)
 - A. `V$DATAFILE`
 - B. `DBA_DATA_FILES`

- C. DBA_REDO_LOGS
 - D. V\$REDO_LOGFILE
 - E. V\$LOGFILE
4. You need to create a database on a UNIX-based system that will use a Japanese National Language character set. When creating the database manually using the CREATE DATABASE command, which environment variables must be initialized for the process to work correctly? (Choose all correct answers.)
- A. ORACLE_HOME
 - B. ORACLE_BASE
 - C. ORACLE_NLS
 - D. ORA_NLS33
 - E. ORACLE_DB
 - F. ORACLE_SID
5. What units can be used to specify the size of the redo logs when issuing the CREATE DATABASE statement? (Choose all correct answers.)
- A. Gigabytes
 - B. Bytes
 - C. Megabytes
 - D. Kilobytes
 - E. Database blocks
 - F. Operating system blocks
6. While issuing the CREATE DATABASE command, after starting the instance to a NOMOUNT state, you receive the following error messages:

```
ORA-01501: CREATE DATABASE failed
ORA-00200: controlfile could not be created
ORA-00202: controlfile: 'C:\CERTDB\DISK1\control01.ctl'
ORA-27038: skgfrcre: file exists
OSD-04010: <create> option specified, file already exists
```

What are possible reasons for the error? (Choose all correct answers.)

- A. The database to be created already exists.
- B. You do not have permissions to create the control file in the specified location.
- C. A previous attempt to create the database failed and was not cleaned up properly.
- D. You do not have SYSOPER privileges on the database.
- E. You do not have SYSDBA privileges on the database.

7. What roles must the individual who will be creating an Oracle database be granted on the database in order to perform the task? (Choose two correct answers.)
- A. DBA
 - B. SYSDBA
 - C. SYSOPER
 - D. OSDBA
 - E. SQLDBA
 - F. OSOPER
8. After a CREATE DATABASE command is successfully executed, what user/password combinations exist in the database? (Choose two correct answers.)
- A. SCOTT/MANAGER
 - B. SYSTEM/CHANGE_ON_INSTALL
 - C. SYS/CHANGE_ON_INSTALL
 - E. SCOTT/TIGER
 - F. SYSTEM/MANAGER
 - G. SYS/MANAGER
9. Which script file is run by Oracle when the CREATE DATABASE command is executed? (Choose the best answer.)
- A. CREATEDB.SQL
 - B. SQL.SQL
 - C. SQL.BSQ
 - D. CATALOG.SQL
 - E. CATPROC.SQL
10. You have received a script that is used to create a database for your newly purchased customer-management system. When you execute the script, it generates an error. Which of the following lines from the script is causing the error? (Choose the best answer.)
- A. CREATE DATABASE CRMDATA
 - B. LOGFILE '/CRMDATA/redo01a.log' SIZE 1024K,
 - C. DATAFILE '/CRMDATA/system01.dbf' SIZE 75M
 - D. CHARACTER SET US7ASCII
 - E. NATIONAL CHARACTER SET WE8ISO8859P1;

Scenarios

1. You have just been hired as the database administrator for a Web-based solutions company. Part of your responsibility is to create new databases and ensure that they are properly configured. The IS Manager has set a policy that all database creation and modification must be done using scripts so that if a problem occurs, the steps can be repeated. You need to create a database with three tablespaces (SYSTEM, DATA, and INDEX) that will be used for decision support. The database should take advantage of all the hardware available on the server to minimize contention and properly configure the instance.

You are asked to put together a script that would perform the necessary tasks. The database will run on a Linux server. What is the best way to create the script?

2. Your large Fortune 200 company has decided to decentralize some of its IS operations. Users have frequently been asking for the ability to create their own databases for departmental applications that will be used only inside the company. Recently, your junior DBAs have been reporting that a number of support calls have been received from departments regarding their new-found authority to create databases. The databases are created properly but they cannot be accessed except from the computer on which they are created. What is the cause of the problem and how can it be rectified?

Lab Exercises

Lab 4-1 Creating an Oracle Database

1. On your computer, create a directory called CERTDB off the root of your C drive (on Windows NT/2000) or off the root on UNIX.
2. In the CERTDB directory, create these additional directories: DISK1, DISK2, DISK3, DISK4, DISK5, and DISK6. These will be used to represent physical disks that your database will use to place files into.
3. Create a database called CERTDB, with the appropriate parameter file, on your computer with the following characteristics:
 - A SYSTEM tablespace with a size of 50MB located on DISK1 (that is, C:\CERTDB\DISK1) with a filename of system01.dbf. The datafile should not grow automatically if it becomes full.
 - Three redo log file groups with one member each. The redo log member files should be called redo01a.log, redo02a.log, and redo03a.log for groups 1, 2, and 3, respectively. The files should be located in DISK3 and all redo log groups should be 1MB (that is, 1,024KB) in size.
 - Two control files called control01.con and control02.con located on DISK1 and DISK2, respectively.

- Database block size of 8,192 bytes. You should also allocate 200 buffers for the database buffer cache.
- Shared pool size of 16,384,000 bytes.
- Redo log buffers size of 65,536 bytes.
- Do not select any Oracle options for the database except the Partitioning option and SQL*Plus Help.
- The password for the INTERNAL user should be oracle.
- The parameter file for the database should be located on DISK1.

Answers to Chapter Questions

Chapter Pre-Test

1. Before you can create a database using the CREATE DATABASE command, you need to create the parameter (INIT.ORA) file. The filename should be INIT<sid>.ORA and must contain at least the DB_NAME, CONTROL_FILE, and DB_BLOCK_SIZE parameters.
2. The CREATE DATABASE statement requires that you specify the name, size, and location of the data file that will make up the system tablespace. You also need to specify the location of at least two redo log file groups and their members. Other elements of the CREATE DATABASE command, including the character set and National Language character set, are optional.
3. To issue the CREATE DATABASE command, you must have either the SYSDBA or OSDBA privilege, through password file or O/S authentication, respectively.
4. In a Windows NT/2000 environment, you need to create the services for the instance. To do this, you need to run the ORADIM utility with two parameters specified: the -NEW and -SID parameters.
5. To issue the CREATE DATABASE command, the instance must be started in NOMOUNT mode. Any attempt to start the instance in any other mode will fail because the control files are not created until you issue the CREATE DATABASE command. To proceed to the MOUNT state, Oracle will need to open the control files.
6. Optimal Flexible Architecture is designed to facilitate a structure on your server that makes administering and facilitating the growth of databases easy. It includes provisions for hosting multiple databases on the same server, locating the datafiles for each database easily, minimizing IO contention by properly placing files on disks in such a way that files with high probability of contention are physically separated. The OFA also takes into account the addition of drives to more smoothly even out the IO on the system.

7. You must create at least two redo log file groups with at least one member in each group. This is because redo log files operate in a circular fashion so that one group of files is always in use and the other will take over when the first becomes full. This allows Oracle to delay committing changes to the datafiles without losing data—the changes are recorded in the redo log files.

You should create at least three redo log file groups with two members each, with each member of the same group on a separate physical disk. Doing so will allow you to continue to work even if a log file member is lost because of disk failure. All log file groups should be sized identically.

8. The control file is created when you issue the CREATE DATABASE command. The name and location of the control files to be created is read from the parameter file when the instance is mounted. In this way, Oracle knows how many to create and where to create them when you issue the CREATE DATABASE command.
9. You can create a database in Oracle using the CREATE DATABASE command or by using the Database Configuration Assistant. The CREATE DATABASE command requires that you do all post- and preconfiguration tasks manually, whereas the Database Configuration Assistant does all of them for you, as well as allowing you to save your choices as scripts for later creation of the database. The preferred method is to use the Database Configuration Assistant to create the scripts and then make any changes before executing them.
10. The size of the database block in the datafiles is determined by the value of the DB_BLOCK_SIZE initialization parameter. The default on most systems is 2048 bytes. To change the database block size, you will need to export all your data from the database and then drop and re-create it. You can't change the database block size after database creation except by re-creating the database.

Assessment Questions

1. **C.** The line that specifies the physical location and size of the data file is causing the error because it has AUTOEXTEND OFF while simultaneously specifying a value for NEXT.
2. **B.** Optimal Flexible Architecture is specifically designed to improve the performance of a database as there are many variables involved that could have a negative impact on performance (CPU, data access patterns, and so on). However, following OFA guidelines will, in many cases, actually provide better performance than might otherwise be realized. OFA was designed to ease administration and make it easier to host multiple databases on the same machine by providing an easy-to-adhere to file structure, which also minimizes IO contention by locating files on several disks, if available.
3. **A, E.** You can use V\$DATAFILE to get a list of datafiles that were initially created and use V\$LOGFILE to get a listing and location of the redo log file members. No DBA_ views are created when you execute the CREATE DATABASE command, so answers B and C are wrong. There is no view called V\$REDO_LOGFILE, so answer D is also incorrect.

4. **A, D, F.** To create a database manually by executing the `CREATE DATABASE` command, you need to ensure that the `ORACLE_SID` environment variable has been initialized with the instance name, the `ORACLE_HOME` environment variable points to the location of the Oracle software installation, and the `ORA_NLS33` environment variable points to the location of the National Language support files. This last environment variable is required in this case because you need to create a database to support Japanese language characters. Although Oracle recommends doing so, initializing the `ORACLE_BASE` environment variable is not necessary. Oracle does not use `ORACLE_NLS`.
5. **B, C, D.** You can specify the size of the redo log files (as well as datafiles) in bytes, kilobytes (K), or megabytes (M).
6. **A, C.** The error message received indicates that the control file already exists on the hard disk and Oracle will not overwrite the file unless you specify the `REUSE` parameter (which was not done here). If the file exists, the most likely reasons are that the database has already been successfully created and the file should be there, or that a previous creation attempt failed and the file was not manually removed, as it should have been. A third option, not given as a possible answer, is that a file with the same name as the control file exists in the location but is not part of any database. The bottom line in all the scenarios is that the file cannot be overwritten, so Oracle returns an error and aborts the database creation process.
7. **B, D.** To create a database, the user will need to be authenticated as a privileged user who has the `SYSDBA` or `OSDBA` role. Although `SYSOPER` and `OSOPER` will allow a user to start the instance, only those individuals with the `SYSDBA` or `OSDBA` role can execute the `CREATE DATABASE` command. The `DBA` role is used to administer a database after it is created. There is no `SQLDBA` role in Oracle by default.
8. **C, F.** After the `CREATE DATABASE` command is executed, a user `SYS` with a password of `CHANGE_ON_INSTALL` and a user `SYSTEM` with a password of `MANAGER` are created.
9. **B.** Oracle's `CREATE DATABASE` statement runs the script `SQL.BSQ` located in the `ORACLE_HOME/rdbms/admin` directory. `CATALOG.SQL` and `CATPROC.SQL` are also valid scripts but they need to be run manually to create the standard data dictionary views and packages.
10. **B.** This line will cause an error because it specifies only a single redo log file group for the database. Oracle requires that at least two log file groups with one member each be specified with creating a database.

Scenarios

1. You could create this database by creating scripts to accomplish the task and manually configuring the Oracle initialization parameter file based upon what you determine are optimal settings. Nothing is wrong with doing so and it will work. However, one of the requirements was that resources on the server be properly allocated to satisfy the requirements of the target database.

The Oracle Database Configuration Assistant uses Optimal Flexible Architecture to determine the best location of files so that contention is minimized. Furthermore, it evaluates the resources available and the requirements outlined for the database to configure the parameter in the INIT.ORA file to take advantage of what can be used. For this reason, the optimal solution is to use the Database Configuration Assistant to provide the ideal configuration settings and then to save the results in a script that can be executed later.

2. The most likely reason that users cannot access the databases they created from remote computers is that they have not properly configured Oracle's networking component Net8. If they used the Database Configuration Assistant, the LISTENER.ORA and the TNSNAMES.ORA files were updated on the machine where the database was created, but these changes were not propagated to other client computers. The result is that no one else can resolve the names of the new databases except for those individuals at the machine on which they are resident.



Although the Oracle8i Architecture and Administration exam will not test you on the specifics of Net8, you may be asked general questions, such as the one presented in this scenario, to show that you understand generally what pieces make a client and a server communicate.

Lab Exercises

Lab 4-1

1. For a Windows NT/2000 system, open a Command Prompt window and issue the following commands:

```
c:  
cd \  
mkdir CERTDB
```

For a UNIX-based system, open a UNIX shell window and issue the following commands:

```
mkdir /CERTDB
```

2. For a Windows NT/2000 system, open a Command Prompt window and issue the following commands:

```
c:  
cd \CERTDB  
mkdir DISK1  
mkdir DISK2  
mkdir DISK3  
mkdir DISK4  
mkdir DISK5  
mkdir DISK6
```

For a UNIX-based system, open a UNIX shell window and issue the following commands:

```
cd /CERTDB
mkdir DISK1
mkdir DISK2
mkdir DISK3
mkdir DISK4
mkdir DISK5
mkdir DISK6
```

- 3. a.** The creation of the database will require that you create and configure a parameter file called `initCERTDB.ora`. The parameter file will look something like this (additional parameters may exist if copied from another database or created by the Database Configuration Assistant):

```
DB_NAME=CERTDB
DB_BLOCK_SIZE=8192
DB_BLOCK_BUFFERS=200
LOG_BUFFER=65536
CONTROL_FILES=(C:\CERTDB\DISK1\control01.con,
                C:\CERTDB\DISK2\control02.con)
#

#If you are using a UNIX-based system, uncomment the
#CONTROL_FILES= lines below and comment out the
#same lines above.
#
CONTROL_FILES=(/CERTDB/DISK1/control01.con,
                /CERTDB/DISK2/control02.con)
SHARED_POOL_SIZE=16384000
```

- 3. b.** To prepare the environment to create the database manually for a Windows NT/2000 system, open a Command Prompt, and issue the following commands (NOTE: The ORADIM command-line parameters span two lines for readability):

```
set ORACLE_SID=CERTDB
oradim -new -sid CERTDB -intpwd oracle
      -startmode manual -pfile "C:\CERTDB\DISK1\initCERTDB.ora"
```

To prepare the environment to create the database manually for a UNIX-based system using a Korn or Bourne shell, open a shell window and issue the following commands:

```
ORACLE_SID=CERTDB; export ORACLE_SID
ORACLE_HOME=<path to Oracle binaries>; export ORACLE_HOME
ORA_NLS33=$ORACLE_HOME/ocommon/nls/admin/data
PATH=$PATH:$ORACLE_HOME/bin
```

- 3. c.** To create the database manually for a Windows NT/2000 system, invoke Server Manager Line mode (svrmgrl) from a command prompt and issue the following commands:

```
spool createCERDB.log
connect internal/oracle;
startup nomount pfile=C:\CERTDB\DISK1\initCERTDB.ora;
CREATE DATABASE CERTDB
LOGFILE 'C:\CERTDB\DISK3\redo01a.log' SIZE 1024K,
        'C:\CERTDB\DISK3\redo02a.log' SIZE 1024K,
        'C:\CERTDB\DISK3\redo03a.log' SIZE 1024K
MAXLOGFILES 32
MAXLOGMEMBERS 4
MAXLOGHISTORY 1
DATAFILE 'C:\CERTDB\DISK1\system01.dbf' SIZE 50M REUSE
        AUTOEXTEND OFF
MAXDATAFILES 254
MAXINSTANCES 1
CHARACTER SET WE8ISO8859P1
NATIONAL CHARACTER SET WE8ISO8859P1;
spool off
```

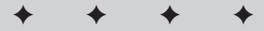
- To create the database manually for a UNIX system, invoke Server Manager Line mode (svrmgrl) from a command prompt and issue the following commands:

```
spool createCERDB.log
connect internal/oracle;
startup nomount pfile=/CERTDB/DISK1/initCERTDB.ora;
CREATE DATABASE CERTDB
LOGFILE '/CERTDB/DISK3/redo01a.log' SIZE 1024K,
        '/CERTDB/DISK3/redo02a.log' SIZE 1024K,
        '/CERTDB/DISK3/redo03a.log' SIZE 1024K
MAXLOGFILES 32
MAXLOGMEMBERS 4
MAXLOGHISTORY 1
DATAFILE '/CERTDB/DISK1/system01.dbf' SIZE 50M REUSE
        AUTOEXTEND OFF
MAXDATAFILES 254
MAXINSTANCES 1
CHARACTER SET WE8ISO8859P1
NATIONAL CHARACTER SET WE8ISO8859P1;
spool off
```


Creating Data Dictionary Views and Standard Packages

EXAM OBJECTIVES

- ◆ Creating Data Dictionary Views and Standard Packages
 - Construct the data dictionary views
 - Query the data dictionary
 - Prepare the PL/SQL environment using the administrative scripts
 - Administer stored procedures and packages
 - List the types of database event triggers



CHAPTER PRE-TEST

1. Which script creates the DBA_ views in your database?
2. Is PL/SQL support available in a database after the CREATE DATABASE command has been issued?
3. What is a package and what are its components?
4. Provide examples of PL/SQL program units that are available in Oracle8i.
5. Which programming languages can be used to create a stored procedure in Oracle?
6. What are triggers and when are they invoked?
7. What is the difference between a function and a stored procedure?
8. What are the four types of scripts that can be used to extend the capabilities of your database?
9. How can you manipulate columns of the BLOB datatype in Oracle8i?
10. What are two ways to change your NLS territory setting for your session in Oracle8i?

In the previous chapter you issued the `CREATE DATABASE` command to create the basic structure of a database in Oracle. At that point, some raw objects were created in the database's `SYSTEM` tablespace, and the `V$` dynamic performance views could be queried. However, most of the functionality that you would expect from Oracle, such as support for PL/SQL in the database, the ability to query the `DBA_`, `ALL_`, and `USER_` data dictionary views, as well as other optional capabilities such as Java support and replication, is not yet available. In order to have Oracle provide these capabilities you need to run certain scripts that create the necessary views and objects to provide it. This chapter will show you how to create the data dictionary views that you will use as a DBA, as well as explain the standard Oracle packages that you will need to create.

Overview of Data Dictionary Views, Standard Packages, and Database Event Triggers

As you saw in Chapter 4, when the `CREATE DATABASE` command is issued, Oracle creates the control files, data files, redo log files, and runs a script called `SQL.BSQ` to create the core data dictionary structure. However, in its basic state, the database really cannot be used for too many things and is like the foundation for a house—just a big hole in the ground with some key properties. In order to make the house complete, you need to put up the interior and exterior walls, plumbing, wiring, and a roof. The creation of the data dictionary views and standard packages is like putting those things in place for a house. Just as every house can look different outside and have a different internal layout, Oracle allows you to use various scripts to install features that you may require to make your database have different characteristics from another.

The Oracle data dictionary consists of a number of elements, once populated. These include:

- ♦ **Base tables**— These are the fundamental core tables that store information about what is in the Oracle database. The data dictionary views and standard packages use the base tables to store their information. These tables are created when the `CREATE DATABASE` command is issued and the `SQL.BSQ` script is run.
- ♦ **Data dictionary views**— These views are created by running additional scripts and are based on the base tables. They provide the capability to query the data dictionary in a more logical format and have more meaningful column names. These views are prefixed with `DBA_`, `ALL_`, or `USER_`.
- ♦ **Dynamic performance views**— Dynamic performance views are created when the `CREATE DATABASE` command is issued, with others being created by running scripts to add functionality to the database. These views start with `V$` and enable the DBA to monitor the performance of the database and get additional information on the database status.

- ♦ **PL/SQL packages, procedures, and triggers**— These PL/SQL program units further extend the database and provide most of the advanced functionality of the database such as allowing you to configure replication or distributed database, and install and configure Java support. They also provide some of the basic functionality required to make the database useful, such as syntax for the ALTER SESSION command that can be used in PL/SQL procedures through the DBMS_SESSION package and its procedures.

The data dictionary is central and critical to the operation of an Oracle database. It provides the information that tells users what is available for manipulation and access. It separates the objects created by one user from objects created by another and ensures that both exist. The data dictionary also allows the Oracle engine to enforce the rules necessary for it to be an RDBMS or ORDBMS. When a query is executed, the Server Process queries the data dictionary about the objects being referenced, as well as the permissions that the user performing the query has on the objects. It ensures that integrity constraints can be easily found and enforced by the engine and much more.

The data dictionary contains a great many things, including information on:

- ♦ **Database structure**— Both the physical (that is, the files that make up the database) and the logical (that is, tablespaces) structure of the database are stored in the data dictionary. While much of this information is also in the control files, the data dictionary is the primary location for logical file structure information, while the control file deals primarily with the physical structure.
- ♦ **Object definitions**— The data dictionary stores information about all database objects, whether or not they require storage space. For example, views do not— unless they are materialized views— require storage for their data, while tables do require storage and are therefore also referred to as *segments*.



Segments and the allocation of database storage to them are covered in detail in Chapter 9.

- ♦ **Segments and their space allocation**— The data dictionary also keeps track of the blocks in the data files that each segment (that is, table, index, cluster, and so on) uses and the total amount of space used. This is to enable SMON to coalesce free space in a tablespace as the space is released by a table being dropped or truncated. It is also used to be able to determine where to allocate additional space for a segment if it is required. The data dictionary also tracks a user's segment space utilization against the quota a user has been given on a tablespace.
- ♦ **Integrity constraints**— When you define an integrity constraint on a table or a view, information about the constraint and its associated rules is stored in the data dictionary and used to enforce the constraint.
- ♦ **Users, roles, and privileges**— In order for individuals to have access to the database, they must have an Oracle user account. The data dictionary stores information on each user created, their quotas, their password management settings, their privileges, roles they have been granted, and so on.



User quotas are discussed in more detail in Chapter 17. Information about users, profiles, and managing permissions can be found in Part V.

- ♦ **Auditing**— Though many people are not aware of this, Oracle does have an auditing facility to track user access to objects and the issuance of certain Oracle commands. Information on what to audit, as well as the audit log table, are both data dictionary components.
- ♦ **Other information**— Oracle also stores other general information in the data dictionary, such as DIRECTORY objects to be used with BFILE data types, National Language Support (NLS) settings, constant definitions such as SYSDATE, and much more.

The bottom line with the data dictionary is simple—it is the one set of objects and information that tells you everything about your database. It is the encyclopedia of your database.

Data dictionary base tables and views

The base tables part of the data dictionary actually stores the data describing the logical and physical structure of the database. Base tables are created when you issue the CREATE DATABASE command. During the creation of the database, one of the tasks that Oracle performs is to run a script called SQL.BSQ. This script is located in the RDBMS/ADMIN directory pointed to by the ORACLE_HOME environment variable. This can be C:\ORA81\RDBMS\ADMIN on a WindowsNT/2000 computer, or /usr/oracle81/rdbms/admin on a UNIX-based computer. The SQL.BSQ script creates a number of objects in the SYSTEM tablespace in the SYS schema. All of the data dictionary objects are owned by the user SYS, and the user SYSTEM is granted full permission to these objects.

Data dictionary base tables typically end or use several \$ in the name, and many base tables names are designed to be difficult to remember. This is because you should not modify the data in the base tables by using standard DML commands but let Oracle modify them when you use DDL statements. Examples of base tables include COL\$ (to store information on columns), IND\$ (to store information about indexes in the database), TAB\$ (containing data about each table in the database), TS\$ (for tablespace information), and many more.

In order to add, remove, or modify the data dictionary, you need to use Data Definition Language (DDL) commands such as CREATE, ALTER, TRUNCATE, or DROP. These commands add, change, or remove entries in the data dictionary to ensure that it correctly reflects the status of your database at all times. Some Data Manipulation Language (DML) commands may also cause the data dictionary to be modified, though the DML command will not operate directly on the data dictionary. For example, adding many rows to a table by performing an INSERT ... SELECT may allocate extents to the table. These new extents and their location will be tracked in the data dictionary. Once the extent is allocated, Oracle automatically adds entries to the data dictionary to track its location and size.

To query the contents of the data dictionary, you normally would not issue SELECT statements against the base tables. This is because the base tables are normalized and would require that you perform several joins to get even simple information, such as the names of columns in a table. For this reason, Oracle has provided a number of data dictionary views that can be used to extract data dictionary information. There are actually three classes of views:

- ♦ **DBA_ Views**—Views with the DBA_ prefix provide a complete picture of all objects in the data dictionary and are for the use of the database administrator.
- ♦ **ALL_ Views**—Views with the ALL_ prefix allow users to get information on objects that they own as well as those that they have been granted permissions to.
- ♦ **USER_ Views**—Views with the USER_ prefix allow users to get a complete picture of all objects that they have created in their own schema.

Each view of any of the above three types is based upon a complex join of data in the base tables to provide you with the most useful information for the objects you are querying. For example, the definition of the DBA_USERS view, providing information on user accounts created in the database, is the following:

```
select u.name, u.user#, u.password,
       m.status,
       decode(u.astatus, 4, u.ltime,
                5, u.ltime,
                6, u.ltime,
                8, u.ltime,
                9, u.ltime,
                10, u.ltime, to_date(NULL)),
       decode(u.astatus,
                1, u.exptime,
                2, u.exptime,
                5, u.exptime,
                6, u.exptime,
                9, u.exptime,
                10, u.exptime,
                decode(u.ptime, '', to_date(NULL),
                       decode(pr.limit#, 2147483647, to_date(NULL),
                              decode(pr.limit#, 0,
                                     decode(dp.limit#, 2147483647, to_date(NULL), u.ptime +
                                             dp.limit#/86400),
                                     u.ptime + pr.limit#/86400))))),
       dts.name, tts.name, u.ctime, p.name, u.defschclass, u.ext_username
from sys.user$ u, sys.ts$ dts, sys.ts$ tts, sys.profname$ p,
     sys.user_astatus_map m, sys.profile$ pr, sys.profile$ dp
where u.datats# = dts.ts#
and u.resource$ = p.profile#
and u.tempts# = tts.ts#
and u.astatus = m.status#
and u.type# = 1
and u.resource$ = pr.profile#
and dp.profile# = 0
```

```
and dp.type#=1
and dp.resource#=1
and pr.type# = 1
and pr.resource# = 1
```

As you can see, it is far easier to make use of the data dictionary views than to query the base tables directly.

Uses of the Oracle data dictionary

The Oracle data dictionary is used by both Oracle server and users in the Oracle database.

The Oracle server uses the data dictionary whenever a user connects to the instance or issues a command. Whenever a user attempts to connect to the instance, the Oracle server needs to determine whether the user has permission to connect or is even a valid user in the database, all of which is stored in the data dictionary. When the user then issues a command, such as “SELECT * FROM Instructors,” the Oracle server uses the data dictionary to determine if the object being referenced exists and the user has permission to access it and perform the command. As you have seen previously, once read, this information is cached in memory in the Shared Pool to reduce the amount of I/O on the SYSTEM tablespace and thereby have data dictionary resolution take place more quickly.

The Oracle server also automatically updates the data dictionary when changes take place in the database structure, such as the addition, removal, or modification of a data file or tablespace, when permissions are granted or revoked to users, when objects are created, altered, or dropped, or when new functionality is added to the data dictionary through the invocation of scripts or procedures.

Oracle users make use of the data dictionary to determine which objects they have access to by querying the data dictionary views, or to determine the structure of an object when using the DESCRIBE command in SQL*Plus. All users have access to the ALL_ and USER_ views mentioned previously, while only database administrators have access to the DBA_ views, as well as the USER_ and ALL_ views. When querying these data dictionary views, users do not need to change the SQL syntax they are already familiar with — the same rules apply to the SELECT statement against data dictionary views as against your own tables and views.

Data Dictionary View Types



Query the data dictionary

Oracle provides several ways to get information on the database. While querying the base tables is not recommended, a set of static data dictionary views, broken down into three different categories, is available. As previously indicated, these include:

- ♦ **DBA_ Views**—Views with the DBA_ prefix provide a complete picture of all objects in the data dictionary and are for the use of the database administrator. The DBA is able to get a complete picture of all objects in all schemas and thereby have information to help solve any problems users may be having accessing database objects.
- ♦ **ALL_ Views**—Views with the ALL_ prefix allow users to get information on objects that they own as well as those that they have been granted permissions to. The ALL_ views provide information on more objects than the USER_ views, but do not provide as much information on each object. This means that a superset of the data available through USER_ views is returned, but only a subset of information on each object. All users have access to the ALL_ views so that they may be able to determine which objects they can manipulate in Oracle.
- ♦ **USER_ Views**—Views with the USER_ prefix allow users to get a complete picture of all objects that they have created in their own schema. The USER_ views provide the same information as the DBA_ views do for each object, but are limited to only those objects in the schema of the user querying the view. Objects in other users' schemas are not available through the USER_ views.

The DBA_, ALL_, and USER_ views are always plural in their naming. For example, you query the DBA_TABLES, DBA_OBJECTS, views and not DBA_TABLE or DBA_OBJECT. This helps to differentiate them from dynamic performance views, discussed later, which are singular in their naming, with only a few exceptions.

Another type of view that can be used to query information in the data dictionary, as well as the operational state of the instance is the V\$ view (also known as the dynamic performance view). These views query *virtual tables* that are defined when the instance is started. The information retrieved through dynamic performance views is not persistent between instance restarts, which means that each time the instance is started the data values are reset.

DBA_ Views

Oracle8i contains over 150 views that are prefixed with DBA_. By default, only users who have been granted the DBA role have access to these views. Only those Oracle users who have been granted the SELECT ANY TABLE privilege can query these views.



Roles and privileges will be discussed in Part V later in this book.

The philosophy behind DBA_ views is quite simple: database administrators need to always have access to all objects in the database so that they may perform their jobs with a minimum of problems. Furthermore, since the database administrator, for all intents and purposes, “owns” the database, DBAs should have complete access to all database contents. The DBA_ views provide this functionality to database administrators.

Although the vast majority of the DBA_ views in Oracle present the same information as is available in the USER_ data dictionary views, the main difference is the inclusion of the OWNER column in the DBA_ views, when dealing with database objects that are created by users. Furthermore, some of the DBA_ views are only available to the database administrator because their scope is database-wide. For example, the DBA_ROLLBACK_SEGS data dictionary view, which provides information on rollback segments, does not have a corresponding USER_ or ALL_ view because users would not create rollback segments — only the database administrator performs this function. Similarly, only the DBA creates user profiles and assigns them to users, so there is no USER_PROFILES view, but there is a DBA_PROFILES view. Finally, the DBA controls any physical attributes of the database, such as the creation of data files to be used by tablespaces in the database, so only a DBA_DATA_FILES view exists, without a corresponding ALL_ or USER_ view.

For example, if you wanted to retrieve a list of tables and the tablespaces in which they are created for the 'STUDENT' schema, you would issue the following query:

```
SQL> SELECT table_name, tablespace_name FROM DBA_TABLES
      2  WHERE owner='STUDENT'
      3* ORDER BY table_name;
```

TABLE_NAME	TABLESPACE_NAME
BATCHJOBS	CERTDB
CLASSENROLLMENT	CERTDB
COURSEAUDIT	CERTDB
COURSES	CERTDB
INSTRUCTORS	CERTDB
LOCATIONS	CERTDB
SCHEDULEDCLASSES	CERTDB
STUDENTS	CERTDB

8 rows selected.

If you then wanted to know which datafiles on disk the CERTDB tablespace used for storage, you could issue the following command:

```
SQL> col file_name format a40
SQL> SELECT file_name, TO_CHAR(bytes, '999,999,999,999') FILESIZE, status
      2  FROM DBA_DATA_FILES
      3* WHERE tablespace_name = 'CERTDB';
```

FILE_NAME	FILESIZE	STATUS
C:\CERTDB\CERTDB01.DBF	10,485,760	AVAILABLE

SQL>

If the user STUDENT wanted to, he or she could issue the first query against the USER_TABLES data dictionary view and return the same result; however, only the DBA can issue the second command to find out in which file or disk the CERTDB tablespace is located.



Throughout this book as topics are introduced you will also be presented with information on which DBA_ views can be used to query information on the items being discussed. This chapter serves only as an introduction to these views and does not provide a comprehensive listing of the views available.

USER_ Views

While DBA_ views are designed to be used by database administrators, USER_ views are designed to allow any user to get a complete picture of the objects that he or she owns (that is, those objects in the user's schema). The number of USER_ views available in Oracle is over 140.

The big difference between DBA_ views and USER_ views is that most USER_ views do not have an OWNER column, as the owner is assumed to be the current user. The information returned for each row when querying any USER_ view is identical to the corresponding DBA_ view, with the exception of the OWNER column.

For example, the DBA_USERS view can be used by the database administrator to get information on users in the database. Its structure is as follows:

```
SQL> DESC DBA_USERS;
Name                                     Null?    Type
-----
USERNAME                                 NOT NULL VARCHAR2(30)
USER_ID                                  NOT NULL NUMBER
PASSWORD                                          VARCHAR2(30)
ACCOUNT_STATUS                           NOT NULL VARCHAR2(32)
LOCK_DATE                                          DATE
EXPIRY_DATE                                       DATE
DEFAULT_TABLESPACE                       NOT NULL VARCHAR2(30)
TEMPORARY_TABLESPACE                     NOT NULL VARCHAR2(30)
CREATED                                   NOT NULL DATE
PROFILE                                   NOT NULL VARCHAR2(30)
INITIAL_RSRC_CONSUMER_GROUP              VARCHAR2(30)
EXTERNAL_NAME                             VARCHAR2(4000)
```

SQL>

If you were a regular user in the database and wanted to get information on your user account, you would query the USER_USERS data dictionary view, whose structure is as follows:

```
SQL> DESC USER_USERS;
Name                                     Null?    Type
-----
USERNAME                                 NOT NULL VARCHAR2(30)
USER_ID                                  NOT NULL NUMBER
ACCOUNT_STATUS                           NOT NULL VARCHAR2(32)
```

```

LOCK_DATE                DATE
EXPIRY_DATE              DATE
DEFAULT_TABLESPACE       NOT NULL VARCHAR2(30)
TEMPORARY_TABLESPACE     NOT NULL VARCHAR2(30)
CREATED                  NOT NULL DATE
INITIAL_RSRC_CONSUMER_GROUP  VARCHAR2(30)
EXTERNAL_NAME            VARCHAR2(4000)

```

SQL>

With the exception of the PROFILE and PASSWORD columns that are visible in the DBA_USERS view, all the other columns provide the same information in both views, though USER_USERS always returns only a single row describing the current user, whereas the DBA_USERS view may return any number of rows.

USER_ views are handy when you need to determine what the structure of a table is, or what objects you have created in your schema. For example, to get a complete list of all objects in your schema, as well as their object type, you can issue the following query:

```

SQL> col object_name format a30
SQL> SELECT object_name, object_type, status
       2 FROM USER_OBJECTS
       3 ORDER BY object_name;

```

OBJECT_NAME	OBJECT_TYPE	STATUS
BATCHJOBS	TABLE	VALID
BATCHJOBS_JOBID_PK	INDEX	VALID
CLASSENROLLMENT	TABLE	VALID
COURSEAUDIT	TABLE	VALID
COURSEAUDIT_PK	INDEX	VALID
COURSES	TABLE	VALID
INSTRUCTORCOST	INDEX	VALID
INSTRUCTORS	TABLE	VALID
INSTRUCTORS_LASTNAME_IDX	INDEX	VALID
LOCATIONS	TABLE	VALID
PK_CLASSID	INDEX	VALID
PK_CLASSID_STUDENTNUMBER	INDEX	VALID
PK_COURSENUMBER	INDEX	VALID
PK_INSTRUCTORID	INDEX	VALID
PK_LOCATIONID	INDEX	VALID
PK_STUDENTNUMBER	INDEX	VALID
SCHEDULEDCLASSES	TABLE	VALID
STUDENTLASTNAME	INDEX	VALID
STUDENTS	TABLE	VALID

19 rows selected.

SQL>

ALL_ Views

When users need to determine which objects are available to them, the ALL_ views provide the information. ALL_ views provide users a complete perspective of what the database looks like to them. Querying ALL_ views lets you know what objects you own or have been granted permissions to by other users. From your perspective, this is the totality of database objects available to you. This is because Oracle's security model will not allow you to see any objects to which you have not explicitly been granted permissions.

For example, if you connect to the database instance as the user Student, to get a listing of tables that you have the capability to access, you may issue the following command:

```
SQL> SELECT owner, table_name FROM ALL_TABLES
      2 ORDER BY owner, table_name;
```

OWNER	TABLE_NAME
MDSYS	CS_SRS
MDSYS	MD\$DICTVER
MDSYS	OGIS_SPATIAL_REFERENCE_SYSTEMS
MTSSYS	MTS_PROXY_INFO
STUDENT	BATCHJOBS
STUDENT	CLASSENROLLMENT
STUDENT	COURSEAUDIT
STUDENT	COURSES
STUDENT	INSTRUCTORS
STUDENT	LOCATIONS
STUDENT	SCHEDULEDCLASSES
STUDENT	STUDENTS
SYS	AUDIT_ACTIONS
SYS	DUAL
SYS	PSTUBTBL
SYS	STMT_AUDIT_OPTION_MAP
SYS	SYSTEM_PRIVILEGE_MAP
SYS	TABLE_PRIVILEGE_MAP
SYSTEM	DEF\$_TEMP\$LOB
SYSTEM	HELP

20 rows selected.

```
SQL>
```

As you can see, the list includes those tables owned by Student, as well as tables owned by SYS, such as DUAL, and tables owned by SYSTEM, MDSYS, and MTSSYS. The list of tables may vary depending on the Oracle options installed on the system and database, as well as which other users' schemas objects you have been given permissions to.

Special data dictionary views

Oracle also has some data dictionary views that do not use the `DBA_`, `ALL_`, or `USER_` prefixes. These include the views `DICTIONARY` and `DICTIONARY_COLUMNS`. The `DICTIONARY` view also has a synonym `DICTIONARY`, which can be used as shorthand.

The `DICTIONARY` view allows you to get information on the `DBA_`, `ALL_`, and `USER_` views, as well as other data dictionary and dynamic performance views that you have access to and you can use to query the data dictionary, as well as any comments that were placed on them to give you more information on their use. This view does not include any tables or views that are not part of the data dictionary, such as those that you may have created in your own schema.

For example, when connected to the instance as `Student`, to get a complete list of views that contain the string `TABLE` in their name, as well as their comments, you would execute the following statement:

```
SQL> SELECT * FROM DICTIONARY
      2* WHERE table_name LIKE '%TABLE%';
```

TABLE_NAME	COMMENTS
ALL_ALL_TABLES	Description of all object and relational tables accessible to the user
ALL_NESTED_TABLES	Description of nested tables in tables accessible to the user
ALL_OBJECT_TABLES	Description of all object tables accessible to the user
ALL_PART_TABLES	
ALL_QUEUE_TABLES	All queue tables accessible to the user
ALL_TABLES	Description of relational tables accessible to the user
ALL_UPDATABLE_COLUMNS	Description of all updatable columns
USER_ALL_TABLES	Description of all object and relational tables owned by the user's
USER_NESTED_TABLES	Description of nested tables contained in the user's own tables
USER_OBJECT_TABLES	Description of the user's own object tables
USER_PART_TABLES	
USER_QUEUE_TABLES	All queue tables created by the user
USER_TABLES	Description of the user's own relational tables
USER_TABLESPACES	Description of accessible tablespaces
USER_UPDATABLE_COLUMNS	Description of updatable columns
TABLE_PRIVILEGES	Grants on objects for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee

16 rows selected.

```
SQL>
```

The DICT_COLUMNS view allows you to find the structure of the views and tables returned by the DICTIONARY view. Its structure is as follows:

```
SQL> DESC DICT_COLUMNS;
Name                                                    Null?    Type
-----
TABLE_NAME                                             VARCHAR2(30)
COLUMN_NAME                                            VARCHAR2(30)
COMMENTS                                              VARCHAR2(4000)

SQL>
```

Dynamic performance views

One of the most useful sets of views from the perspective of the database administrator is the dynamic performance, or V\$ views. These views are often referred to as *virtual tables* as they do not have any corresponding physical structure, like a regular table, and they are not based on any physical tables, like a normal database view is. They are initialized and populated on instance startup, and maintained by Oracle as the instance is running. When the instance is shut down, any values they contained disappear.

Dynamic performance views are designed to help the DBA gauge performance of the database, determine the current running status, monitor database operations and so on. Although some of them are available to regular Oracle users, the vast majority are only accessible to the database administrator. Their names are, with a small number of exceptions, specified in the singular such as V\$DATABASE, or V\$DATAFILE. This is in contrast to the data dictionary views, such as DBA_, ALL_, or USER_, whose names are specified in the plural.

The Oracle database does not have a consistent number of dynamic performance views as they may be added to the database by running scripts, by turning on database features, or specifying INIT.ORA parameters. The complete list of views that are available for a given instance can be retrieved by querying the V\$FIXED_TABLE dynamic performance view when connected to the instance as a user with the DBA role. For example, to find out how many dynamic performance views are available, you may issue the following command:

```
SQL> SELECT COUNT(*) FROM V$FIXED_TABLE;

COUNT(*)
-----
        641

SQL>
```

An important thing to remember about V\$ views is that they are available even if the database is not open. As long as the instance is mounted, the V\$ views can be queried, although their results may not always be available, as shown here:

```
SQL> SELECT STATUS FROM V$INSTANCE;

STATUS
-----
OPEN

SQL> SELECT COUNT(*) FROM V$FIXED_TABLE;

  COUNT(*)
-----
        641

SQL> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT;
ORACLE instance started.

Total System Global Area  89114652 bytes
Fixed Size                 75804 bytes
Variable Size             57118720 bytes
Database Buffers          31842304 bytes
Redo Buffers               77824 bytes
Database Mounted

SQL> SELECT STATUS FROM V$INSTANCE;

STATUS
-----
MOUNTED

SQL> SELECT COUNT(*) FROM V$FIXED_TABLE;

  COUNT(*)
-----
        641

SQL> SELECT * FROM V$TABLESPACE;

 TS# NAME
-----
  0 SYSTEM
  3 TEMP
  1 RBS
  5 INDX
  4 TOOLS
```

```

6 DRSYS
2 USERS
8 OEM_REPOSITORY
9 CERTDB

```

9 rows selected.

SQL>

Of the 641 dynamic performance views returned by the previous query when the instance was open, as well as mounted, a much smaller portion actually deals with V\$ views. This is because the V\$FIXED_TABLE view also includes X\$ and GV\$ views. The X\$ views are base views, rarely queried directly, upon which the V\$ and GV\$ views are based. The GV\$ views and V\$ views return the same results, with the V\$ views being synonyms for the GV\$ views, in most cases.

The actual number of V\$ views in this database is found by issuing the following command:

```

SQL> SELECT COUNT(*) FROM V$FIXED_TABLE
2 WHERE NAME LIKE 'V$%';

```

```

COUNT(*)
-----
185

```

SQL>

In general, the most commonly used views deal with the instance, memory, and performance issues. Their information is not accessed from the database, but rather the control file and memory. For example, to get a list of currently running INIT.ORA parameters, you can query the V\$PARAMETER view, or to find out the physical location of the control file, the V\$CONTROL_FILE view can be queried, as shown here:

```

SQL> COL NAME FORMAT A55
SQL> SELECT * FROM V$CONTROLFILE;

```

```

STATUS NAME
-----
D:\ORACLE\ORADATA\ORCL\CONTROL01.CTL
D:\ORACLE\ORADATA\ORCL\CONTROL02.CTL
D:\ORACLE\ORADATA\ORCL\CONTROL03.CTL

```

SQL>



Later in this book, you make extensive use of all the different types of views introduced here. For this reason, a detailed discussion of every view available did not take place. Future chapters introduce you to the views you need to do your job, as well as answer the relevant questions on the exam.

Stored Program Units

Objective

Administer stored procedures and packages

List the types of database event triggers

The Oracle data dictionary is not only composed of views and base tables. Oracle also creates other database objects to make the use and administration of the database easier. Some of these are stored procedures, functions, triggers, and packages — collectively known as stored program units. They are created when the data dictionary is built with the CREATE DATABASE statement and further extended with other scripts that are run, such as CATALOG.SQL and CATPROC.SQL (discussed later in this chapter).

Stored program unit types

Oracle provides support for several different types of stored program units, including:

- ♦ **PL/SQL Program Units** — Functions, procedures, and packages written in Oracle's PL/SQL language fit into this type. PL/SQL allows you to mix SQL code, to manipulate data, with procedural constructs such as loops, conditional logic (IF..ELSE...END IF), and identifiers (such as variables, cursors, PL/SQL records, and so on).



The "Oracle8i DBA: Architecture and Administration" exam does not test your in-depth knowledge of PL/SQL and how it can be used to create stored program units such as triggers, procedures, packages, and others. However, it is a good idea to have a fundamental understanding of PL/SQL since it enables you to create programs that can make your DBA tasks more efficient. For more information on PL/SQL and how to use it, consult the *PL/SQL User's Guide and Reference*, a part of the Oracle documentation set.

- ♦ **Java Program Units** — Oracle8 introduced support for Java in the database, and Oracle8i continues to allow you to create stored program units using Java. After you install the JServer component of Oracle in the database, you are able to create Java program units and have them exposed to SQL and PL/SQL by publishing the call specification. The call specification maps the Java method you have created, its parameters and their datatypes, as well as any return parameters and datatypes to their corresponding SQL and PL/SQL components. In this way Java program units can call PL/SQL program units, and vice versa.



Like PL/SQL program units, the "Oracle8i DBA: Architecture and Administration" exam will not test your in-depth knowledge of Java. If you want to know how to configure Java support and the basics of using Java in Oracle8i, please consult the *Oracle8i Java Developer's Guide*, a part of the Oracle documentation set.

- ♦ **External Program Units**— Oracle8i introduced support for external program units, primarily written in the C programming language, that are accessed in the form of a shared library (on UNIX systems) or dynamic link library (DLL) on WindowsNT/2000 platforms. External program units can be used to perform complex calculations that may not be efficient in Oracle, or to extend the functionality of the database by adding support for features not present in Oracle. External program units can be functions or procedures that can be called from Oracle because their names, parameters and datatypes, and return parameters and datatypes are published in Oracle to allow PL/SQL or Java program units to reference and use them.



Like PL/SQL program units, the “Oracle8i DBA: Architecture and Administration” exam will not test your in-depth knowledge of external program units and how to make use of them. For more information on how to make external program units work, as well as how to configure database support for them, refer to the “External Routines” chapter of the *Oracle8i Application Developer’s Guide – Fundamentals*, a part of the Oracle documentation set.

Benefits of stored program units

No matter which stored program unit type you decide to implement, or have already implemented, the advantages are similar. In fact, as soon as you create a database and run the scripts that are outlined in this chapter, you are making use of stored program units. Oracle relies upon stored program units such as packages, functions, and procedures to perform core database tasks.

Some of the benefits of stored program units include the following features:

- ♦ Once created, stored program units reside in the data dictionary with the source code, by default.
- ♦ Once created, stored program units are pre-compiled to reduce the overhead of running them. As outlined in Chapter 1, when a statement is sent to the Oracle server, it goes through three phases — parse, execute, and fetch. With stored program units, the first phase is done.
- ♦ Once invoked, stored program units are loaded into the shared pool and may optionally be pinned so that they are never flushed out, thereby increasing performance.
- ♦ Java and external program units can be called from SQL and PL/SQL after their call specification has been defined. Java and PL/SQL program units can call one another, whereas external program units can only be called from Java and PL/SQL.
- ♦ Any stored program unit, no matter how many different users are making calls to it, will only be loaded once in the shared pool making memory utilization more efficient.
- ♦ Stored program units can perform any complex task and can help in the enforcement of complex business rules.

- ♦ Users granted permissions to execute a stored program unit may be allowed to manipulate other database objects that they may not have permissions to. In this way, you have more control over the security of the data in the database by limiting access to the underlying objects through a procedure or package.
- ♦ User-defined functions can extend the SQL language of Oracle and provide functionality that meets your unique requirements.

The above list is only a sample of some of the benefits that stored program units provide. In your own environments, you may come up with a few more that satisfy requirements specific to you.

Stored PL/SQL program units

The majority of stored program units developed in Oracle environments still tend to be done in PL/SQL. Not only is PL/SQL probably the easiest programming environment to learn (Java and C seem to be way too complex for most tasks databases are used for), it also does not suffer in breadth within Oracle. Using PL/SQL you can create any program unit that Oracle supports including procedures, functions, packages, triggers, and type methods. The syntax of PL/SQL is relatively easy for anyone who has done some basic programming, and it includes full support for standard SQL DML commands, although SELECT statements are somewhat different, and DDL statements in PL/SQL are not allowed (although they can be performed when using dynamic SQL and the EXECUTE IMMEDIATE PL/SQL command).

Stored PL/SQL program units are simply a set of SQL and PL/SQL commands that are logically related to perform a specific task. Each stored program unit is given a logical name using Oracle's naming conventions and is created within a particular schema. The CREATE command is used to create program units, the CREATE OR REPLACE command is used to modify an existing program unit without removing permissions, and the DROP command is used to remove program units from the database. Stored program units can be invoked from client tools such as SQL*Plus, Enterprise Manager, and others, or may be called from other program units or SQL commands, in the case of functions and methods. Once called, stored program units are loaded into the Shared Pool and remain there until flushed out due to lack of space, unless pinned by the DBA.

Stored procedures

Stored procedures written in PL/SQL are program units that perform a specific task, and usually include code to ensure that data is properly manipulated and exceptions are properly handled. For example, a procedure to calculate the sales tax on a course might look something like this:

```
SQL> CREATE OR REPLACE PROCEDURE calculate_tax
  2  (p_course IN NUMBER)
  3  IS
  4  v_tax NUMBER;
  5  v_price NUMBER;
```

```
6 BEGIN
7 SELECT RetailPrice
8     INTO v_price
9     FROM Courses
10    WHERE CourseNumber = p_course;
11
12    v_tax := v_price *.15;
13
14 DBMS_OUTPUT.PUT_LINE(
15 'Tax on that course is ' || TO_CHAR(v_tax, '$9,990.00'));
16 EXCEPTION
17    WHEN NO_DATA_FOUND THEN
18        DBMS_OUTPUT.PUT_LINE(
19 'Course number ' || p_course || ' does not exist. ');
20    WHEN OTHERS THEN
21        DBMS_OUTPUT.PUT_LINE(
22 'Error ' || SQLCODE || ' occurred. ');
23 DBMS_OUTPUT.PUT_LINE(
24 'The error message is: ' || SQLERRM);
25 END;
26 /
```

Procedure created.

SQL>

As you can see in the previous code example, procedures can take parameters (`p_course`) and those parameters can be input parameters (IN), which is the default, output parameters (OUT), which will return data to the calling environment, or input and output parameters (IN OUT), which do both. Parameters are not required for a procedure, but they are typically used as they make the procedure more flexible allowing it to be used in different ways.

Procedures can also have identifiers defined in them, such as variables or cursors, and so on. Variables, and other identifiers are defined following the keyword `IS` or `AS` and before the keyword `BEGIN`, indicating the start of the executable section, or main body, of the procedure. The area where identifiers are declared is called the declaration section. A procedure may define one, more, or no identifiers — depending on its requirements and what it does.

The next part of the procedure is the main body, which starts with the keyword `BEGIN` and contains the actual procedure code. Here you can use valid PL/SQL commands to perform any operations that are necessary to complete the task of the procedure. Note that `SELECT` statements follow a different syntax than in regular SQL, and must have their results stored in variables or PL/SQL records. Also, a single `SELECT` statement is only allowed to return one row, whereas `INSERT`, `UPDATE`, and `DELETE` statements work the same way as in regular SQL syntax from the SQL*Plus command line.

Finally, after the main body, the keyword `EXCEPTION` indicates that the procedure has an exception handler where exceptions or errors will be processed should they occur, such as `NO_DATA_FOUND`, indicating that the `SELECT` statement did not return a row. Exceptions can be user-defined exceptions declared in the declaration section of the procedure, or Oracle pre-defined exceptions such as `NO_DATA_FOUND`, `TOO_MANY_ROWS`, and so on. The generic `WHEN OTHERS` in the exception handler allows you to handle any other exception within the code of the procedure.

To make use of the `calculate_tax` stored procedure and determine the tax on course number 310 (Advanced PL/SQL), you would issue the following commands:

```
SQL> set serveroutput on;
SQL> EXECUTE calculate_tax(310);
Tax on that course is      $300.00

PL/SQL procedure successfully completed.

SQL>
```

Although the preceding example is somewhat simplistic, PL/SQL stored procedures can perform extremely complex operations and, in many cases, do just that. For more information on how to create PL/SQL stored procedures and PL/SQL, refer to the *PL/SQL User's Guide and Reference*, part of the Oracle8i documentation set.

Functions

Another PL/SQL stored program unit that enables you to extend the functionality of your Oracle database is a PL/SQL function. A PL/SQL function is similar to a stored procedure but may be used in regular SQL syntax and cannot have `OUT` or `IN OUT` parameters defined. Furthermore, a PL/SQL function must specify a return datatype and will always return a value. For example, if you wanted to return the price, sales tax and full price for each course in the `Courses` table, you can create a function to calculate the sales tax, and then use that in your SQL statements, as follows:

```
SQL> SQL> CREATE OR REPLACE FUNCTION tax_calc
 2   (p_price IN NUMBER)
 3   RETURN NUMBER
 4   IS
 5   v_tax NUMBER;
 6   BEGIN
 7     v_tax := p_price *.15;
 8     RETURN v_tax;
 9   END;
10 /
```

Function created.

```
SQL>
```

```
SQL> SELECT CourseNumber, TO_CHAR(RetailPrice, '$99,999.99') as Price,
2      TO_CHAR(tax_calc(RetailPrice), '$99,999.99') as Tax,
3      TO_CHAR((RetailPrice + tax_calc(RetailPrice)), '$99,999.99') as TotalCost
4 FROM Courses;
```

COURSENUMBER	PRICE	TAX	TOTALCOST
100	\$2,000.00	\$300.00	\$2,300.00
110	\$2,000.00	\$300.00	\$2,300.00
201	\$4,000.00	\$600.00	\$4,600.00
200	\$4,000.00	\$600.00	\$4,600.00
210	\$4,500.00	\$675.00	\$5,175.00
220	\$3,000.00	\$450.00	\$3,450.00
300	\$2,500.00	\$375.00	\$2,875.00
310	\$2,000.00	\$300.00	\$2,300.00
320	\$1,750.00	\$262.50	\$2,012.50

9 rows selected.

SQL>

The `tax_calc` function takes a number in as a parameter and also returns a number. Because it is not tied to any one table, it can be used to calculate taxes on anything in the database, provided the value passed to it is a number. The `SELECT` statement uses the `tax_calc` function on the `RetailPrice` of each course to return both the tax to be paid and the total cost, with taxes, for each course.

Functions must follow a few rules. They cannot modify data in any table in the database, and may also be prohibited from changing the value of any variables that are declared outside the function itself. For more information on functions, refer to the *PL/SQL User's Guide and Reference*, part of the Oracle8i documentation set.

Triggers

PL/SQL code can be used to write database triggers. Database triggers are PL/SQL programs that fire when certain actions occur within the database. They are useful for ensuring database integrity, or protecting data from changes, or for logging user actions or dealing with Oracle server errors.

A single table can have many database triggers, and one database trigger can be invoked by multiple events and perform many actions.

Tip

A useful guideline for determining whether to create separate triggers or to combine them into one trigger is to ask yourself how many different business rules you are enforcing. Create one database trigger for each business rule. Triggers may call other procedures or functions, which may actually perform more complex actions.

Triggering events

Three events in the database can cause a database trigger to fire: `DELETE`, `INSERT`, and `UPDATE`. Database triggers fire regardless of how the `DELETE`, `INSERT`, or `UPDATE` command is issued; this is why they are useful for ensuring database

integrity. For example, if you have a Web page and an Oracle Developer application that sends updates to the database and you have a support team that uses SQL*Plus to update the database, the database triggers will fire regardless of who or what application issues the command.

Statement-level triggers

Statement-level triggers fire once for each triggering event. In the trigger code, you can perform validation and calculations, execute SQL statements, and call other PL/SQL programs. When you raise an error in a trigger, the triggering UPDATE, INSERT, or DELETE is rolled back. Statement-level triggers are used to prevent users from performing modifications to a table under specified conditions.

When you create a statement-level trigger, you must specify the following in the trigger heading:

- ♦ Trigger name
- ♦ Trigger table
- ♦ Triggering event
- ♦ BEFORE or AFTER event

The trigger name is a program name you give to the trigger. The trigger table is the name of the database table that is being updated when you want the trigger to fire. The triggering event is INSERT, UPDATE, DELETE, or a combination of the three. The triggering event specifies what commands on the trigger table will cause the trigger to fire. A BEFORE or an AFTER event specifies whether the database trigger will fire before or after the update that caused the trigger to fire.



Tip

If you are creating a trigger and it does not matter whether the trigger fires before or after the triggering event, fire the trigger after the event for best performance results.

All these parameters are specified in the trigger header as follows:

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE|AFTER trigger_event
ON trigger_table
```

For example, suppose a batch program CLASS_STATUS updates the status of scheduled classes based on the records in the ClassEnrollment table. When CLASS_STATUS is running, you do not want anyone to update the ClassEnrollment table. CLASS_STATUS updates a record in the table BatchJobs with the status “RUNNING” when it executes. You want to create a database trigger that will stop users from updating, inserting, or deleting records from the ClassEnrollment table when CLASS_STATUS is running, as follows:

```

SQL> CREATE OR REPLACE TRIGGER check_batch_job
 2 BEFORE INSERT OR UPDATE OR DELETE ON ClassEnrollment
 3 DECLARE
 4   v_status   batchjobs.status%TYPE;
 5 BEGIN
 6   SELECT status
 7   INTO v_status
 8   FROM batchjobs
 9   WHERE jobname = 'CLASS_STATUS';
10
11 IF v_status = 'RUNNING' THEN
12   RAISE_APPLICATION_ERROR (-20001,'Error: Batch job class_status is
running. Updates are not allowed');
13 END IF;
14 END;
SQL> /

```

Trigger created.

SQL>

This code checks the status for the CLASS_STATUS record in the BatchJobs table. If the status is “RUNNING,” you will raise an error that will rollback the triggering event and prevent the user from making changes to the ClassEnrollment table. If you attempt to update the classenrollment table when the batch job CLASS_STATUS is running, you will get an error, and the update will be rolled back.

```

SQL> UPDATE classenrollment
 2 SET status = 'Confirmed'
 3 WHERE classid = 53
 4 AND studentnumber = 1003
 5 /
UPDATE classenrollment
  *
ERROR at line 1:
ORA-20001: Error: Batch job
class_status is running, updates are not allowed
ORA-06512: at "STUDENT3.CHECK_BATCH_JOB", line 10
ORA-04088: error during execution of trigger
'STUDENT3.CHECK_BATCH_JOB'

```

Row-level triggers

Row-level triggers fire once for each record updated, deleted, or inserted. They are often used to perform complex validation or calculations, to populate default values, to modify data in related tables, or to modify data values.

Row-level triggers are created by adding the statement FOR EACH ROW to the trigger header, as in the following syntax example:

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE|AFTER trigger_event
ON trigger_table
FOR EACH ROW
```

Because row-level triggers fire for each actual record being updated, you can access the values contained in those records. To access the values in the record, you specify the name of the column whose value you wish to access with the prefixes :old or :new, as follows:

- ♦ If you are updating a record, :old returns the values of the original record in the database, and :new returns the new values being sent to the database for update.
- ♦ If you are deleting a record, :old returns the values contained in the record about to be deleted, and :new returns NULL.
- ♦ If you are inserting a record, :old returns NULL, and :new returns the values about to be inserted into the database.

The following example is a trigger that ensures that the state column in the Students table is always entered in uppercase letters:

```
SQL> CREATE OR REPLACE TRIGGER upper_state
2 BEFORE INSERT OR UPDATE OF State ON Students
3 FOR EACH ROW
4 BEGIN
5     :new.State := UPPER(:new.State);
6 END;
SQL> /
```

Trigger created.

```
SQL> UPDATE Students
2 SET City = 'Boston', State = 'ma'
3 WHERE StudentNumber = 1005
4 /
```

1 row updated.

```
SQL> SELECT StudentNumber, City, State
2 FROM Students
3 WHERE StudentNumber = 1005
4 /
```

STUDENTNUMBER	CITY	ST
-----	-----	-----
1005	Boston	MA

SQL>

The WHEN clause can be used on row-level triggers to prevent unnecessary firing of a database trigger. A condition is specified in the WHEN clause. If that condition is not met, the database trigger does not fire. The WHEN clause is added to the trigger header after the FOR EACH ROW clause. The syntax is as follows:

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE|AFTER trigger_event ON trigger_table
FOR EACH ROW
WHEN (condition)
```

For example, U.S. ZIP codes are numeric, and Canadian postal codes are alphanumeric. If a Canadian record is entered, you want to ensure the postal code is all uppercase. This is unnecessary in the United States, because only numbers are specified. To avoid wasting processing time when entering U.S. records in the students table, you add a WHEN clause so the trigger only fires for Canadian students:

```
SQL> CREATE OR REPLACE TRIGGER canada_postal_code
  2 BEFORE INSERT OR UPDATE OF PostalCode ON Students
  3 FOR EACH ROW
  4 WHEN (new.Country = 'Canada')
  5 BEGIN
  6   :new.PostalCode := UPPER(:new.PostalCode);
  7 END;
SQL> /
```

Trigger created.



Caution

When referencing the :new and :old prefixes in the WHEN clause, you do not use the colon (:) in front of the prefix.

```
SQL> UPDATE Students
  2 SET PostalCode = 'm5h 5f6'
  3 WHERE StudentNumber = 1003
  4 /
```

1 row updated.

```
SQL> SELECT StudentNumber, PostalCode
  2 FROM Students
  3 WHERE StudentNumber = 1003
  4 /
```

```
STUDENTNUMBER POSTALCODE
-----
1003 M5H 5F6
```

Trigger predicates

When you have a trigger that fires for multiple triggering events, such as INSERT and UPDATE, you may wish to perform different actions depending on which event actually caused the trigger to fire. Trigger predicates enable you to check which action was performed from within the trigger. There are four trigger predicates:

- ♦ INSERTING: Returns TRUE if INSERT statement fired trigger
- ♦ DELETING: Returns TRUE if DELETE statement fired trigger
- ♦ UPDATING: Returns TRUE if UPDATE statement fired trigger
- ♦ UPDATING(column_name): Returns TRUE if UPDATE statement, which updates the specified column, fired the trigger

These predicates are referenced within the database trigger code. They return TRUE or FALSE depending on the action that caused the trigger to fire.

For example, you might want to use a database trigger to keep an audit of changes to the courses table. You need to track when new courses are added and deleted and when course prices are changed.

```
SQL> CREATE OR REPLACE TRIGGER Audit_Courses
  2 AFTER INSERT OR DELETE OR UPDATE ON Courses
  3 FOR EACH ROW
  4 BEGIN
  5     IF INSERTING THEN
  6         INSERT INTO CourseAudit
  7             (CourseNumber,Change,Price,ChangedBy,DateChanged)
  8             VALUES
  9                 (:new.CourseNumber, 'INSERT', :new.RetailPrice, USER, SYSDATE);
 10     ELSIF DELETING THEN
 11         INSERT INTO CourseAudit
 12             (CourseNumber,Change,Price,ChangedBy,DateChanged)
 13             VALUES
 14                 (:old.CourseNumber,'DELETE',:old.RetailPrice, USER, SYSDATE);
 15     ELSIF UPDATING THEN
 16         INSERT INTO CourseAudit
 17             (CourseNumber,Change,Price,ChangedBy,DateChanged)
 18             VALUES
 19                 (:old.CourseNumber, 'UPDATE PRICE', :old.RetailPrice, USER, SYSDATE);
 20     END IF;
 21 END;
 22 /
```

Trigger created.

```
SQL> INSERT INTO Courses
  2 (CourseNumber, CourseName, ReplacesCourse, RetailPrice, Description)
  3 VALUES (400,'Database Concepts',NULL,1000,
  4 'A course which gives an overview of database concepts and capabilities');
```

1 row created.

```
SQL> UPDATE Courses
  2 SET RetailPrice = 1500
  3 WHERE CourseNumber = 400
  4 /
```

1 row updated.

```
SQL> DELETE FROM Courses
      2 WHERE CourseNumber = 400
      3 /
```

1 row deleted.

```
SQL> SELECT *
      2 FROM CourseAudit;
```

COURSENUMBER	CHANGE	DATECHANGED	PRICE	CHANGEDBY
400	INSERT	10-MAY-01	1000	STUDENT
400	UPDATE PRICE	10-MAY-01	1000	STUDENT
400	DELETE	10-MAY-01	1500	STUDENT

Order of firing

When you create database triggers, it is important to understand when database triggers fire with respect to each other and with respect to the triggering event. The following shows the order in which triggers and events are executed:

1. User executes a DML statement (INSERT, UPDATE, or DELETE).
2. Oracle fires any BEFORE statement-level triggers.
3. Oracle fires any BEFORE row-level triggers.
4. Oracle executes DML on that row.
5. Oracle fires any AFTER row-level triggers.
6. If more than one row is modified, steps 3 through 5 are repeated until all rows have been processed.
7. Oracle fires any AFTER statement-level triggers.

If multiple database triggers on the same table are triggered by the same triggering event, the order in which those triggers will be fired cannot be determined. When you require the commands in one trigger to be fired before another, you should combine the code into one database trigger and list the code in the order you want it executed.

INSTEAD OF triggers

If you are using complex views (that is, views that are based on more than one table) in your database, you may not be able to use UPDATE, DELETE, and INSERT statements on the view to update the tables accessed by the view. The INSTEAD OF trigger enables you to update the tables accessed by any view, including complex views. INSTEAD OF triggers can be placed only on views and always fire for each row.

You might have a view that shows all the ScheduledClasses and course information.

```
SQL> CREATE OR REPLACE VIEW CourseSchedule
2   AS SELECT sc.ClassID, sc.CourseNumber, sc.LocationID,
3           c.CourseName, c.RetailPrice
4   FROM ScheduledClasses sc, Courses c
5   WHERE sc.CourseNumber = c.CourseNumber
6   /
```

View created.

SQL>

If you try to add a new course through the view, you will get an error because Oracle does not allow you to insert data into more than one table in a single SQL statement, as shown here:

```
SQL> INSERT INTO CourseSchedule
2   (ClassID, CourseNumber, LocationId, CourseName, RetailPrice)
3   VALUES (50, 500, 200, 'SQL for beginners', 3000)
4   /
   (ClassID, CourseNumber, LocationId, CourseName, RetailPrice)
   *
```

ERROR at line 2:

ORA-01776: cannot modify more than one base table through a join view

SQL>

Using an INSTEAD OF trigger, you can tell Oracle how to add a record to the courses table when a user INSERTS into the view.

```
SQL> CREATE OR REPLACE TRIGGER add_course
2   INSTEAD OF INSERT ON CourseSchedule
3   FOR EACH ROW
4   BEGIN
5   INSERT INTO Courses
6   (CourseNumber, CourseName, ReplacesCourse, RetailPrice, Description)
7   VALUES (:new.CourseNumber, :new.CourseName, NULL,
8           :new.RetailPrice, NULL);
9   END;
10  /
```

Trigger created.

SQL>

Now when you insert a row into the CourseSchedule view, a record is created in the Courses table.

```
SQL> INSERT INTO CourseSchedule
  2     (ClassID, CourseNumber, LocationID, CourseName, RetailPrice)
  3     VALUES (50, 500, 200, 'SQL for beginners', 3000)
  4     /
```

1 row created.

```
SQL> SELECT CourseNumber, CourseName, ReplacesCourse, RetailPrice
  2     FROM Courses
  3     WHERE CourseNumber = 500
  4     /
```

COURSENUMBER	COURSENAME	REPLACESCOURSE	RETAILPRICE
500	SQL for beginners		3000

Database event triggers

Objective

List the types of database event triggers

In addition to creating triggers that fire when an INSERT, UPDATE, or DELETE is executed on a particular table or view, you can create triggers that fire when certain events occur in the database. These are called *event triggers*. There are two types of event triggers.

The first type of event triggers are *resource manager*, or *system event*, triggers that can fire after database startup, before database shutdown, and after server errors. These triggers must be created by the database administrator since they apply to the database as a whole. The DBA must connect to the instance with SYSDBA privileges as system event triggers need to reside in the SYS schema. The ON DATABASE keyword in the event trigger definition denotes a resource manager or system event trigger.

For example, you can create a trigger that writes error information to a table whenever a server error occurs.

```
SQL> connect system/manager@orcl.delphi.bradsys.com as SYSDBA;
Connected.
SQL> CREATE TABLE Error_Log (
  2     UserName varchar2(30),
  3     ErrorTime date,
  4     Error varchar2(200));
```

Table created.

```

SQL> CREATE OR REPLACE TRIGGER Track_Errors
 2 AFTER SERVERERROR
 3 ON DATABASE
 4 BEGIN
 5     INSERT INTO Error_Log (UserName, ErrorTime, Error)
 6       VALUES (ora_login_user, SYSDATE, 'Error: ' ||ora_server_error(1));
 7 END;
 8 /

```

Trigger created.

SQL>

Each event trigger has a number of attribute functions that can provide information about the triggering event. In the preceding example, `ora_login_user` returns the username of the person who caused the triggering event. `ora_server_error` accepts a position number and returns the error number with that position in the stack. Position one is the top of the stack.

The second type of event trigger is the *client* or *user* event trigger, which can fire after LOGON, before LOGOFF, or before and after most DDL and DCL commands.

For example, if you want to track when a user accesses your database, you could fire a trigger whenever a user logs on to add a record to an audit table.

```

SQL> CREATE OR REPLACE TRIGGER Track_logon
 2 AFTER LOGON
 3 ON DATABASE
 4 BEGIN
 5     INSERT INTO Audit_Action (Username,LogonTime,Action)
 6       VALUES (ora_login_user, SYSDATE, 'LOGON');
 7 END;
 8 /

```



Caution

Event triggers are available only in Oracle8i and higher. To create event triggers ON DATABASE, you must set the COMPATIBLE parameter in the INIT.ORA file to 8.1.6 or higher. You also need the ADMINISTER DATABASE TRIGGER privilege.

You can also create triggers that log the creation, modification, or deletion of tables or views, or other database objects in your schema. For example, to log an event whenever an object in your schema is altered, you could create the following trigger:

```

SQL> CREATE OR REPLACE TRIGGER Track_Alter
 2 AFTER ALTER
 3 ON SCHEMA
 4 BEGIN
 5     INSERT INTO Audit_Action (UserName,LogonTime,Action)
 6       VALUES (ora_login_user, SYSDATE, 'ALTER');
 7 END;
 8 /

```

Like row-level triggers, database event triggers can also have a WHEN clause that must evaluate to TRUE, in which case the trigger fires, or FALSE, in which case the trigger does not fire. For example, to create an event trigger that fires on any DDL command executed in the database by the user SYSTEM or SYS, you can issue the following command:

```
SQL> CREATE OR REPLACE TRIGGER Track_DBA_DDL
 2  AFTER DDL
 3  ON DATABASE
 4  WHEN (USER IN ('SYS','SYSTEM'))
 5  BEGIN
 6      INSERT INTO Audit_Action
 7          (Username, LogonTime, Action,
 8           ObjectName, ObjectType, ObjectOwner)
 9      VALUES (ora_login_user, SYSDATE, ora_sysevent,
10              ora_dict_obj_name, ora_dict_obj_type,
11              ora_dict_obj_owner);
12  END;
13  /
14  /
```

Enabling and disabling triggers

Once a database trigger has been created, it can be disabled and re-enabled using the ALTER TRIGGER command. When a trigger is created, it is automatically enabled. A disabled trigger does not fire when the triggering event takes place. A disabled trigger remains disabled until it is re-enabled using the ALTER TRIGGER command.

```
SQL> ALTER TRIGGER my_trig DISABLE;
SQL> ALTER TRIGGER my_trig ENABLE;
```

You can also use the ALTER TABLE command to enable or disable all triggers on a particular table.

```
SQL> ALTER TABLE Students ENABLE ALL TRIGGERS;
SQL> ALTER TABLE Students DISABLE ALL TRIGGERS;
```

Packages

Any PL/SQL programs can be put inside a package. A package is a grouping of PL/SQL programs stored together. By placing your PL/SQL programs inside a package, you can improve performance and increase functionality.

Performance can be improved by putting PL/SQL code in packages when you frequently call a number of programs one after the other. For example, you might want to write a program to enroll a student in a class:

1. Your program calls a check_status program that checks whether the course has been canceled.
2. The program calls a check_max program to ensure the class is not full.

3. The program calls a `create_enrollment` program that creates the enrollment record.

If you do not use a package, each of these programs is loaded into memory when it is called. If you put all these programs into a package, all the programs will be loaded into memory at once when the first program in the package is called.

Increased functionality can be added when packages are used because you can create variables in a package that hold their value throughout the database session. Without packages, you can create local variables in PL/SQL programs, but the values contained in these variables are lost when the program execution stops. When you create a package, values in any variables declared in the program specification are kept in memory and may be accessed at any time throughout the database session.

When you create a package, you must first create a package specification that lists all the public programs and variables, and then you create a package body that specifies all the PL/SQL code and any private variables you wish to create.

Package specification

The package specification is a list of all the PL/SQL procedures and functions you are going to include in the package that will be public. *Public* programs are programs that can be accessed by anyone with permissions to execute the package. The package specification also lists any public variables you wish to create. A public variable is a variable that can be read or modified by anyone with permissions to execute the package.

To create the package specification, you must specify the following:

- ♦ A package name
- ♦ A list of programs and their parameters
- ♦ A list of public variables

The syntax for creating a package is as shown here:

```
CREATE OR REPLACE PACKAGE package_name
IS
    variable_declarations;
    program_declarations;
END;
/
```

For example, you might want to create an enrollment package that contains the public programs `CHECK_CANCELLED` and `ENROLL_STUDENT`, and a variable called `class_maximum`.

```

CREATE OR REPLACE PACKAGE enrollment
IS
    class_maximum NUMBER := 16;

    FUNCTION check_cancelled (p_classid IN NUMBER)
        RETURN BOOLEAN;

    PROCEDURE enroll_student
        (p_studentnumber IN NUMBER,
         p_classid IN NUMBER,
         p_price IN NUMBER);
END;
/

```

Package body

Once you have created the package specification, you can create the package body. The package body contains the code for all of the programs listed in the package specification.

Additional programs can be included in the package body that are not listed in the package specification. These are called *private* programs because they can be called only by other programs within the package. Additional variables can be included in the package body that are not listed in the package itself. These are called *private* variables because they cannot be accessed from outside the package.

To create the package body, you use the CREATE PACKAGE BODY command and specify the same package name you used for the package specification.

```

CREATE OR REPLACE PACKAGE BODY package_name
IS
    variable_declarations;
    program code;
END;
/

```

For example, for your enrollment package, you need three programs: the two public programs, `check_cancelled`, `enroll_student` and `check_maximum`, as well as `Check_maximum`, which is a private function because it was not declared in the package specification.

```

CREATE OR REPLACE PACKAGE BODY enrollment
IS
    FUNCTION check_cancelled (p_classid IN NUMBER)
        RETURN BOOLEAN
    IS
        v_status ScheduledClasses.status%TYPE;
    BEGIN
        SELECT status
           INTO v_status
          FROM scheduledclasses
         WHERE classid = p_classid;

```

```
        IF v_status = 'Cancelled' THEN
            RETURN(TRUE);
        ELSE
            RETURN(FALSE);
        END IF;
    END check_cancelled;

FUNCTION check_maximum (p_classid IN NUMBER)
    RETURN BOOLEAN
IS
    v_total NUMBER;
BEGIN
    SELECT COUNT(*)
        INTO v_total
        FROM classenrollment
        WHERE classid = p_classid;

    IF v_total >= class_maximum THEN
        RETURN(TRUE);
    ELSE
        RETURN(FALSE);
    END IF;
END check_maximum;

PROCEDURE enroll_student
    (p_studentnumber IN NUMBER,
    p_classid IN NUMBER,
    p_price IN NUMBER)
IS
BEGIN

    IF check_cancelled(p_classid) THEN
        RAISE_APPLICATION_ERROR
            (-20002,'Course is cancelled');
    END IF;

    IF check_maximum(p_classid) THEN
        RAISE_APPLICATION_ERROR
            (-20003,'Course is full');
    END IF;

    INSERT INTO ClassEnrollment
        (ClassId, StudentNumber, Status,
        EnrollmentDate, Price, Grade, Comments)
    VALUES (p_classid, p_studentnumber, 'Hold',
        SYSDATE, p_price, NULL, NULL);

    COMMIT;
END enroll_student;

END enrollment;
/
```

Accessing programs and variables in packages

When you call a program in a package, you must begin the program name with the package name as a prefix.

```
SQL> EXECUTE enrollment.enroll_student(1000,51, 2000)
```

The only time you do not need to use the package prefix is when you are calling a program in a package from another program within the same package.

When you want to access a variable that is declared in a package, you must begin the variable name with the package name as a prefix.

```
SQL> BEGIN enrollment.class_maximum := 20; END;
      2 /
```

The only time you do not need to use the package prefix is when you are referencing a variable in a package from a program in the same package, as the `check_maximum` function does when verifying that the number of students in the class does not exceed the value of the `class_maximum` package variable.

To determine what the parameters of the various public functions and procedures are, you can use the `describe` command against the package, as follows:

```
SQL> desc enrollment;
FUNCTION CHECK_CANCELLED RETURNS BOOLEAN
Argument Name                Type                In/Out Default?
-----
P_CLASSID                    NUMBER              IN
PROCEDURE ENROLL_STUDENT
Argument Name                Type                In/Out Default?
-----
P_STUDENTNUMBER              NUMBER              IN
P_CLASSID                    NUMBER              IN
P_PRICE                      NUMBER              IN
```

```
SQL>
```

Oracle data dictionary packages

Perhaps you have been wondering why the previous section introduced you to a number of programming concepts in a book about database administration. The answer is quite simple — packages and other program units are heavily used by Oracle to perform maintenance tasks and to enable you to determine if you have structural problems in the database, such as corruption. Oracle also uses package extensively to support its own internal workings, such as use of the `DBMS_LOB`

package and its contents to deal with manipulation of large objects in your database. Not only does the DBMS_LOB package provide routines for managing large objects, but it also sets limits for LOBs, such as a maximum size of 4GB.

While Oracle has many packages created as part of the database creation process, depending on the options chosen, some that are key and that you should keep in mind include:

- ♦ **DBMS_LOB** — The DBMS_LOB package provides routines for opening (DBMS_LOB.OPEN), reading (DBMS_LOB.READ), writing (DBMS_LOB.WRITE) and closing (DBMS_LOB.CLOSE) columns using large object types (BLOB, CLOB, NCLOB, BFILE). Procedures for creating and managing temporary LOBs (DBMS_LOB.CREATETEMPORARY, DBMS_LOB.FREETEMPORARY) are also available, as are routines for comparing LOBs (DBMS_LOB.COMPARE), and much more.
- ♦ **DBMS_SESSION** — The DBMS_SESSION package allows a user or developer to change various session parameters within PL/SQL program units. While you would normally use the ALTER SESSION command to change settings in your own session, DDL statements are not allowed in PL/SQL. DBMS_SESSION package procedures allow you to change active roles (DBMS_SESSION.SET_ROLE procedure), turn tracing on or off (DBMS_SESSION.SET_SQL_TRACE), change NLS parameters (DBMS_SESSION.SET_NLS) and more.
- ♦ **DBMS_UTILITY** — The DBMS_UTILITY package is a database administrator's best friend. Through this package, you can do such things as perform a database-wide analysis of all objects and provide the optimizer with current statistics (DBMS_UTILITY.ANALYZE_DATABASE), or limit the analysis to a schema only (DBMS_UTILITY.ANALYZE_SCHEMA), or compile all of the program units in a schema (DBMS_UTILITY.COMPILE_SCHEMA), or do such simple things as get the database version number (DBMS_UTILITY.DB_VERSION), or the current setting of an INIT.ORA parameter (DBMS_UTILITY.GET_PARAMETER_VALUE).
- ♦ **DBMS_SPACE** — The DBMS_SPACE package has only two public procedures: DBMS_SPACE.UNUSED_BLOCKS, which returns the number of unused blocks in a segment, and DBMS_SPACE.FREE_SPACE, which returns that number of free blocks belonging to a specific free list group.
- ♦ **DBMS_ROWID** — The DBMS_ROWID package is designed to allow you to convert between Oracle7 and Oracle8 ROWID formats (DBMS_ROWID.ROWID_TO_EXTENDED and DBMS_ROWID.ROWID_TO_RESTRICTED), as well as extract details of a ROWID (DBMS_ROWID.ROWID_INFO) or specific information such as the object id (DBMS_ROWID.ROWID_OBJECT), file number (DBMS_ROWID.ROWID_RELATIVE_FNO and DBMS_ROWID.ROWID_TO_ABSOLUTE_FNO), or block number (DBMS_ROWID.ROWID_BLOCK_NUMBER) from a given ROWID. It can also be used to generate ROWIDs for testing purposes (DBMS_ROWID.ROWID_CREATE).

- ♦ **DBMS_SHARED_POOL**—The `DBMS_SHARED_POOL` package provides a routine to pin frequently accessed objects, such as procedures, packages, triggers, and types in the shared pool (`DBMS_SHARED_POOL.KEEP`) so that they will not be aged out or to turn off the pinning capability so that these objects can be aged out of the shared pool normally (`DBMS_SHARED_POOL.UNKEEP`). It also allows you to find which objects in the shared pool are larger than a specified size threshold (`DBMS_SHARED_POOL.SIZE`).

As you continue to work through this book, these and other Oracle supplied packages will be introduced to you. The use of each package and its procedures and functions will be dealt with in the chapter that most closely matches the database objects dealt with by the package and its intended purpose.

View stored object information

Once you have created stored objects in Oracle, there are a number of ways to view data dictionary information on them. One method is to make use of DBA Studio and view the schema information, as shown in Figure 5-1.

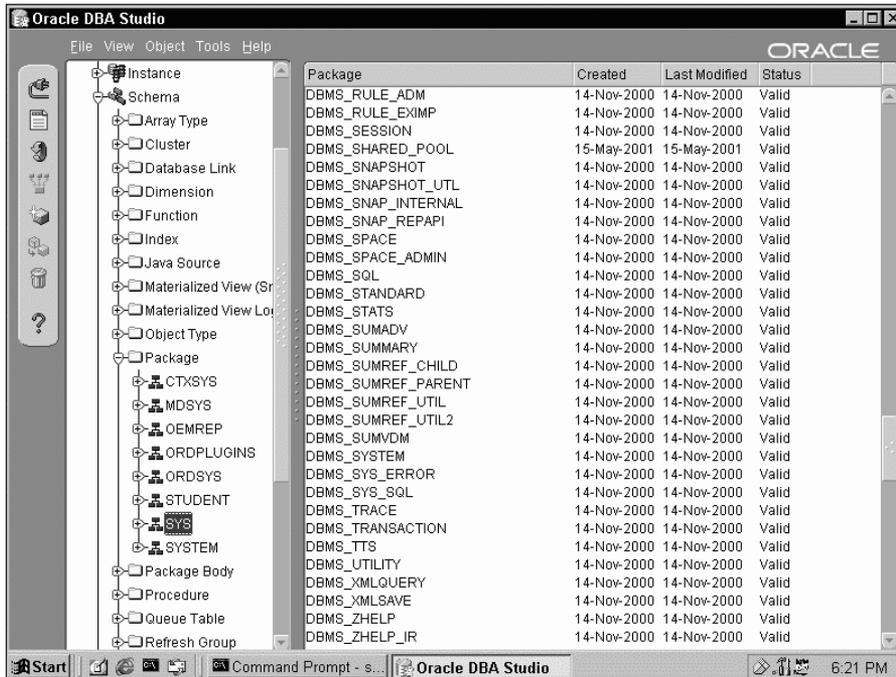


Figure 5-1: DBA Studio allows you to view objects in your database.

Another method is the `DBA_OBJECTS` view, which has a structure as follows:

```
SQL> desc DBA_OBJECTS
Name                                                    Null?    Type
-----
OWNER                                                    VARCHA2(30)
OBJECT_NAME                                              VARCHA2(128)
SUBOBJECT_NAME                                           VARCHA2(30)
OBJECT_ID                                                NUMBER
DATA_OBJECT_ID                                           NUMBER
OBJECT_TYPE                                              VARCHA2(18)
CREATED                                                  DATE
LAST_DDL_TIME                                           DATE
TIMESTAMP                                                VARCHA2(19)
STATUS                                                    VARCHA2(7)
TEMPORARY                                                 VARCHA2(1)
GENERATED                                                 VARCHA2(1)
SECONDARY                                                 VARCHA2(1)

SQL>
```

Of the columns presented, pay special attention to the `STATUS` column. Objects in Oracle can have two possible values for the `STATUS` column: `VALID` and `INVALID`. If the object's status is `VALID`, then it and all of its dependent objects are properly formulated and the object can be invoked without any problems. However, if the object's status is `INVALID`, it does not necessarily mean that the object itself cannot be used. It may simply indicate that, since the last time the object was compiled, a dependent object has changed and a recompilation is necessary to make the object `VALID` again.

For example, the enrollment package, mentioned previously, is dependent upon the `ClassEnrollment` table. If the `ClassEnrollment` table were modified to add a new column, this would invalidate the enrollment package (that is, change its status to `INVALID`).

To make the package valid again, all you, or a user, would need to do is to run one of the procedures in the package. This would force Oracle to recompile the package and bring it to a `VALID` state again. The fact that a new column was added would not cause the package to remain invalid, but dropping a column that the package depends upon may continue to make it invalid until references to the column are removed from the package.

**Tip**

Oracle also includes a view that enables you to track dependencies of Oracle objects. This is the `DBA_DEPENDENCIES` view, and its corresponding `ALL_DEPENDENCIES` and `USER_DEPENDENCIES` views. These can be used to determine which objects another object depends upon. Also, the `UTLDTREE.SQL` utility script in the `ORACLE_HOME\RDBMS\ADMIN` directory has additional routines that can help you track object dependencies.

Creating Data Dictionary Views

After this long discussion on the various Oracle objects and what the data dictionary consists of, you probably are wondering what all this is leading up to. The answer is simple — all of the information presented to this point is to show you the various objects that can be created in Oracle, and to also demonstrate that Oracle itself uses all of these objects to manage any database. In fact, the `CREATE DATABASE` command, as shown in Chapter 4, does nothing to prepare a database for use by you. The `CREATE DATABASE` command gets the ball rolling but you need to create additional objects, such as packages, functions, stored procedures, views, and tables, that Oracle itself uses to manage the database and to make it usable for you as a database administrator. All of those wonderful `DBA_`, `ALL_`, and `USER_` views that were mentioned earlier do not exist in the database after the `CREATE DATABASE` statement is issued — they need to be created.

Scripts to create data dictionary views and packages

Parts of the data dictionary are created by running scripts that are located in the `RDBMS\ADMIN` (for Windows NT/2000) or `rdbms/admin` (for Linux/UNIX) directories in the path where you installed the Oracle software. While many scripts exist, two that are required to create the basic structure of the data dictionary are `CATALOG.SQL` and `CATPROC.SQL`.

The `CATALOG.SQL` Script

The first script that needs to be run after the `CREATE DATABASE` statement successfully completes is the `CATALOG.SQL` script. This script creates the `V$` dynamic data dictionary views, and the `DBA_`, `ALL_` and `USER_` data dictionary views, as well as any other data dictionary views needed such as `SESSION_PRIVS`. It also creates synonyms for many of the views, and creates comments on the various data dictionary tables and columns. As part of the creation process, the `CATALOG.SQL` script also grants the necessary privileges on the objects created so that users and DBAs can each access the views they should be allowed to query. The `CATALOG.SQL` script also configures data types, exceptions, procedures and functions that will later be used to configure PL/SQL support for the database. Finally, support for Import/Export, SQL*Loader, auditing and the basic installed options is also set up.

To run the CATALOG.SQL script, you can use Server Manager line mode (SVRMGRL) or SQL*Plus. You must connect to the instance as a user with SYSDBA privileges, such as INTERNAL. If you are using SVRMGRL, you need to be logged in to the computer where the database was created, and properly configure the ORACLE_SID environment variable, as follows:

```
C:\CERTDB>SET ORACLE_SID=CERTDB

C:\CERTDB>svrmgrl

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SVRMGR> connect internal;
Connected.
SVRMGR>
```

Once you have properly configured the environment, you can execute the CATALOG.SQL script and monitor its progress. The script will not spool its output to a file but if you want to review the results of the script and check for errors, you may want to spool to a file before invoking the script. To spool the output to a file and run the script, issue the following commands:

```
SVRMGR> spool catalog.out
SVRMGR> @%ORACLE_HOME%\RDBMS\ADMIN\CATALOG.SQL;
Statement processed.
Statement processed.
Statement processed.
Statement processed.
drop public synonym v$d1m_misc
      *
ORA-01432: public synonym to be dropped does not exist
Statement processed.
Statement processed.
Statement processed.
drop public synonym v$d1m_latch
      *
ORA-01432: public synonym to be dropped does not exist
Statement processed.
Statement processed.
Statement processed.
drop public synonym v$d1m_convert_local
      *
ORA-01432: public synonym to be dropped does not exist
Statement processed.
Statement processed.
```

...

The output of the CATALOG.SQL script executing has been truncated for reasons of space, but you should know that the script takes five to fifteen minutes to run, depending upon the speed of the machine and other factors. The ORA-01432 errors that you see in the output simply indicate that the script tried to drop an object that did not exist. This is not a problem and occurs the first time the script is run. If you ran the script again, and you can safely do so without causing any harm to the database, you would normally not see these errors.

After the script has completed execution, remember to turn off spooling, as follows:

```
SVRMGR> spool off
SVRMGR>
```

At this point you should be able to query the V\$, DBA_, ALL_ and USER_ views from any session. To test this, you may wish to invoke SQL*Plus and connect to the instance as the user SYSTEM with a password of MANAGER and query the DBA_USERS and USER_USERS views, as follows:

```
SQL> connect system/manager@certdb.mars.bradsys.com
Error accessing package DBMS_APPLICATION_INFO
ERROR:
ORA-06554: package DBMS_STANDARD must be created before using
PL/SQL
```

```
Connected.
SQL> SELECT username FROM DBA_USERS;
```

```
USERNAME
-----
SYS
SYSTEM
OUTLN
```

```
SQL> SELECT username FROM USER_USERS;
```

```
USERNAME
-----
SYSTEM
```

```
SQL>
```

If you had errors when running the CATALOG.SQL script, or connecting to the instance later does not let you query some of the basic views, you can safely re-run the script. This is because the script replaces all data dictionary views that you have in the database when it is run, but does not change the underlying base tables. Your data will still be intact. In fact, if a DBA accidentally dropped one of the DBA_ views, the best way to re-create the missing view is to rerun CATALOG.SQL.


```

Statement processed.
drop public synonym ALL_ERRORS
*
ORA-01432: public synonym to be dropped does not exist
Statement processed.
Statement processed.
Statement processed.
drop public synonym DBA_ERRORS
*
ORA-01432: public synonym to be dropped does not exist
Statement processed.
Statement processed.
Statement processed.
...

```

Similar to the running of CATALOG.SQL, any ORA-01432 errors are normal the first time CATPROC.SQL is run and should not be of any concern. If you are re-running the script, which can also be done safely at any time, you should not see these errors.

To confirm that the scripts worked correctly, you can connect to the instance as SYSTEM with a password of MANAGER using SQL*Plus and verify that the error you received regarding DBMS_APPLICATION_INFO after running the CATALOG.SQL script no longer exists. You can also try creating a small PL/SQL block to ensure that PL/SQL support is functioning properly, as follows:

```

SQL> connect system/manager@certdb.mars.bradsys.com
Connected.
SQL> set serveroutput on
SQL> BEGIN
  2  DBMS_OUTPUT.PUT_LINE('This is a sample PL/SQL block');
  3  END;
  4  /
This is a sample PL/SQL block

PL/SQL procedure successfully completed.

SQL>

```

If you were to review the contents of the CATALOG.SQL and CATPROC.SQL script, you will notice that they also make calls to execute additional scripts using the “@@<scriptname>” syntax. This syntax indicates that the script being called resides in the same location as the script calling it, and that program execution should return to the calling program. While CATALOG.SQL performs most of its work itself, CATPROC.SQL relies upon calls to many other scripts almost exclusively, as shown by this small sample of code from the script:

```
...
Rem basic procedural views
@@catprc
@@catjobq

Rem Remote views
@@catrpc

Rem Setup for pl/sql
@@dbmsstdx
@@plitblm
@@plspur
@@pipidl
rem granting execute on the package created in pipidl.sql
rem to execute_catalog_role...
grant execute on pidl to execute_catalog_role
/
@@pidian
rem granting execute on the package created in pidian.sql
rem to execute_catalog_role...
grant execute on diana to execute_catalog_role
/
@@diutil
@@pistub
@@utlhttp
@@prvtpck1.plb
Rem packages implementing PL/SQL file data type
@@utlfile
@@prvtfile.plb
Rem more PL/SQL I/O packages
@@utlraw
@@prvtrawb.plb
@@utltcp
@@prvttcp.plb
@@utlinad
@@prvtinad.plb
@@utlsmtp
@@prvtsmtp.plb
...
```

Other scripts and data dictionary objects

Running CATALOG.SQL and CATPROC.SQL will not install support for all the features Oracle has. In fact, running these two scripts is designed to install support for the core set of features available in Oracle8i. There are additional scripts in the RDBMS\ADMIN directory that can be run to configure support for more options in Oracle.

If you were to do a listing of files in the RDBMS\ADMIN directory, you would find files categorized with the following filenames:

- ♦ **CAT*.SQL**—Scripts with a CAT prefix in their name are used to create data dictionary views and to support utilities that are beneficial to the user or database administrator. Many of these scripts are run by CATALOG.SQL and CATPROC.SQL, though not all. An example of a script that you may wish to run after the fact is CATADT.SQL, which installs support for object types and many object features within the Oracle database. The reason this script, as well as some others, is not run by CATALOG.SQL or CATPROC.SQL is that the functionality it provides may not be necessary in your database if you are not using object types. Oracle does not want to clutter the data dictionary with information that you do not use.
- ♦ **DBMS*.SQL and PRVT*.PLB**—Scripts create Oracle packages and their bodies that extend the functionality of the Oracle server. An example of a package that is not created by CATALOG.SQL or CATPROC.SQL is the DBMS_SHARED_POOL package. In order to create this package you need to run the DBMSPOOL.SQL script, which will in turn call the PRVTPOOL.PLB script with the wrapped package body. In most cases, DBMS*.SQL scripts will call the appropriate PRVT*.PLB scripts within them, though you can use the PRVT*.PLB scripts to re-create a package body should it become corrupt.

**Tip**

Any script ending in a PLB extension contains wrapped package body code. *Wrapped* package bodies are PL/SQL code that has been put through an Oracle wrapper utility program that converts the source code to alphanumeric data. This is done to prevent end users from seeing the actual source code of the program and is used by Oracle, and other developers, to preserve their intellectual property. For more information on wrapping Oracle packages and other PL/SQL program units, consult the *Oracle8i Application Developers Guide*.

- ♦ **UTL*.SQL**—Scripts whose names begin with UTL are used to provide support for utility functions within the Oracle database. The database administrator can use these scripts to create objects making it easier to find chained rows in a table (UTLCHAIN.SQL), to create the table used by the EXPLAIN PLAN command (UTLXPLAN.SQL) to show the execution plan of a query, or to find out what the dependency tree of a given object is (UTLDTREE.SQL). These scripts are run against the database when needed and are not run by the CATALOG.SQL and CATPROC.SQL scripts.

In general, Oracle only installs support for what you need. The CAT*.SQL, DBMS*.SQL, PRVT*.PLB and UTL*.SQL scripts allow you to add functionality to your database as your needs establish over time. This incremental addition of features ensures that the database is not overburdened by having to keep track of objects that are never used, while providing flexibility to add those features when you need them.

The Next Step

After you have created the database by issuing the CREATE DATABASE command, created the data dictionary views and key objects by running CATALOG.SQL, and installed support for PL/SQL in the server as well as established key features for the database using CATPROC.SQL, the next step is to build your database. In doing so you need to consider storage characteristics for your data, your hardware environment, and your data and its access patterns. A discussion of all of these elements, as well as others, is presented in the upcoming chapters.

Key Point Summary

In preparing for the “Oracle8i DBA: Architecture and Administration” exam, please keep these points regarding data dictionary views, standard packages, triggers, and other PL/SQL program units in mind:

- ♦ The CREATE DATABASE command only establishes the base data dictionary components of a database.
- ♦ CATALOG.SQL is used to create the data dictionary views and CATPROC.SQL is used to create PL/SQL functionality in the database. Both scripts must be run in every Oracle database.
- ♦ CATALOG.SQL, CATPROC.SQL and additional scripts to extend the data dictionary are located in the ORACLE_HOME\RDBMS\ADMIN directory, where ORACLE_HOME represents the Oracle home directory specified when you installed the Oracle software.
- ♦ DBA_ views provide the database administrator with a complete picture of all aspects of the database.
- ♦ USER_ views allow a user to get information on objects and other elements in their schema.
- ♦ ALL_ views allow a user to get information on objects in their schema, and those object in other schemas that they have been given permissions to.
- ♦ V\$ dynamic performance views are used to track performance of the instance and database. Their values are reset every time the instance is started.
- ♦ Stored procedures are program units that perform a particular task. They may have input, output or input/output parameters and can manipulate database data. Stored procedures must be executed off the command line and cannot be used in SQL statements.
- ♦ Functions are program units that perform a particular task and must specify a return value. They typically cannot modify database data and only have input parameters. User-defined functions can be used almost anywhere Oracle functions can.

- ♦ Triggers are program units that automatically fire when an event on which they have been defined takes place. Triggers on DML statements can perform validation of complex business rules and help to ensure database integrity, as well as log user activity. Database event triggers can be used to configure database settings or audit user actions. Triggers cannot be invoked interactively.
- ♦ Packages are a collection of one or more procedures, functions, variables, exceptions, types, or other identifiers grouped under a logical name. They enable you to have program elements often used together that are loaded all at once into the shared pool and thereby improve performance for subsequent executions.
- ♦ Program units in Oracle can be written in PL/SQL or Java. Oracle also supports calls to external procedures written in C, C++, or other supported languages.
- ♦ Oracle uses packages and other program units to extend the functionality of the database. Examples include DBMS_SESSION, DBMS_LOB, DBMS_SPACE, DBMS_ROWID, and others.



STUDY GUIDE

Now that you have learned about what the Oracle data dictionary is composed of, and the roles played by some of the objects available, you should test your understanding by reviewing the assessment questions and performing the exercises below.

Assessment Questions

1. You need to keep track of any DROP statements issued by Bob for objects in his schema. You decide to create a database event trigger. How would you create the trigger? (Choose two correct answers.)
 - A. ON DATABASE
 - B. ON SCHEMA
 - C. AFTER DDL
 - D. AFTER DROP
 - E. FOR EVERY ROW
 - F. INSTEAD OF
2. If John wanted to get the name and owner of all the tables in the database that he has access to, which of the following commands would he issue? (Choose the best answer.)
 - A. `SELECT OWNER, TABLE_NAME FROM ALL_TABLE`
 - B. `SELECT OWNER, TABLE_NAME FROM V$TABLES`
 - C. `SELECT OWNER, TABLE_NAME FROM DBA_TABLES`
 - D. `SELECT OWNER, TABLE_NAME FROM USER_TABLES`
 - E. `SELECT OWNER, TABLE_NAME FROM ALL_TABLES`
3. Which of the following Oracle users owns the data dictionary? (Choose the best answer.)
 - A. SYSTEM
 - B. SYS
 - C. ORACLE
 - D. STUDENT
 - E. SCOTT

4. If you wanted to create a stored program unit that calculates the tax for each enrollment in a course when using a SELECT statement to query the ClassEnrollment table, which type of program unit should you create? (Choose the best answer.)
- A. Stored procedure
 - B. Function
 - C. Package
 - D. Type
 - E. Java method
5. When executing CATALOG.SQL, how must you connect to the instance before invoking the script? (Choose two correct answers.)
- A. INTERNAL
 - B. SYS
 - C. SYSTEM
 - D. SCOTT
 - E. SYS AS SYSDBA
6. Which is not a benefit of putting procedures, functions, and other database objects in a package? (Choose the best answer.)
- A. The entire package is loaded into memory at once when it is first accessed.
 - B. Each user running a package procedure has a separate copy in their PGA.
 - C. Package variables persist for the duration of a session.
 - D. Procedures and functions can be made private.
7. Which of the following does CATALOG.SQL create? (Choose all correct answers.)
- A. V\$ dynamic performance views and synonyms
 - B. EXPLAIN_PLAN table
 - C. DBA_ data dictionary views
 - D. USER_ data dictionary views
 - E. The control files

8. A junior database administrator has accidentally dropped the DBMS_LOB package from the database. Since your database makes heavy use of BLOB columns, the DBMS_LOB package is required by your users. How would you re-create the DBMS_LOB package while keeping all existing data intact? (Choose the best answer.)
- A. Re-run the SQL.BSQ script.
 - B. Re-run the DBMSLOB.SQL script.
 - C. Re-run the CATPROC.SQL script.
 - D. Re-run the CATALOG.SQL script.
 - E. Re-create the database.
9. You recently were hired as a junior database administrator for a large Oracle shop. After coming in early one morning and reviewing the data dictionary information in one of the database, you notice that a number of objects shown by the DBA_OBJECTS view have a status of INVALID. Indexes for your key tables are re-created each night. What should you do to resolve the problem? (Choose the best answer.)
- A. DROP and then CREATE the INVALID objects.
 - B. ALTER the objects and change their state to VALID.
 - C. Issue a RECREATE command for each object.
 - D. Shutdown and re-start the instance.
 - E. Do nothing.
10. What two components make up a package definition? (Choose two correct answers.)
- A. Package procedure
 - B. Package specification
 - C. Package function
 - D. Package variable
 - E. Package body
 - F. User-defined type

Scenarios

1. You have been asked to review the configuration of a database on one of your servers. You connect to the instance as `INTERNAL` and attempt to query the contents of the data dictionary. Your results are the following:
 - Querying `V$TABLESPACE` returns only the `SYSTEM` tablespace.
 - Querying the `STATUS` column of `V$INSTANCE` indicates that the instance is `OPEN`.
 - Attempts to query any `DBA_` views return errors stating that the object does not exist.

Why are you not able to query the `DBA_` views and how do you solve the problem?

2. Your database consists of all the tables and other database objects to support your order entry system. The system is accessed by a Web server where users can enter orders through your Web site, as well as by 200 order entry clerks. The order entry component consists of some 30 stored procedures and functions. Occasionally users complain that a certain portion of the system takes a long time to load while at other times it works fine. What can you do to ensure consistent performance?
3. Occasionally, you find that some objects in your database disappear for no apparent reason. Other objects are still around but their data is no longer visible. Reviewing the alert log file reveals no apparent reason for this to be happening. What can you do to find the cause of the error?

Lab Exercises

Lab 5-1 Creating Data Dictionary Views

1. Open a command prompt (for Windows) or shell window (for UNIX) on your machine and change directories until you are in the location where you created the `CERTDB` database in Chapter 4.
2. Set the `ORACLE_SID` environment variable to the name of your instance, typically `CERTDB`.
3. Invoke Server Manager line mode (`svrmgrl`) and connect to the instance as `INTERNAL` with whatever password you specified when you created the database (typically “oracle”).
4. Verify that the instance is an `OPEN` state and that you are connected to the `CERTDB` database.
5. Attempt to query the `DBA_USERS` view? What happens? Why?
6. Configure Server Manager line mode so that the result of all actions is spooled to a file called `CATALOG.OUT` in the current directory.

7. Invoke the CATALOG.SQL script from the appropriate location.
8. Attempt to query the DBA_USERS view now. What happens? Why?
9. Turn off spooling and review the CATALOG.OUT spool file for any abnormal errors (that is, ignore the ORA-01432 errors).
10. Exit Server Manager line mode.

Lab 5-2 Installing PL/SQL Support in the Database

1. Invoke Server Manager line mode (svrmgrl) and connect to the instance as INTERNAL with whatever password you specified when you created the database (typically “oracle”).
2. Verify that the instance is an OPEN state and that you are connected to the CERTDB database. If you are not connected to the CERTDB instance, exit Server Manager line mode and configure the ORACLE_SID environment variable appropriately, and then try again.
3. Configure Server Manager line mode so that the result of all actions is spooled to a file called CATPROC.OUT in the current directory.
4. Invoke the CATPROC.SQL script from the appropriate location.
5. Turn off spooling and review the CATPROC.OUT spool file for any abnormal errors (that is, ignore the ORA-01432 errors).
6. Exit Server Manager line mode.

Answers to Chapter Questions

Chapter Pre-Test

1. The DBA_ views are created when the CATALOG.SQL script is executed. Prior to executing this script, while connected as a user with the SYSDBA privilege, no DBA_, ALL_, or USER_ views exist in the database.
2. No, after the CREATE DATABASE command has been issued, PL/SQL support does not exist in an Oracle database. In order to install support for PL/SQL in the database, the database administrator must run the CATPROC.SQL script, which will install and configure PL/SQL support.
3. A package is a collection of one or more procedures, functions, types, variables, and other program units that are grouped together under a single logical name. A package is always composed of a package specification, which outlines the published components that can be invoked and their parameter, and a package body, that has the actual code for the procedures, functions, type methods, as well as (optionally) private procedures, functions, and components that are used by the package.

4. Oracle8i supports stored procedures, functions, object types and methods, and packages as program units.
5. PL/SQL and Java can be used to create stored procedures natively in Oracle8i. Support also exists for external procedures to be created using C or C++, or other languages provided that they properly expose their constructs for use within Oracle.
6. Triggers are program units, developed in PL/SQL or Java, that are automatically invoked when the event on which the trigger is defined occurs. Triggers can be defined for DML events (INSERT, UPDATE, DELETE) or database events (LOGON, LOGOFF, SHUTDOWN, STARTUP, ALTER, DROP, and so on). DML triggers can be configured to be invoked once for the statement or once for each row to which the DML applies. Database event triggers can be configured at the database or schema levels.
7. A function is a program unit that must return a value, and may take input parameters. A stored procedure is a program unit that does not return a value and may have input, output or input/output parameters. Both functions and stored procedures can be written in PL/SQL or Java.
8. The four types of scripts that can be used to extend the capabilities of your database include the CAT*.SQL, DBMS*.SQL, PRVT*.PLB, and UTL*.SQL scripts. CAT*.SQL scripts extend the data dictionary by creating data dictionary views. DBMS*.SQL and PRVT*.PLB scripts create additional objects in the data dictionary including packages, procedures, functions and other tables and views. UTL*.SQL scripts provide for additional utilities in the database that can be used to track chained rows or determine the execution of a SQL statement.
9. The DBMS_LOB package provides procedures and functions to manipulate columns of BLOB, CLOB, NCLOB, and BFILE datatypes. In order to manipulate a BLOB column, you need to make use of these procedures and functions.
10. To change your NLS territory setting for your session in Oracle8i, you can use the ALTER SESSION SET NLS_TERRITORY= command, or the DBMS_SESSION.SET-NLS package procedure.

Assessment Questions

1. **B, D.** Since you want to keep track of any drop statements issued by Bob on his own schema, the easiest way to do so is to create a database event trigger on his own schema to fire after the DROP command has been issued. You could also create a trigger ON DATABASE and then check to see which user is performing the action within the trigger code, but this would be less efficient as the trigger would fire for every DROP statement issued by any user.
2. **E.** If John wanted to get a list of all tables he has access to, he would query the ALL_TABLES view. If he were a DBA, he could also query the DBA_TABLES view, but you do not know this so don't assume that John is a DBA. The USER_TABLES view would let him see which tables he owns, but not other tables he has access to. There are no ALL_TABLE or V\$TABLES views in an

Oracle database by default. In fact, ALL_ views, as well as DBA_, and USER_ views, are almost always plural.

3. **B.** The data dictionary is owned by the user SYS. The user SYSTEM has been given permissions to view all data dictionary objects but does not own the data dictionary.
4. **B.** A user-defined function can be used in a SELECT statement and can be passed the price of the course as a parameter in order to calculate the tax. You cannot use a procedure within the column list of a SELECT statement. Packages do not exist by themselves and always include procedures, functions, or others objects.
5. **A, E.** When executing CATALOG.SQL, you must be connected to the instance as a user with SYSDBA privileges. INTERNAL is a connection as the user SYS with SYSDBA privileges. A connection as any Oracle user that has been granted the SYSDBA role will work, although only answer A and E qualified in this case.
6. **B.** One of the benefits of packages is that they can be loaded into the shared pool the first time any part of the package is referenced. Making copies of each procedure in the package in the user's PGA runs counter to this idea and is therefore the incorrect answer. Package variables will persist for the duration of a user's session, thereby acting as global variables once initialized, and parts of the package's code can be made private by creating them in the package body.
7. **A, C, D.** The CATALOG.SQL script creates data dictionary views, including V\$ dynamic performance views, DBA_, ALL_, and USER_ views. The EXPLAIN_PLAN table is created by running the UTLXPLAN.SQL script, while the control files are created by executing the CREATE DATABASE or CREATE CONTROLFILE command.
8. **C.** Re-running the CATPROC.SQL script re-creates the DBMS_LOB package. You can also re-create the package by running the DBMSLOB.SQL and PRVTLOB.PLB scripts — running DBMSLOB.SQL by itself will not create the package body and would not get everything back as needed.
9. **E.** Chances are that the reason some objects have a status of INVALID is that the index re-creation has caused references to objects they depend upon to be broken. This is not necessarily a problem as Oracle re-compiles all dependent objects the next time they are referenced and change the status back to VALID. The best course of action in this case is to do nothing. Dropping and then creating the objects would remove all permissions; you cannot alter an object's status to VALID — this is not an option for the ALTER command; there is no RECREATE command in Oracle8i; and, shutting down and restarting the instance will not change the object's status.
10. **B, E.** A package is made up of a package specification that outlines the publicly accessible parts of the package, as well as any parameters for procedures and functions, and a package body, which includes the actual code for the public procedures and functions, as well as any objects private to the package. A package can contain only a specification, if all it has is variables, but this is not normally done, therefore answers B and E are correct (besides, the question asked for two parts).

Scenarios

1. The most likely reason that you are not able to query the DBA_ views is that they do not yet exist in the database. Since you connected to the instance as INTERNAL, this is not a permissions issue because INTERNAL has full access to the database. The only possible option is that CATALOG.SQL has not yet been run. It is interesting to note that some V\$ views already exist in the database, though CATALOG.SQL also creates the rest and adds synonyms for the basic ones.
2. This scenario is crying out for a package. Because you have a well-defined set of procedures and functions that make up the order entry portion of the database, you should create a package that includes all 30 of the program units. You then need to modify the source code for the Web-based and user-based order entry client application to make use of the package procedures and functions. Creating a package ensures that the first time any component of the package is loaded, the entire package goes into the shared pool, thereby making it unlikely that it will be flushed from memory. The current problem is being caused by some of the procedures and functions being aged out of the shared pool due to lack of space. It is also possible to solve some of the problem by increasing the amount of memory in the shared pool, or pinning the package or procedures in memory using the DBMS_SHARED_POOL.KEEP package procedure.
3. To find if the cause of the disappearing objects is user action, you can create a database event trigger on the DROP event and log who dropped what objects to a table. You can then review the contents of the table and find out who the guilty party is. You should create the trigger ON DATABASE and AFTER DROP. This is because database event triggers cannot prevent the action from taking place but only record that it happened.

Lab Exercises

Lab 5-1

1. Open a command prompt (for Windows) or shell window (for UNIX) on your machine and change directories until you are in the location where you created the CERTDB database in Chapter 4.

```
C:\>CD C:\CERTDB (for Windows)
```

```
cd /CERTDB (for UNIX)
```

2. Set the ORACLE_SID environment variable to the name of your instance, typically CERTDB.

```
C:\CERTDB>SET ORACLE_SID=CERTDB (for Windows)
```

```
$ ORACLE_SID=CERTDB; export ORACLE_SID (for UNIX)
```

- 3. Invoke Server Manager line mode (svrmgrl) and connect to the instance as INTERNAL with whatever password you specified when you created the database (typically “oracle”).**

```
C:\CERTDB>svrmgrl

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SVRMGR> connect internal;
Connected.
SVRMGR>
```

- 4. Verify that the instance is an OPEN state and that you are connected to the CERTDB database.**

```
SVRMGR> SELECT instance_name, status FROM V$INSTANCE;
INSTANCE_NAME      STATUS
-----
certdb              OPEN
1 row selected.
SVRMGR>
```

- 5. Attempt to query the DBA_USERS view? What happens? Why?**

```
SVRMGR> SELECT user_name FROM DBA_USERS;
SELECT user_name FROM DBA_USERS
*
ORA-00942: table or view does not exist
SVRMGR>
```

The query fails because the DBA_ views are not yet created in the database.

- 6. Configure Server Manager line mode so that the result of all actions is spooled to a file called CATALOG.OUT in the current directory.**

```
SVRMGR> spool CATALOG.OUT;
```

- 7. Invoke the CATALOG.SQL script from the appropriate location.**

```
SVRMGR>>@ORACLE_HOME%\RDBMS\ADMIN\CATALOG.SQL (for Windows)

SVRMGR>@$ORACLE_HOME/rdbms/admin/catalog.sql (for UNIX)
```

- 8. Attempt to query the DBA_USERS view now. What happens? Why?**

```
SVRMGR> SELECT username FROM DBA_USERS;
USERNAME
-----
SYS
SYSTEM
```

```

OUTLN
3 rows selected.
SVRMGR>

```

The query succeeds and returns the default users installed at database creation time (SYS and SYSTEM), as well as the OUTLN user created by the CATALOG.SQL script.

9. Turn off spooling and review the CATALOG.OUT spool file for any abnormal errors (i.e. ignore the ORA-01432 errors).

```
SVRMGR> spool off;
```

10. Exit Server Manager line mode.

Lab 5-2

1. Invoke Server Manager line mode (svrmgrl) and connect to the instance as INTERNAL with whatever password you specified when you created the database (typically “oracle”).

```
C:\CERTDB>svrmgrl
```

```
Oracle Server Manager Release 3.1.7.0.0 - Production
```

```
Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.
```

```
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production
```

```
SVRMGR> connect internal;
Connected.
SVRMGR>
```

2. Verify that the instance is an OPEN state that you are connected to the CERTDB database. If you are not connected to the CERTDB instance, exit Server Manager line mode and configure the ORACLE_SID environment Variable appropriately and then try again.

```
SVRMGR> SELECT instance_name, status FROM V$INSTANCE;
INSTANCE_NAME      STATUS
-----
certdb              OPEN
1 row selected.
SVRMGR>
```

3. Configure Server Manager line mode so that the result of all actions is spooled to a file called CATPROC.OUT in the current directory.

```
SVRMGR> spool CATPROC.OUT;
SVRMGR>
```

4. Invoke the CATPROC.SQL script from the appropriate location.

```
SVRMGR>%ORACLE_HOME%\RDBMS\ADMIN\CATPROC.SQL (for Windows)
```

```
SVRMGR>${ORACLE_HOME}/rdbms/admin/catproc.sql (for UNIX)
```

5. Turn off spooling and review the CATPROC.OUT pool file for any abnormal errors (i.e. ignore the ORA-01432 errors).

```
SVRMGR> spool off;
```

6. Exit Server Manager line mode.

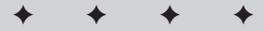
Maintaining the Control File

EXAM OBJECTIVES

- ◆ Maintaining the Control File
 - Explain the uses of the control file
 - List the contents in the control file
 - Multiplex the control file
 - Obtain control file information

6

CHAPTER



CHAPTER PRE-TEST

1. What kind of information is kept in the control file?
2. How many copies of the control file should exist and where should they be located?
3. When is control file information updated?
4. Name at least three views that obtain information from the control file?
5. Can control files grow dynamically? If so, which initialization parameters can affect control file growth?
6. How do you multiplex a control file?

♦ Answers to these questions can be found at the end of the chapter. ♦

This chapter outlines how to manage one of the most critical components of an Oracle8i database—the control file. The first step in knowing how to manage any component of Oracle8i is to understand what it is used for and how it works. I cover this first, followed by what kind of information is kept in the file and why this makes it such a key component of the database. Then you find out how to safeguard the control file from possible harm, as well as how to protect yourself from the possible loss of the file. Finally, you learn how to determine the contents of the control file and which data dictionary views are used to query the contents, as well as which data dictionary views return their information from the control file.

If you needed to name one of the most critical files that make up an Oracle database, you should pick the control file. It is as close to an Achilles heel as Oracle has. Losing a control file could mean losing data if all the information that is needed to re-create it is not available. It is the one file that, above all others, should be protected completely.

Overview of the Control File

Objective

Explain the uses of the control file

Every Oracle8i database has at least one control file, a small binary file that is required to start an Oracle instance. It contains information on the location of key critical database files. Oracle reads the control file during the mount phase of starting an Oracle instance. If any copy of the control file cannot be read at that time, whether you are using one or more files, the instance cannot be mounted and the database cannot be opened. Simply put, the control file *controls* whether or not an instance can be opened and made available to users.

The control file is updated continuously by the checkpoint process (CKPT) to ensure that its contents are current. This is because the control file is also used to determine what the last update is that occurred on a datafile, and when it happened. By having this information in the control file, it is possible to recover up to the point of failure and not lose any data should a datafile become unavailable due to the loss of a disk, or some other failure. The control file is a critical element in the recovery of Oracle data, as it knows what took place with each datafile last.

The contents of the control file cannot be changed by the DBA, or any user of the Oracle database. The file itself is stored in a binary format that only Oracle is able to read and write. For this reason, no other maintenance is required or available except to ensure that at least one copy of the control file is available to Oracle.

Control File Contents

Objective

List the contents of the control file

Simply said, the control file contains information about all other critical database files. For the most part, the number of entries for each specific element (such as datafiles, redo log members, and so forth) is fixed at the time of control file creation. In other words, the control file starts out with a static size based upon the options specified when it was created.

The information that can be found in the control file includes:

- ♦ **The name of the database.** When a database is created, its name is recorded in the control file. If you need to change the name of the database, you cannot do so without re-creating the control file. For this reason, it is important to be sure the database name you specify in the Oracle initialization file with the `DB_NAME` parameter or in the `CREATE DATABASE` statement is one that you can live with.
- ♦ **The database identifier.** Every Oracle database has an internal identifier that is generated when the database is created. The database identifier, or database id, is also stored in the control file.
- ♦ **The date and time the database was created.** The control file also records that date and time of database creation. This is rarely used but can be used in database recovery.
- ♦ **The archiving mode of the database.** Whether or not the database is in `ARCHIVELOG` or `NOARCHIVELOG` mode is also recorded in the database. Again, this plays a part in the recoverability characteristics of the database.
- ♦ **The names and locations of the datafiles for the database.** Every datafile belonging to tablespaces in the database is recorded in the control file, along with the last time these files were updated and the last system change number (SCN) or transaction written to the datafile. The number of datafiles about which information can be kept is fixed at the time the control file was created. In other words, if you specified that the maximum number of datafiles you want to have in a database is 30, this creates thirty slots or records in the control file to hold this information. If you want to have more datafiles in the database, you need to re-create the control file.
- ♦ **The names and locations of all redo log file groups and their members.** The control file also keeps track of the last log switch number and SCN for each log group. The status of log members (that is, whether or not the files are valid) is also maintained. This information is also used to ensure that up-to-date information is available for recovery. The number of log members and log groups that a database may have is determined when the log file is created. This is similar to what happens for datafiles — the number of available records is determined at creation time and cannot be changed.

- ♦ **The location and status of archived log files.** The specific number of entries is also determined at control file creation time. Information about what is contained in each archive redo log file is entered into the control file when logs are archived. This also helps in recovery.
- ♦ **Backup information.** When Recovery Manager (RMAN) is used to perform backups, information is placed in the control file about what was backed up and where it is located. The number of backup entries is dynamic and may cause the control file to grow over time. The time period for which backup information is kept is specified in days and is determined by the Oracle initialization parameter `CONTROL_FILE_RECORD_KEEP_TIME`. The default is seven days, although this value can be adjusted by the DBA if needed. Because information on which datafiles were backed up and when is part of the backup history, and the number of datafiles can differ between databases, it is hard to predict exactly how large the control file can become.

When looking at the types of entries that exist in the control file, they can be broadly considered to be reusable and non-reusable. The fixed entries storing information on the database structure (that is, datafiles, redo log files and archive log files) are determined when the control file is created. The non-reusable entries are those placed into the control file by Recovery Manager (RMAN) as backups take place. How these are used and created is covered in more detail in the Backup and Recovery book.

Creating the Control File

You can create the control file by using one of two Oracle SQL statements:

```
CREATE DATABASE  
CREATE CONTROLFILE
```

The `CREATE DATABASE` statement is covered in Chapter 4. The `CREATE CONTROLFILE` statement is used to re-create the control file if all copies of the control file are invalid or to make changes in the structure of the control file, such as increasing the number of datafiles that a database may have.

Whether the `CREATE DATABASE` or `CREATE CONTROLFILE` statements are used, the non-reusable portions of the control file are created. Parameters of either statement that determine the non-reusable portions of the control file include:

<code>MAXLOGFILES</code>	The maximum number of log file groups that can exist in the database
<code>MAXLOGMEMBERS</code>	The maximum number of redo log file members per log file group that the database may have
<code>MAXDATAFILES</code>	The maximum number of datafiles that the database may have

MAXLOGHISTORY	The maximum number of archive redo log files to be kept track of in the control file
MAXINSTANCES	The maximum number of instances that are allowed to connect to the database

If any of the above parameters need to be changed, you need to re-create the control file. These entries are fixed and cannot be changed once the control file is created, except by re-creating it.

Another optional parameter that is important when creating the database and/or control file is the `CONTROLFILE REUSE` parameter. This option to the `CREATE DATABASE` and `CREATE CONTROLFILE` syntax tells Oracle not to generate an error if a file with the same name already exists when the command is issued. Oracle will not overwrite an existing disk file and this optional parameter is needed if you are re-creating the control files to increase the number of records contained therein or to change the database name.

You will notice by looking at the `CREATE DATABASE` syntax, as well as the `CREATE CONTROLFILE` syntax (presented later in this chapter), that the location of the control file is not specified in any Oracle command. In fact, the location of the control file is only specified by the Oracle initialization parameter `CONTROL_FILES`. An example initialization file is as follows:

```
db_name = "ORCL"
instance_name = ORCL
service_names = ORCL
db_files = 1024 # INITIAL

control_files = ("C:\Oracle\oradata\ORCL\control01.ctl",
"D:\Oracle\oradata\ORCL\control02.ctl",
"E:\Oracle\oradata\ORCL\control03.ctl")

open_cursors = 100
max_enabled_roles = 30
db_file_multiblock_read_count = 8 # INITIAL
```

Notice that it is possible to specify more than one control file using the `CONTROL_FILES` parameter. When you create a database or issue the `CREATE CONTROLFILE` command (both of which can only be done in a `NOMOUNT` state for the instance), Oracle reads the contents of the parameter, as well as the options (above) specified in the command, and creates all control files identified in the parameter file.

Protecting the Control File

The best way to protect the control file is to ensure that more than one copy exists, and each copy is stored on a separate physical disk. This can be done by specifying

more than one control file in the Oracle initialization file before issuing commands that would create the files. If this is done at database creation time, all files will be created at that time; if not, you will need to make additional copies manually.

Multiplexing the control file

Multiplexing the control file simply means having more than one copy of the file, preferably on separate physical disks. This can most easily be accomplished by specifying more than one filename in the `CONTROL_FILES` parameter in the Oracle initialization file. When doing this, it is important to ensure that the physical path where the file will be stored already exists — Oracle will not create it by default.



Tip

When using Oracle Database Configuration Assistant to create your database, multiple control files are created automatically by default. However, they are all created on the same disk drive and need to be moved. The Database Configuration Assistant also creates the necessary paths to store the files.

If you need to create additional control files in order to multiplex them, follow these steps:

STEP BY STEP: Multiplexing the Control File

1. Shut down the Oracle instance using the `SHUTDOWN` command. In order to shut down the instance, you must be connected as a user with `SYSDBA` privileges.
2. After the database is shut down, copy the current control file (or one of them, if you already have more than one), to the new location. You can specify a different file name for the file from the original. It is also recommended that you copy the control file to a different physical hard disk from the original. This will help protect you in the case of disk failure.

If you are running your Oracle database on a UNIX machine, make sure the UNIX group that owns the Oracle software has read and write permissions on the new control file copy, otherwise Oracle will not be able to open the file and the instance will not start. If you are running Windows NT or Windows 2000, make sure the Administrators local group and the SYSTEM system group has full control of the new control file copy.



Caution

Do not copy the control file while the Oracle instance is started. The file will not be usable.

3. Modify the Oracle initialization file (that is, the `INIT<SID>.ORA` file) to include the name and location of the new control file in the `CONTROL_FILES` parameter. Ensure that you include the full path and correct filename in the Oracle initialization file and separate each control file copy in the Oracle initialization file by a comma. Oracle will not allow you to start the instance if there are any errors in the `CONTROL_FILES` parameter, such as a misspelling of the filename or path.

4. Start up the Oracle instance after connecting as a user with SYSDBA privileges. As I mentioned previously, the instance will not start if the CONTROL_FILES parameter is not properly configured. If you do receive an error, shut down the instance. Verify the INIT<SID>.ORA entries. Make any necessary corrections, and then start the instance again.

Each of the above steps is critical to properly multiplex the control file. If you forget to modify the Oracle initialization file after having copied the control file to another location, and then re-start the instance, you need to repeat the process. This is because each time the instance is started, the control file is updated and the old copy is now out of date. Oracle will not start unless all control files are identical.



Tip

Moving the control file is a similar process to multiplexing, with the exception that you need to move and not copy the file. You still need to modify the Oracle initialization file to specify the new location, as well as shut down the instance before moving the file, and re-start it after modifying the Oracle initialization file.

Backing up the control file

Even though it is a critical file, it is possible to make backup copies of the control file structure or the control file as a whole. To make a copy of the control file in its binary form while the Oracle instance is started, assuming you are connected as a user with the SYSDBA privileges, you would need to issue the command:

```
ALTER DATABASE BACKUP CONTROLFILE TO <location>
```

where <location> is the full path, including the filename, of the backup copy of the control file you wish to create.

If you did not want to make a binary copy of the control file, but simply wanted to create the necessary Oracle commands to re-create the control file, you can issue the following command:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE
```

This command creates a trace file in the path pointed to by the Oracle initialization parameter USER_DUMP_DEST. The contents of this trace file include the necessary code to re-create the control file if it is lost and looks similar to the following example:

```
# The following commands will create a new control file and use it
# to open the database.
# Data used by the recovery manager will be lost. Additional logs may
# be required for media recovery of offline datafiles. Use this
# only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "ORCL" NORESETLOGS NOARCHIVELOG
MAXLOGFILES 32
```

```

MAXLOGMEMBERS 2
MAXDATAFILES 32
MAXINSTANCES 16
MAXLOGHISTORY 1815
LOGFILE
GROUP 1 'E:\ORACLE\ORADATA\ORCL\REDO03.LOG' SIZE 1M,
GROUP 2 'E:\ORACLE\ORADATA\ORCL\REDO02.LOG' SIZE 1M,
GROUP 3 'E:\ORACLE\ORADATA\ORCL\REDO01.LOG' SIZE 1M
DATAFILE
'E:\ORACLE\ORADATA\ORCL\SYSTEM01.DBF',
'E:\ORACLE\ORADATA\ORCL\RBS01.DBF',
'E:\ORACLE\ORADATA\ORCL\USERS01.DBF',
'E:\ORACLE\ORADATA\ORCL\TEMP01.DBF',
'E:\ORACLE\ORADATA\ORCL\TOOLS01.DBF',
'E:\ORACLE\ORADATA\ORCL\INDX01.DBF',
'E:\ORACLE\ORADATA\ORCL\DR01.DBF',
'E:\ORACLE\ORADATA\ORCL\OEMREPO1.DBF'
CHARACTER SET WE8ISO8859P1
;
# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
# Database can now be opened normally.
ALTER DATABASE OPEN;
# No tempfile entries found to add.
#

```

You should backup the control files (preferably both a binary and a trace copy) whenever the database structure changes. This means when you add a new datafile, log file member, or log file group, as well as when you change the location of any of these files using the Oracle command `ALTER DATABASE RENAME FILE`. If you delete a datafile or any other Oracle database file, you should also create a backup copy of the control file. In this way you will always have a way to recover the current structure of the database, should all control files be lost.



Backing up the control file and how to make use of the backup copies is not tested by the “Oracle8i: Architecture and Administration” exam. For more information on how to perform these tasks, consult the *Oracle8i Backup and Recovery Guide* in the Oracle documentation set.

Displaying Information about the Control File

Oracle provides two primary views to get information about the control files for a database. They are `V$CONTROLFILE`, which lists the control files associated with a database, and `V$CONTROL_FILE_RECORD_SECTION`.

To determine which control files are currently active and in use, while the instance is in a mount of open state issue the command

```
SELECT * FROM V$CONTROLFILE
```

This returns the name and location of each control file in use, as follows:

```

STATUS  NAME
-----
        E:\ORACLE\ORADATA\ORCL\CONTROL01.CTL
        E:\ORACLE\ORADATA\ORCL\CONTROL02.CTL
        E:\ORACLE\ORADATA\ORCL\CONTROL03.CTL
3 rows selected.

```

You can also query the V\$PARAMETER view or the Server Manager command SHOW PARAMETER, which returns the currently running Oracle initialization parameters for the value of the CONTROL_FILES parameter. However, when using either of these, the output may be truncated and may not provide a complete listing of all control files for the database. For this reason, use V\$CONTROLFILE.

As you learned earlier in this chapter, the control file is made up of a number of records detailing the structure of the database. To find out how many records are available, used, and the size of each record, you can query the V\$CONTROL_FILE_RECORD_SECTION view. To do so, issue the following command:

```
SELECT * FROM V$CONTROL_FILE_RECORD_SECTION
```

It will display a row for each record type, as well as the total number available and used, as in the following sample output:

TYPE	RECORD_SIZ	RECORDS_TO	RECORDS_US	FIRST_INDE	LAST_INDEX	LAST_RECID
DATABASE	192	1	1	0	0	0
CKPT PROGRESS	4084	16	0	0	0	0
REDO THREAD	104	16	1	0	0	0
REDO LOG	72	32	3	0	0	3
DATAFILE	180	32	8	0	0	3
FILENAME	524	97	11	0	0	0
TABLESPACE	68	32	8	0	0	3
RESERVED1	56	32	0	0	0	0
RESERVED2	1	1	0	0	0	0
LOG HISTORY	36	1815	29	1	29	29
OFFLINE RANGE	56	145	0	0	0	0
ARCHIVED LOG	584	1608	0	0	0	0
BACKUP SET	40	204	0	0	0	0
BACKUP PIECE	736	66	0	0	0	0
BACKUP DATAFILE	116	70	0	0	0	0
BACKUP REDOLOG	76	107	0	0	0	0
DATAFILE COPY	660	74	0	0	0	0
BACKUP CORRUPTION	44	185	0	0	0	0
COPY CORRUPTION	40	204	0	0	0	0
DELETED OBJECT	20	1633	0	0	0	0
PROXY COPY	852	134	0	0	0	0
RESERVED4	1	8168	0	0	0	0

22 rows selected.

As you may recall from the earlier discussion on how the control file is created, the number of available records for each type is determined when the control file is created. If you find that you are running out of a particular record type—for example, the number of datafile records is almost exhausted—you will need to re-create the control file by using the `CREATE CONTROLFILE` command.

Views using the control file

A number of Oracle V\$ views make use of the control file to return their results. These include the following key V\$ views:

V\$BACKUP	Returns the backup state of datafiles. The backup state is kept in the control file and determines whether or not the instance can be opened. If a file is in a backup state, the backup must be ended before the instance can be shutdown, or the file in the middle of a backup will need to be taken offline.
V\$DATAFILE	Returns the name of each datafile in the database, as well as the tablespace number it belongs to. To see which files belong to which tablespace, join this view to the V\$TABLESPACE view, or use the DBA_DATA_FILES view after the instance is opened.
V\$TABLESPACE	Returns the name and number of each tablespace in the database. To find out which files belong to the tablespace, join this view to the V\$DATAFILE view on the TS# column.
V\$TEMPFILE	Returns the name and location of any temporary files used by the database.
V\$ARCHIVE	When archiving is taking place, returns information on the file being archived and the status of archiving.
V\$LOG	Returns the log file groups in existence in the database.
V\$LOGFILE	Returns the list of redo log files defined in the database, which log file group they belong to, the file state, and the SCNs recorded in each file.
V\$LOGHIST	Returns the beginning and ending change number of the contents of redo log files. This view is populated whether or not archiving is enabled and simply shows what took place for each log switch.
V\$ARCHIVED_LOG	Returns the last set of archived redo log files and their contents.
V\$DATABASE	Contains general information about the database including the database name, ID, creation date, last checkpoint, current SCN, archiving state, and others.

Key Point Summary

In preparing for the “Oracle8i DBA: Architecture and Administration” exam, please keep these points regarding maintaining control files in mind:

- ♦ The control file is a critical file for the operation of any Oracle database. The Oracle instance cannot start unless the control file and all other key files belonging to the database are in sync. If a control file does not exist, it needs to be created using the CREATE CONTROLFILE command.
- ♦ Control files should be multiplexed; that is, multiple copies of the control file should exist on separate physical disks. This helps protect you in case of disk failure by ensuring that at least one copy of the control file is available, thereby allowing you to recover up to the point of failure.
- ♦ The control file contains reusable and non-reusable sections. The reusable sections are used by Oracle Recovery Manager.
- ♦ The length of time information is kept in the control file is specified in days and determined by the Oracle initialization parameter CONTROLFILE_RECORD_KEEP_TIME. This part of the control file can grow and cause the control file to grow if the value is increased.
- ♦ The size of the non-reusable record section of the control file is determined at creation time. If you need to increase the MAX number of datafiles or redo log groups that your database will have, you need to re-create the control file. For this reason, it is important that you specify a sufficient number of datafiles, redo logs, and redo log members at creation time — re-creating a control file can be time consuming and sometimes difficult.
- ♦ The control file is a potential Achilles heel in Oracle and should be backed up whenever your database structure changes. You should create both a binary copy as well as a trace copy to ensure that you can recover from all potential failures.



STUDY GUIDE

Now that you have learned about the Control File, you should test your understanding by reviewing the assessment questions and performing the exercises below.

Assessment Questions

1. When an instance is started, from which file does Oracle determine where the control file is located and how many control files to open? (Choose the best answer.)
 - A. Archived Redo Log File
 - B. System Tablespace Datafile
 - C. Listener Control File
 - D. Oracle Initialization File
 - E. LOGIN.SQL File
2. Information about which of the following is not stored in the control file? (Choose the best answer.)
 - A. Oracle Datafiles
 - B. Redo Log Files
 - C. Clusters
 - D. Backups
 - E. Archive Log Status
3. The best way to protect the control file from loss or failure due to system problems is? (Choose the best answer.)
 - A. Backup the control file every time your database structure changes.
 - B. Multiplex the control file on the same physical disk as the Oracle software.
 - C. Create multiple backups of the control file daily.
 - D. Backup the control file to trace on a regular basis.
 - E. Multiplex the control file on each physical disk on the machine.

4. To determine how many more datafiles your control file can support, you could issue which of the following queries? (Choose the best answer.)
 - A. `SELECT * FROM V$CONTROLFILE_RECORD_SECTION`
 - B. `SELECT * FROM V$CONTROLFILE_ENTRIES`
 - C. `SELECT * FROM V$CONTROLFILE`
 - D. `SELECT * FROM V$DATAFILE`
 - E. `SELECT * FROM DBA_CONTROL_FILE_ENTRIES`

5. You want to increase the length of time backup information is maintained in the control file from 7 days to 14 days. Which of the following Oracle initialization settings would you place in your `INIT.ORA` file? (Choose the best answer.)
 - A. `BACKUP_HISTORY=14`
 - B. `CONTROL_FILE_RECORD_KEEP_TIME=14`
 - C. `LOG_ARCHIVE_RETAIN=14`
 - D. `CONTROL_FILES=14`
 - E. `CONTROL_FILE_BACKUP_RETENTION=14`

Scenarios

1. You have just been hired as a consultant to help Guru Database Marketing Ltd. bulletproof their Oracle installation. You are asked to determine whether or not Guru Database Marketing Ltd. has adequately protected their control files in the five critical databases in use. The databases are stored on a single server with a RAID5 hardware array for the datafiles, a set of mirror hard disk drives where the redo log files are stored, as well as a hard disk where the operating system and Oracle software files are located. You are provided a printout of the Oracle initialization files for each of the five databases.
 - A. Which `INIT.ORA` parameters should you look at to determine if the control files are properly protected?
 - B. How many copies of the control file should exist for each of the databases and why?
 - C. Where should the control file copies be located and why?
 - D. If you needed to correct any deficiencies in the control file configuration, outline the steps you would take.

2. Your Oracle database has grown to include 60 datafiles and five log file groups. Each log file group has three members located on three different hard disks. Your control file is multiplexed on the same three hard disks. You attempt to create a new tablespace and associate two datafiles with the tablespace of 100MB in size each. You have 4GB of free disk space on each of the two drives where the datafiles will be located. The creation of the tablespace fails.
 - A. What is the most likely cause of the failure, assuming the paths to the file locations already exist and there is sufficient disk space?
 - B. What would you need to do to resolve the problem and allow for the creation of the new tablespace and datafiles?
3. You have just recently shutdown your instance and made a copy of the existing control file to another hard disk on your server. You have also modified the Oracle initialization file with the name of the new control file. When you attempt to start your instance, you receive an error that the new copy of the control file cannot be open and the instance is not mounted past the NOMOUNT stage. Outline two possible reasons the instance cannot be mounted.

Lab Exercises

Lab 6-1 Identifying the Control File Configuration

1. Connect to your instance using Server Manager line mode (svrmgrl) as a user with SYSDBA privileges.
2. Determine the names and locations of the control files for your database by displaying the contents of the CONTROL_FILES parameter using the SHOW PARAMETER command, as well as by querying the V\$CONTROLFILE view. Record the results in Table 6-1.

Table 6-1
Control File Configuration

<i>File Number</i>	<i>Location</i>
1	
2	
3	
4	

3. Do you have one or more control files?
4. How many control files should you have, as a minimum?
5. For each additional control file you think you should create, add the name and location to Table 6-1. You will need this for Lab 6-2.

Lab 6-2 Multiplexing the Control File

1. Using Server Manager line mode (svrmgrl), connect to your instance as a user with SYSDBA privileges.
2. Shutdown the instance using the SHUTDOWN IMMEDIATE command. Note that on most systems, this command will take a little while to shutdown the instance, so be patient.
3. Exit Server Manager.
4. Copy your existing control file to the new name and location you specified in Lab 6-1.
5. By using Windows NT Explorer, for NT systems, verify that the *Administrators* local group and the *SYSTEM* system group have full control permissions to each copy of the control file you created; if not, assign the appropriate permissions. For UNIX systems, use the `ls -l` command to verify that the UNIX group that owns the Oracle software has read and write permissions on the new control file copies; if not, use `chmod 660` and `chgrp`, or other UNIX commands to change the permissions on the file so that Oracle can read and write the file.
6. Locate your Oracle initialization file, such as `INITORCL.ORA`, and, using Notepad or `vi`, or any text editor, modify the `CONTROL_FILES` parameter to include the new control files. Make sure to place parentheses around the list on control files and separate each individual control file entry by a command (`,`). If the paths to your control files include spaces, make sure to fully enclose the entire control filename in double quotes (`"`).
7. Using Server Manager line mode, re-start your instance using the `STARTUP` command ensuring you specify the appropriate Oracle initialization file using the `PFILE` parameter of the `STARTUP` command.
8. Did the instance start properly? If so, CONGRATULATIONS. You have multiplexed your control files.
9. If the instance did not start properly, verify the file permissions, or, a more common error, verify the filenames in the Oracle initialization file. The most common reason for the instance not starting is typing mistakes when modifying the `INIT.ORA` file. I

Tip

If your instance did not start, shut it down before re-starting as the `INIT.ORA` file is only read on instance startup. Further, you will not be able to proceed past the `NOMOUNT` state without a valid control file parameter in the `INIT.ORA` file.

Lab 6-3 Verifying the Control File Configuration

1. Connect to your instance using Server Manager line mode (svrmgrl) as a user with SYSDBA privileges.
2. Determine the names and locations of the control files for your database by displaying the contents of the CONTROL_FILES parameter using the SHOW PARAMETER command, as well as by querying the V\$CONTROLFILE view. Record the results in Table 6-2.

Table 6-2
Control File Configuration

<i>File Number</i>	<i>Location</i>
1	
2	
3	
4	

3. Do you have more than one control file?
4. Did you successfully multiplex your control files?
5. Exit Server Manager line mode.

Answers to Chapter Questions

Chapter Pre-Test

1. The control file contains the database name, identifier, creation date, archiving status, datafile location, last update and status information, redo log file location, last update, SCN, and status information, tablespace names, log history, current log switch number, archiving history, last checkpoint, and backup information.
2. You should have at least two copies of the control file — preferably on separate physical disks. This protects against the loss of one of the control files due to disk failure and enables you to recover up to the point of failure, if necessary.
3. Control file information is updated whenever a checkpoint or log switch occurs. Control files are also updated whenever the database instance is shut down and started up.

4. The views that get their information from the control file include V\$DATABASE, V\$DATAFILE, V\$LOG, V\$LOGFILE, V\$TABLESPACE, V\$TEMPFILE, V\$LOGHIST, V\$ARCHIVE, V\$BACKUP, and V\$ARCHIVED_LOG.
5. Control files can grow dynamically because they also contain information about the last series of backups. This information is updated when Recovery Manager is used to perform backups and is maintained for a period of seven days by default. The retention time for backup information can be specified in days by modifying the Oracle initialization parameter CONTROL_FILE_RECORD_KEEP_TIME.
6. Multiplexing a control file requires you to make more than one copy of the control file. This can be done at database creation by specifying more than one file with the CONTROL_FILES parameter in the Oracle initialization file, or by copying an existing control file to a new location when the instance is shut down and then modifying the CONTROL_FILES parameter to include the new copy.

Assessment Questions

1. **D.** The Oracle initialization file is read to find out the value of the CONTROL_FILES parameter in order to open the control files during instance startup.
2. **C.** Cluster information is stored in the data dictionary and not in the control file.
3. **E.** The best way to protect the control file is to make sure a working copy exists on each physical disk on the system. In this way, should one disk disappear, the other copies of the control file can be used to keep the database running. Backing up the control file, either to trace or making a binary copy, is also recommended but is not the *best* way to protect the control file — which is what the question asked.
4. **A.** To determine how many datafile entries are still available in the control file, you need to look at the records in the control file. For this reason, a query of the V\$CONTROLFILE_RECORD_SECTION view is required.
5. **B.** CONTROL_FILE_RECORD_KEEP_TIME specifies the number of days backup entries will be kept in the control file.

Scenarios

1. In order to determine which control files exist, look at the CONTROL_FILES parameter of each of the parameter file printouts you were provided. Each database should have at least two copies of the control file on separate physical disks to ensure complete recovery in the case of failure. If this was not the case, you would recommend that each instance be shut down, a copy of the existing control file be made on another disk, the Oracle initialization file modified, and then the instance restarted. This would multiplex the control file and help protect against failure and potential data loss.

2. The most likely cause of the failure — assuming there is sufficient disk space and the paths are correct — is that you have reached the maximum datafiles that are available to the database. This is because the number of datafile records available is at its maximum and needs to be resized. When the control file was initially created, the value of the `MAXDATAFILES` parameter was set too low and now needs to be readjusted to suit the new requirements. To do this, you need to re-create the control file.

**Exam Tip**

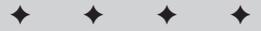
Re-creating the control file is not required for the Oracle8i Architecture and Administration exam.

3. The possible reasons that you are not able to start your instance include insufficient permissions or an incorrect filename for the `CONTROL_FILES` parameter in the Oracle initialization file. To correct the first, make sure you assign read and write permissions, as a minimum, to either the group that owns the Oracle software (on UNIX systems) or the Windows NT/2000 *Administrators* local group and *SYSTEM* system group on Windows NT/2000 platforms. To correct the second problem, make sure you verify the path and filename of the control files on disk are the same as in the `INIT.ORA` file. For UNIX systems, also remember that case is important. The most common reason for this problem is the incorrect spelling of the filename in the `INIT.ORA` file.

Maintaining Redo Log Files

EXAM OBJECTIVES

- ◆ Maintaining Redo Log Files
 - Explain the uses of the online redo log files
 - Obtain log and archive information
 - Control log switches and checkpoints
 - Multiplex and maintain online redo log files
 - Plan online redo log files
 - Troubleshoot common redo log file problems
 - Analyze online and archived redo logs



CHAPTER PRE-TEST

1. What information is kept in the online redo log files?
2. What is the relationship between redo log groups and redo log members?
3. When are redo log files written to and by which process?
4. How does running your database in ARCHIVELOG mode protect your database from data loss?
5. How do you multiplex online redo log files?
6. What are the three Oracle database initialization parameters that control checkpoints?
7. Does the size of the redo log files affect the frequency of checkpoints?
8. What are three dynamic performance views that contain information about redo log files?
9. How can you tell if your database is in ARCHIVELOG mode or not?
10. What can the LOGMINER utility be used for?

This chapter covers one of the more important areas of an Oracle database, the online redo log files. To appreciate the importance of the files and the need for their proper configuration, you first need to understand how they are used by Oracle. This is the first topic covered, followed by the relationship and function of log groups and members. You will then find out what information Oracle stores in the online redo log files as well as which processes are responsible for putting it there. This is followed by an explanation of the importance of checkpoints and log switches, as well as which factors influence their frequency. Next, you will learn about a critical topic—to archive or not to archive? This section covers the pluses and minuses of both options as well as the situations where one option is preferred over the other. Obtaining information about log files and maintenance operations is followed by the final section on LogMiner. The LogMiner utility provides for the extraction of both statistical and recovery information from the redo log and archive log files.

The online redo log files are required elements of all Oracle databases. Tuned correctly, major performance issues associated with large volume transaction processing databases and batch processing databases can be minimized. Tuned incorrectly, and the database users will be singing the blues. The files are key elements of both instance recovery and database recovery. Data loss can result if the files are missing or become corrupted. The DBA can, however, prevent this through proper configuration. The files need to be protected from loss and sized to satisfy the performance requirements of users.

Overview of the Online Redo Log Files

Objective

Explain the uses of the online redo log files.

Oracle uses the online redo log files primarily for recovery. Instance and database failures are recovered using the log files. The online redo log files are operating system (OS) files stored on disk. The files record all the change information made against the database (DML, DDL, and DCL commands) when a commit is issued. There are a few exceptions to this, such as transactions using the `NOLOGGING` or `unrecoverable` option, direct load inserts, and insert appends.

All changes made in an Oracle database are initially made in the database buffer cache. Oracle changes the data in the cache and also records the rollback block and data block changes in the redo log buffer. This gives Oracle the ability to rebuild the before and after image of the data blocks if required. Changes made in memory complete much quicker than making the same changes on disk. The problem with making changes in memory is that the changes are not protected from failure. If the instance crashes, those changes made only in memory will be lost. Oracle safeguards against this situation by writing committed changes to the online redo log files. It is the Logwriter's (LGWR) responsibility to perform this action. Upon successfully

writing all the transaction information to the current online redo log file, the LGWR process notifies the user that the commit has completed. If a failure happens, no committed transactions will be lost because Oracle can reapply all the transactions from the current log file to the appropriate data file. The number of changes that need to be applied from the online redo log files depends on when Oracle last performed a database synchronization. This event is called a *checkpoint*.

Checkpoints are responsible for taking all the dirty or changed buffers in the database buffer cache and writing them to disk. Checkpoints also update all data files and control files with the checkpoint information. Oracle uses this information to ensure that all data files and control files were last updated at the same time. This is important information for recovery.

For example, if the last database checkpoint occurred at 9:00 a.m. and a database failure occurs at 9:20 a.m., when the DBA restarts the database, Oracle scans the current online redo log for any transactions made since the last checkpoint. If transactions occurred between 9:00 a.m. and 9:20 a.m., then Oracle automatically starts instance recovery. All committed transactions between those two times are rebuilt before the database is made available for use. The more transactions that occurred between the checkpoints, the longer it is going to take Oracle to rebuild the database and make it available for use.

One of the main functions of a DBA is controlling the frequency with which checkpoints occur. More frequent checkpoints mean less work to be done during instance recovery. The downside to more frequent checkpoints is potentially poor database performance. Checkpoints can be very resource intensive, effectively slowing down the database, therefore, you need to find a balance between recovery time and acceptable database performance. Checkpoint frequency is determined either by the size of the online redo log files or one of three database initialization parameters.

Now that you know the basics of online redo log files and checkpoints, you may find yourself asking the same questions that many others have asked before you: “Why does Oracle just not write changes made in the database buffer cache to the data files? Would this not eliminate the need for online redo log files and checkpoints?” These questions are answered by understanding another role of online redo log files, archiving.

The online redo log files are used in a circular fashion as seen in Figure 7-1. This means that after one online redo log file becomes full, LGWR starts writing the change information to the next redo log file. When this file fills, it starts writing to the next one. This process continues until the last file becomes full. When this file is filled, Oracle starts writing to the first file again — overwriting this file and destroying all the transaction information it contained. Now what happens if a disk fails? The system administrator would repair or replace the disk and restore the information contained on that disk from a backup. Assuming that there were Oracle data files on this disk, those files would not be synchronized with the rest of the data

files. Oracle would report an error. The only way to recover those files, and make them current with the rest of the database, would be to apply all the changes stored in the online redo log files. However, if the online redo log files have been overwritten, Oracle no longer has the ability to perform this type of recovery. This would result in data loss and unhappy users who would be forced to re-enter information into the database—every DBA's worst fear.

To prevent this, Oracle has the ability to run the database in something called ARCHIVELOG mode. When the online redo log files fill, Oracle takes a copy of the file in something called an archive log file. These files are stored on disk or tape and can be used to recover from the scenario just described. No data is lost and no users are unhappy. Without redo log files and archive log files, Oracle would have no recovery capabilities, and this is a key element for all good databases.

Planning Redo Log Files

Objective

Plan online redo log files

Redo log file structure

As previously mentioned, the online redo log files are just operating system files. The format of the files depends upon the OS that Oracle is running on, however, the content will always be the same. The files store all the changes made by DML, DDL, and DCL statements performed. As you can imagine, the files are of tremendous importance. If the files are lost or corrupted, the end result may be the loss of data, so the DBA must protect against this by storing multiple copies of the files. This is referred to as multiplexing.

Redo log groups

Redo log files are part of redo log groups. The redo log groups are just logical entities. There is no physical component. The physical component of a group is the redo log file, or member. A group with more than one member is said to be multiplexed. The LGWR process will perform simultaneous writes to all redo log files that belong to the current group. Each group is assigned a unique log sequence number when it becomes the current group, the result of a log switch.

Cross-Reference

Log switches are covered in more detail in the section called “Log switches.”

The log sequence number is recorded in every data file and control file and is used by Oracle to determine database synchronization. If a data file contains a log sequence number that is older than the current log sequence number, then Oracle knows that the file is old and in need of recovery.

The optimal number of groups per database depends upon a number of critical factors, such as the speed of the system and the volume and nature of transactions being performed. It also depends upon the backup strategy you decide to implement. If you choose hot database backups, more redo log groups are typically required as more redo information is generated during these types of backups.

In certain situations, two log groups are all that is required. For high volume online transaction processing (OLTP) databases, more are needed. A group must always be available for LGWR; otherwise, it waits until the file becomes available. Waiting for the LGWR process causes poor database performance and generally unhappy customers since they cannot make any changes to the database during this time in this situation. If LGWR needs to wait for a group to become available—typically from incomplete checkpoints—it records a message in the ALERT file and generates an LGWR trace file. If the ALERT file has LGWR wait messages then more groups are required.

Groups are made up of members and while the same number of members is not required for every group, there is little benefit to not having that configuration. The only time the number of members between groups should differ is during maintenance operations.

Determining the optimal size of files in the redo log group is one of the most important factors for any transactional database. The minimum size is 50KB; the maximum is OS dependent. The size plays a key role in determining the frequency and the duration checkpoints. The size must be large enough to allow Oracle enough time for checkpoints to complete. If the files are too small and checkpoints too frequent, the database will begin to experience poor performance, as LGWR waits for checkpoints to complete. When LGWR is waiting, transactions are not processing.



Checkpoints are covered in more detail in the section called “Checkpoints.”

An important thing to remember about redo log groups is that Oracle requires a minimum of two. If two are not available, the database cannot start. If the database is up and running and a group becomes unavailable, leaving only one, then the database closes.

Redo log members

The redo log files are called *log members*. These are the physical files that exist on disk. The size of files is determined when the file is created. The size of the files is always inherited from the group. The size of all redo log files should be the same for all groups in the database. This promotes consistency in checkpoints. All members in a group will contain identical information. Only one member is required per group, but multiple members averts the problem of creating a single point of failure for the database. This is referred to as *multiplexing*.

Members from the same group should be on separate physical disks. This protects the database from a single point of failure for the group—the disk—and also helps reduce contention for writes by the LGWR. It is recommended that log files be placed on the fastest physical device on the system. LGWR is a busy process and the longer it waits to perform a write, the slower the database becomes. While this is true, it is uncommon to have different speed drives on the same system, therefore, what is more important is placing the files on inactive drives, drives with little or no activity except for the log files. With this configuration, LGWR does not compete with other system IO requests. Members should also be stored on separate physical drives from the drives storing archive redo log files when running the database in ARCHIVELOG mode. This helps avoid contention between the LGWR and ARCHIVER (ARCn) processes. Members should be placed on disks separate from data files. This eliminates contention between the LGWR and DATABASE WRITER (DBWn) process. This will also help prevent the loss of both data files and log files in the event of a disk failure.

Figure 7-1 illustrates the circular fashion by which online redo log groups are used, as well as an example of multiplexing online redo log members. Note that each group has two members (multiplexed), all groups are the same size, and members from the same group are on different physical disks.

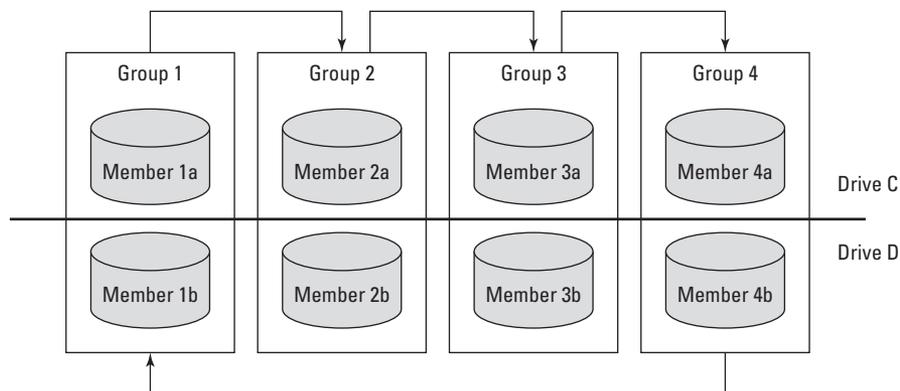


Figure 7-1: Online redo log groups and members configuration

Creation of initial redo log groups and members

The initial redo log groups and their members are created when the database is created. This is a requirement. The location of the files (members) and the numbers assigned to the groups is part of the CREATE DATABASE syntax. You should note that it is possible to create multiple members per group at database creation time. This is highly recommended.

The maximum number of redo log group members is also determined at database creation time. The value specified for `MAXLOGFILES` determines the maximum number of groups the database can have and `MAXLOGMEMBERS` is the maximum number of members per group. The maximum value for both is OS dependent. Here is an example of the `CREATE DATABASE` syntax.

```
CREATE DATABASE orcl
LOGFILE GROUP 1 ('C:\Oracle\oradata\orcl\redo01a.log',
                'E:\Oracle\oradata\orcl\redo01b.log') SIZE 1024K,
GROUP 2 ('C:\Oracle\oradata\orcl\redo02a.log',
         'E:\Oracle\oradata\orcl\redo02b.log') SIZE 1024K,
GROUP 3 ('C:\Oracle\oradata\orcl\redo03a.log',
         'E:\Oracle\oradata\orcl\redo03b.log') SIZE 1024K,
GROUP 4 ('C:\Oracle\oradata\orcl\redo04a.log',
         'E:\Oracle\oradata\orcl\redo04b.log') SIZE 1024K

MAXLOGFILES 32
MAXLOGMEMBERS 4
MAXLOGHISTORY 1
DATAFILE 'C:\Oracle\oradata\orcl\system01.dbf' SIZE 58M
REUSE AUTOEXTEND ON NEXT 640K
MAXDATAFILES 254
MAXINSTANCES 1
CHARACTER SET WE8ISO8859P1
NATIONAL CHARACTER SET WE8ISO8859P1;
```

This database would be initially created with four redo log groups and have two members per group. As you can see, the size of the files is only specified once, at the group level. All files belonging to this group will be the same size.

 **Tip**

When creating the initial log groups, make sure that you create a minimum of three. Oracle requires you to have at least two groups so if you are doing any maintenance in the future — such as relocating groups or fixing corrupted groups — and you only have two groups, you must always add a new group to get your database to three and then drop the problem group. If you try to drop the problem group first, the statement will fail because the action would leave Oracle with only one group.

Also in this example, the value for `MAXLOGFILES` is 32 meaning that this database could have an additional 28 groups on top of the four already created. The value for `MAXLOGMEMBERS` is 4 meaning that each group could have up to four members. The minimum number is one and in our case we have started with two members per group. This means that we could add an additional two members to each of our groups.

 **Tip**

The only downside to specifying higher values for `MAXLOGFILES` and `MAXLOGMEMBERS` is that it will make your control files bigger. When using the Oracle Database Configuration Assistant to create your databases the default value for `MAXLOGMEMBERS` is 2. The problem with having a maximum of two members per group comes up when you are performing maintenance operations later on. If you want to relocate members to different disks, you must drop members first

before you can add it. This leaves your database with a single point of failure. By setting it to a higher number, you can add the new members to the different drives and then drop the old members. This is a much safer option.

Multiplexing redo log files

If you want to avoid creating a single point of failure in your database, it is recommended that you multiplex your redo log files. Multiplexing online redo log files implies that you have more than one log member per group. Oracle requires a minimum of one valid log member per group for LGWR. Figure 7-1 is an example of multiplexed redo log files. Each group has two members, a and b. When LGWR is writing to group 1, it will write simultaneously to member 1a and member 1b. Both files will contain the same information. If either member becomes unavailable, the LGWR process writes an error to the ALERT file and generates a trace file with the details as to which file is no longer accessible. The status column of the V\$LOGFILE view will also indicate the invalid group. The important thing to note is that the database will continue processing. This is unlike control files. With control files, if any file becomes unavailable, the database stops processing and the DBA must resolve the problem before the database is operational. When the redo log files are multiplexed, the DBA can resolve the problem without interrupting the database.

Problems start when there are no log files in the current group available for LGWR. When this happens, Oracle will stop processing and the DBA will be forced to rectify the problem before the database becomes operational. If there were transactions in the online redo log files that had not been written to the data files before the database shutdown, those transactions would be lost. You can avoid this situation through multiplexing. The only time that the multiplexed online redo log files can be the cause of data loss is when all the members for a given group become unavailable. This most often occurs when all the members for a given group reside on the same physical disk and that disk fails. In Figure 7-1, you can see that all “a” members are on disk C and all the “b” members are on disk D. If disk D fails, Oracle can still use the members on disk C. If members 1a and 1b were on disk D and it failed, all members become unavailable and data loss may result. The other situation where this occurs is when a member of a multiplexed group becomes unavailable and the problem is not immediately fixed. If the group has two members and one fails, the group is left with one member. This is now a single point of failure. If that member fails, leaving no valid members for the group, data loss may result.

Online redo log files should always be multiplexed and members for a given group should always reside on separate physical disks.

Using Online Redo Log Files

Objective

Control log switches and checkpoints

Log writer

At any given time, one redo log group is the current group. The LGWR process is responsible for writing to members of the current redo log group. The LGWR process performs a write when one of the following criteria is met:

- ♦ A transaction commits
- ♦ The redo log buffer becomes one-third full
- ♦ More than one megabyte of changed records exists in the redo log buffer
- ♦ A timeout occurs (every three seconds)
- ♦ At every checkpoint

LGWR writes to the redo log groups in a circular fashion. It continues writing to the current group until it becomes filled. When it fills, a log switch occurs and LGWR will start writing the next group. This will continue until the last group fills at which point LGWR will start writing to the first group again. Overwriting those redo log files.

Log switches

A *log switch* is the event during which LGWR stops writing to the current redo log group and starts writing to another. In Figure 7-1, the arrows between Group 1 and 2, 2 and 3, 3 and 4, and 4 back to 1 illustrate log switches. At every log switch, Oracle will assign a unique log sequence number to the new group. This log sequence is assigned to a member in the log group to identify them and also in the control file and the header of all data files. The log sequence number is used by Oracle during recovery. If the log sequence number in the header of a data file is not the same as the control file, recovery is required.

The frequency of automatic log switches depends on three factors, the size of the online redo log group, the volume of transactions, and the type of transactions being performed. LGWR writes out all the information contained in the redo log buffer, for a given transaction, when a commit occurs. Inserts and deletes are typically larger than updates. This is because the redo log buffer stores the before and after image of a transaction. For inserts, the before image is simply a rowid while the after image is the entire row. The inverse is true for deletes. The before image is the entire row while the after image is a rowid. For updates, Oracle only needs to store the rowid, the column or columns that are changing, and the before and after data values.

Controlling the frequency of automatic log switches is one of the DBA's most important tasks. This is because at every log switch, a checkpoint occurs. Checkpoints are resource intensive events, generally resulting in minor pauses in database activity

while the checkpoint completes. The smaller the online redo log files relative to the transaction volume and the nature of the transactions, the more frequent are the log switches and checkpoints.

Checkpoints

Checkpoints are database synchronization events. At every checkpoint, all dirty buffers in the database buffer cache are written to the data files by DBWn. The CKPT background process updates the header of all data files and the control files to reflect the completion of the checkpoint. All files are now synchronized. If an instance failure occurs, Oracle only needs to recover the transactions that have occurred since the last checkpoint. Checkpoints occur for all data files or for individual files. Checkpoints on individual files occur only when data files are taken offline.

The DBWn process is always writing out dirty buffers from the database buffer cache, therefore, the number of dirty buffers that need to be written out during a checkpoint depends on how active the DBWn process has been since the last checkpoint. Many of the dirty buffers will have already been written out prior to the checkpoint event being triggered. The response delay for a checkpoint is typically one to two seconds but depends on the number of blocks that need to be written out. If the response delay is larger than two seconds it is usually an indication that checkpoints are not happening frequently enough.

Checkpoints can be forced manually by the ALTER SYSTEM CHECKPOINT command or automatically when one of the following events occurs:

- ♦ At every log switch
- ♦ When the instance is shutdown using the normal, immediate, or transactional options



Tip

If a SHUTDOWN ABORT is required, you should always attempt to perform a manual checkpoint using the ALTER SYSTEM CHECKPOINT command. If this succeeds, then instance recovery will not be required when the database is restarted.

- ♦ When forced by one of the following database initialization parameters: LOG_CHECKPOINT_INTERVAL, LOG_CHECKPOINT_TIMEOUT or FAST_START_IO_TARGET. (These parameters are covered in the section “Controlling Log Switches and Checkpoints.”)

Controlling log switches and checkpoints

An often-confusing Oracle subject is that of log switches and checkpoints. The next two sections address those two points.

Controlling log switches

Log switches occur automatically when the current redo log group fills or manually when the ALTER SYSTEM SWITCH LOG FILE command is executed. The ALTER SYSTEM system privilege is required to perform this command. As a result, unless log switches are forced manually, the frequency depends on the size of the redo log files and the frequency and nature of transactions as I previously mentioned. The recommended size for redo log files varies from 512KB to 100MB depending upon the database. The log switch event is always recorded in the ALERT file so the DBA can examine the alert file to determine if the switches are happening too frequently. Since it is usually difficult to alter the volume and nature of transactions, the only recourse for DBAs to alter the frequency of log switches is to alter the size of the online redo log files. This will be covered in the section “Maintaining Redo Log Files.”

Controlling checkpoints

In order to control checkpoints, you must first determine what is causing the checkpoint event. Most DBAs prefer to have automatic checkpoints occurring at log switches. Since DBAs are guaranteed that a checkpoint will occur at every log switch, this makes checkpoints easier to control. To increase the frequency of checkpoints, you decrease the size of the online redo log files. To decrease the frequency of checkpoints, you increase the size of the online redo log files.

**Tip**

All online redo log files should be the same size. This promotes consistency in log switches. If files in one group are bigger than files in another group and transaction volume is consistent, inconsistent checkpoint frequency will occur. If all online redo log files are the same size and transaction volume is consistent, the DBA can expect consistent checkpoints and database performance.

Checkpoints can also occur if triggered by one of the following database initialization parameters:

LOG_CHECKPOINT_INTERVAL If the target checkpoint position in the current online redo log group trails the current position being written to by the LGWR process by more blocks than specified by LOG_CHECKPOINT_INTERVAL, then a checkpoint event is triggered. The LOG_CHECKPOINT_INTERVAL is specified in OS blocks and not Oracle database blocks. The default value for LOG_CHECKPOINT_INTERVAL is OS dependent.

**Tip**

Setting this parameter to a value larger than the number of OS blocks occupied by the online redo log files ensures that the checkpoint event will never be triggered as a result of this parameter, since checkpoints are always forced at log switches.

LOG_CHECKPOINT_TIMEOUT If the target checkpoint position in the current online redo log group trails the current position being written to by the LGWR process by more seconds than specified by LOG_CHECKPOINT_TIMEOUT, then a checkpoint event is triggered. The parameter is specified

in seconds. The default value is 1800 seconds. This means that no more than 1800 seconds or 30 minutes can elapse between checkpoints. The counter is reset to zero at every checkpoint. Many DBAs will leave the default value to ensure that, at a minimum, no more than 30 minutes can elapse without a checkpoint. If checkpoints are typically occurring every 20 minutes, then this parameter will only cause checkpoints during periods of reduced activity.



Tip

Setting the `LOG_CHECKPOINT_TIMEOUT` parameter to zero turns off the parameter.

FAST_START_IO_TARGET Setting this value to 10,000 ensures that there can be no more than 10,000 dirty buffers in the database buffer cache. More accurately, it ensures that if an instance failure occurs, no more than 10,000 blocks would need to be read during recovery. The parameter is intended to ensure bounded recovery time. If set, DBAs can now estimate how long instance recovery will take by determining how long it takes to perform the 10,000 IOS. It ensures that the target checkpoint position in the current online redo log group trails the current position being written to by the LGWR process by more data blocks than specified by `FAST_START_IO_TARGET`. When this parameter is being used, checkpoint frequency will vary, depending on the number of different blocks being modified in the database. The default value for the parameter is the number of database buffers specified by the parameter `DB_BLOCK_BUFFERS`. The valid range of values is 1000 to the number of `DB_BLOCK_BUFFERS`.



Tip

Setting the `FAST_START_IO_TARGET` parameter to zero turns off the parameter.

Determining which, if any, of the parameters is affecting the frequency of checkpoints can be determined by querying the `V$INSTANCE_RECOVERY` view. There are seven columns in the view and, for readability, the names have been abbreviated.

```
SQL> col recovery_estimated_ios head rec_est_io
SQL> col actual_redo_blks head act_red_blks
SQL> col target_redo_blks head target_blks
SQL> col log_file_size_redo_blks head log_size
SQL> col log_chkpt_timeout_redo_blks head timeout
SQL> col log_chkpt_interval_redo_blks head interval
SQL> col fast_start_io_target_redo_blks head fast_st_io
SQL>
SQL> select * from v$instance_recovery;
```

```
rec_est_io act_red_blks target_blks log_size timeout interval fast_st_io
-----
120          1764          1843          1844          5838          10000          5883
```

The columns have the following meaning:

- ♦ `rec_est_io` is the estimated number of IO operations required for instance recovery.
- ♦ `act_red_blks` marks the position of the last write by the LGWR process in blocks.
- ♦ `target_blks` represents the position in the redo log file where the next checkpoint will occur in blocks. When the `act_red_blks` reaches this value, a checkpoint will be triggered. It is important to note that this value will change dynamically as it is determined, based upon the values of the three initialization parameters (`LOG_CHECKPOINT_INTERVAL`, `LOG_CHECKPOINT_TIMEOUT`, and `FAST_START_IO_TARGET`) and the size of the current online redo log file. The parameter `LOG_CHECKPOINT_TIMEOUT` will force the value to change and more time passes without a checkpoint than is the nature of the parameter. The `FAST_START_IO_TARGET` parameter will force the value to change as more blocks are modified in the database buffer cache.
- ♦ `log_size` represents the size of the current online redo log file in blocks.
- ♦ `timeout` represents the value of the `LOG_CHECKPOINT_TIMEOUT` parameter in blocks.
- ♦ `interval` represents the value of the `LOG_CHECKPOINT_INTERVAL` parameter in blocks.
- ♦ `fast_st_io` represents the value of the `FAST_START_IO_TARGET` parameter in blocks.

You can see that the size of the online redo log files, represented by the `log_size` column, is having the greatest influence on the frequency of checkpoints.

Now, if we were to change the value for `LOG_CHECKPOINT_TIMEOUT` to a low value, like ten seconds, we would expect this to have the greatest influence on the frequency of checkpoints.

```
SQL> alter system set log_checkpoint_timeout=10;
```

```
System altered.
```

```
SQL> select * from v$instance_recovery;
```

```
rec_est_io act_red_blks target_blks log_size  timeout  interval fast_st_io
-----
          3             2           2     1844         2     10000     5614
```

After reducing the value of `LOG_CHECKPOINT_INTERVAL` to ten seconds, you can now see that the `timeout` column has changed to two, matching the `target_blks` column. The `LOG_CHECKPOINT_TIMEOUT` parameter is indeed having the greatest influence on the frequency of checkpoints.

In most situations, the easiest way to control checkpoint frequency is by adjusting the size of the online redo log files. This generally leads to more consistent checkpoints. The decision to use one of the database initialization parameters is made when bounded recovery time is of high priority.

Archiving Redo Log Files

To archive or not to archive, that is the question. Luckily for us, it has a simple and straightforward answer. The decision can be made by answering the following question, “Can I afford to lose data?” or, more precisely, “Can I afford to lose any changes made to the database?” If your answer is no, then you must run your database in ARCHIVELOG mode. In most production environments where users are constantly adding, updating, and deleting information, running the database in ARCHIVELOG mode is mandatory. The last thing a DBA ever wants to do is go to the users and try to explain why they need to reenter data. In fact, if your database is connected to the Internet and you have Internet users adding, updating, and deleting information, it is impossible to have them reenter data because many times you don’t know who they are.

To fully understand the costs and benefits of running your database in ARCHIVELOG mode or NOARCHIVELOG mode, you first need to understand what implications the two modes have on a database.

NOARCHIVELOG mode

When your database is running in NOARCHIVELOG mode, Oracle will perform a checkpoint at log switches. Eventually, after the last log group has filled and a checkpoint occurs, LGWR will start writing to the first online redo log group again, overwriting the information contained in the files. The online redo log files are not being backed up, just overwritten.

Since the DBWn is writing out all the dirty buffers in the database buffer cache during checkpoints, your data files are being updated. The information in the online redo log files is no longer required for instance recovery. It is important to note that Oracle will not let LGWR start overwriting the information in a log file until the checkpoint triggered when the log file was switched has completed. You have no risk of losing any data under this scenario except if you were to lose a data file. If you were to lose a data file, you will have lost the ability to recover that data file because the information contained in the online redo log files to recover it has been overwritten. Figure 7-2 gives an example of this scenario.

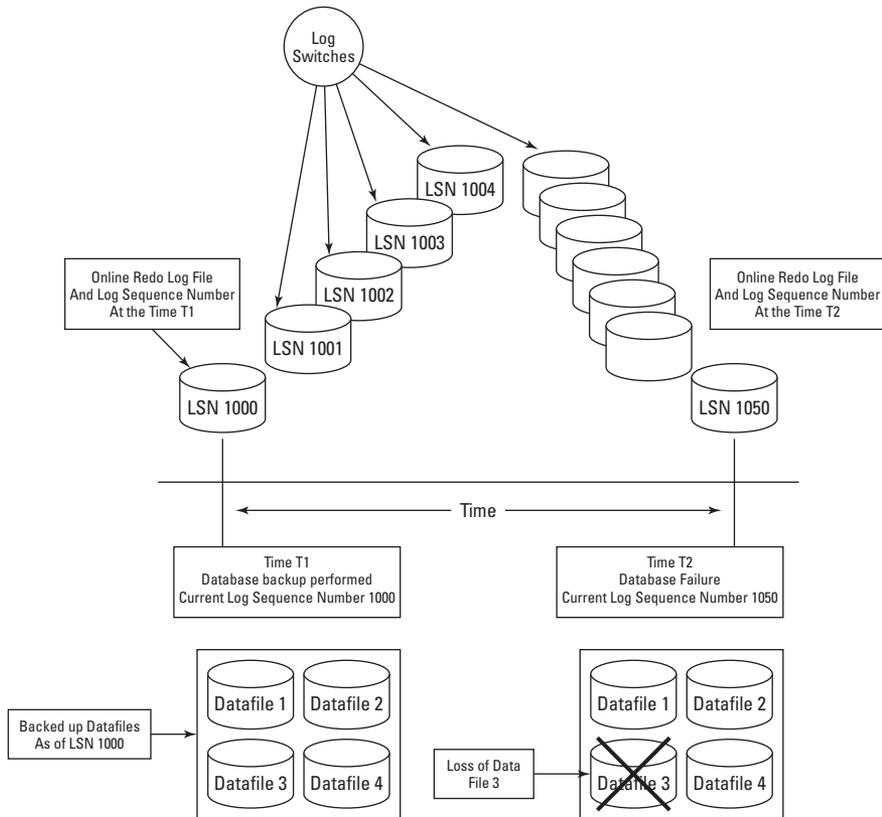


Figure 7-2: Running a database in NOARCHIVELOG mode

In this example, a backup was taken at Time T1. All the backed up data files would contain the current log sequence number in the header. Over time, you can see that log switches are taking place and eventually the online redo log file that contained the transactions for log sequence number 1001 will be overwritten. At Time T2 there is a failure. The disk that data file 3 resides on fails. The current log sequence number for the database is 1050.

After the disk is repaired or replaced, data file 3 must be restored from the backup taken at Time T1. The problem is that all the online redo log files required to get data file 3 up to the time T2 have long since been overwritten. Oracle cannot recover the data file. Since Oracle will not allow for data files to be unsynchronized with the rest of the database, all the data files taken in the backup at Time T1 will need to be restored and all the transactions from log sequence number 1001 to log sequence number 1050 will be lost.

It is for this reason that most production databases cannot afford to run in NOARCHIVELOG mode. The only situation in which NOARCHIVELOG mode is acceptable is when the data is static, such as a data warehouse or in test environments where the loss of data is acceptable.

ARCHIVELOG mode

The main difference between ARCHIVELOG and NOARCHIVELOG mode is that in ARCHIVELOG mode, when a log switch occurs, not only does Oracle perform a checkpoint, it takes a copy of current online redo log files. In ARCHIVELOG mode a copy of the redo log file must be taken before the online redo log file is overwritten.

The backed up online redo log files are called archive redo log files. The archive is done either manually or automatically. When done manually, the DBA must issue either the ALTER SYSTEM ARCHIVE LOG CURRENT or ALTER SYSTEM ARCHIVE LOG ALL commands for the file to be archived. If the archive does not happen, then Oracle will not allow the online redo log file to be overwritten. If the log switch switches to a group that has not been archived, the database will stop processing until the archive completes. The phone will start ringing off the hook as users discover they are no longer able to process transactions.

Automatic archiving is generally much more appealing for DBAs. With automatic archiving, there is a process that is responsible for performing the archive. When a log switch occurs, the ARCn process will automatically write the redo log file to the destination specified by the database initialization parameter LOG_ARCHIVE_DEST. To enable automatic archiving, the database initialization parameter LOG_ARCHIVE_START must be set to true.

The benefit of running the database in ARCHIVELOG mode is that when a failure happens and a data file is lost, Oracle can use the archive log files to rebuild a backed-up data file to the time the failure occurred. There is zero data loss. Figure 7-3 is an example of a database running in ARCHIVELOG mode.

With the database running in ARCHIVELOG mode, as the log switches occur, the online redo log files will be archived, preserving a copy of the file and all the transactions contained within.

In the example shown in Figure 7-3, a backup was taken at Time T1. All the backed-up data files would contain the current log sequence number in the header. Over time, you can see that log switches are taking place and eventually the online redo log file that contained the transactions for log sequence number 1001 will be overwritten. At Time T2 there is a failure. The disk that data file 3 resides on fails. The current log sequence number for the database is 1050.

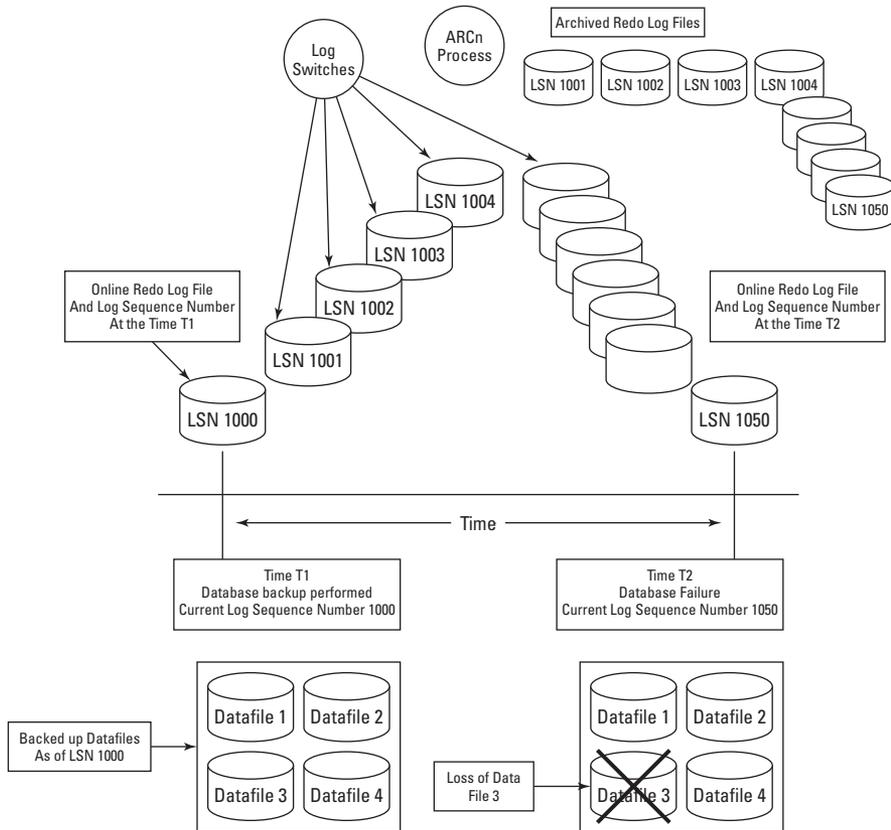


Figure 7-3: Running a database in ARCHIVELOG mode

After the disk is repaired or replaced, data file 3 must be restored from the backup taken at Time T1. The DBA will issue the appropriate database recovery command and Oracle will rebuild data file 3 to the time of the failure Time T2. There will be zero data loss.

It is for this reason that most production databases run in ARCHIVELOG mode. Data loss is usually not acceptable. However, running in ARCHIVELOG mode offers other benefits as well, such as hot or online database backups and point in time recovery. Running in ARCHIVELOG mode is not without costs. The system will require space to store the archive log files and the DBA must ensure that that space does not become full. You must remember that if the online redo log file is not archived, it cannot be overwritten when its turn comes up after a log switch. If it has not been archived, Oracle will stop processing until it is. This will cease operations on the database and users are generally not happy when they cannot proceed with their work.



The exam does not contain questions pertaining to getting the database in ARCHIVELOG mode, just questions about the differences between ARCHIVELOG mode and NOARCHIVELOG mode as well as questions about automatic and manual archiving.

Maintaining Redo Log Files

Objective

Obtain log and archive information

Obtaining information about redo log files and archiving

Oracle provides several dynamic performance views and special commands to obtain information about online redo log files, as well archiving.

V\$THREAD

To view the number of online redo log groups, the current online redo log group, and the log sequence number of the current group, run the following query:

```
SELECT groups, current_group#, sequence#
FROM V$THREAD
```

The GROUPS column in the output represents the total number of online redo log groups, the CURRENT_GROUP# represents the current online redo log group and the SEQUENCE# represents the log sequence number assigned to the current group.

GROUPS	CURRENT_GROUP#	SEQUENCE#
3	1	193

This database has three online redo log groups and the current online redo log group is 1 and the assigned log sequence number is 193. When a log switch occurs, the CURRENT_GROUP# and SEQUENCE# will change. After issuing the ALTER SYSTEM SWITCH LOGFILE command and rerunning the query, the output looks like this:

```
ALTER SYSTEM SWITCH LOGFILE
```

GROUPS	CURRENT_GROUP#	SEQUENCE#
3	2	194

V\$LOG

The V\$LOG view contains information about individual online redo log groups. To obtain information about the log groups, run the following query:

```
SELECT group#, sequence#, bytes, members, archived, status
FROM V$LOG
```

GROUP#	SEQUENCE#	BYTES	MEMBERS	ARC	STATUS
1	193	1048576	1	YES	INACTIVE
2	194	1048576	1	NO	CURRENT
3	192	1048576	1	YES	INACTIVE

GROUP# represents the assigned group number, SEQUENCE# is the assigned log sequence number from the last log switch, BYTES is the size of the group in bytes, MEMBERS is the number of online redo log members assigned to the group, ARCHIVED indicates if the online log group has been archived, and STATUS represents the status of the group. The following values for the STATUS column are the most common:

- ♦ UNUSED indicates the online redo log group has never been written to. After online redo log groups have been added, this is the initial status. The status changes once the group is switched to during a log switch.
- ♦ CURRENT indicates that the group is the current online redo log group that LGWR is writing to.
- ♦ ACTIVE indicates that the online redo log group is active or has been used and it is not the current group, however, it is needed for recovery. Generally when logs have this status, they are waiting for a checkpoint or archiving to complete.
- ♦ CLEARING indicates that the log is being cleared as a result of the ALTER DATABASE CLEAR LOGFILE command. This is a maintenance command covered in the section on Maintaining Redo Log Files. Once the file has been cleared, the status changes to UNUSED.
- ♦ INACTIVE indicates the online redo log group is no longer required for instance recovery. Checkpoints and/or archiving events have completed. This is along with the status of CURRENT are two most common for log groups.



If the status of the group is not changing from ACTIVE to INACTIVE it is usually an indication that archiving has not succeeded. Check the ALERT file for errors. This usually happens when the drive storing the archive redo log files becomes full.

V\$LOGFILE

The V\$LOGFILE view contains information about the individual online redo log files. To obtain information about the online redo log files, run the following query:

```
SELECT *
FROM V$LOGFILE
```

GROUP#	STATUS	MEMBER
1		C:\CERTDB\DISK3\REDO01.LOG
2		C:\CERTDB\DISK3\REDO02.LOG
3		C:\CERTDB\DISK3\REDO03.LOG

Where the GROUP# column indicates the group that the file belongs to. The STATUS column indicates the current status of the online redo log file. The following values in the status column are the most common:

- ♦ NULL, which is the current status in the preceding example. The word NULL will not appear. This is the most common status and indicates that there are no problems with the files.
- ♦ STALE indicates that the files have not been accessed by Oracle. If you add members to a log group, the new members added will have status of stale until the group that the members were assigned to becomes the current group. The confusing part about this status is that if you add a member to the current online redo log group, the file will maintain the status of STALE until that group becomes the current group again. No activity is required by the DBA if the status is STALE.
- ♦ INVALID indicates that the file is inaccessible by Oracle. When files indicate this status, red flags should be waving, and the DBA should immediately resolve this problem. When the file is INVALID, LGWR stops writing to it. If the problem is not fixed, data loss can occur from another failure.
- ♦ DELETED indicates that the file is no longer used.

Information about archiving is also obtained through dynamic performance views plus a special command called ARCHIVE LOG LIST.

V\$DATABASE

By querying the V\$DATABASE view, you can obtain whether or not your database is running in ARCHIVELOG or NOARCHIVELOG mode. Run the following query:

```
SELECT name, log_mode
FROM V$DATABASE
```

NAME	LOG_MODE
CERTDB	ARCHIVELOG

The NAME column is the database name and the LOG_MODE column indicates if the database is in ARCHIVELOG or NOARCHIVELOG mode.

V\$INSTANCE

By querying the V\$INSTANCE view, you can determine the status of the archiver process. Run the following query:

```
SELECT archiver
       FROM V$INSTANCE

ARCHIVE
-----
STARTED
```

A status of STARTED means that automatic archiving has been enabled for the database and that the ARCn process is running. If the value is STOPPED, this indicates that either manual archiving or no archiving is being performed.

ARCHIVE LOG LIST

The information available for archiving taken from the V\$DATABASE and the V\$INSTANCE views is also available by running a special command, ARCHIVE LOG LIST in Server Manager line mode.

```
ARCHIVE LOG LIST

Database log mode                Archive Mode
Automatic archival              Enabled
Archive destination             C:\CERTDB\DISK6\archive
Oldest online log sequence      192
Next log sequence to archive    194
Current log sequence            194
```

Apart from the archive mode and the status of the automatic archiving, the command also returns the location for the archive log files, the log sequence number of the oldest online redo log file, the value of the next log sequence number to be archived, and the value of the current log sequence number — sort of a one-stop shopping for log and archive log information.



Archiving changed with release 8.0 of Oracle and then again with release 8.1. The value of Archive destination from the ARCHIVE LOG LIST command may not be accurate. V\$ARCHIVE_LOG_DEST contains the accurate information about the location of archive log files.

Managing Redo Log Files

Objective

Multiplex and maintain online redo log files

Performing maintenance operations to online redo log files is hopefully something that does not happen often in a database. Maintenance is generally required as a result of a failure or poor initial design. If log files become corrupted, lost, or the files are not multiplexed, maintenance operations will be required. Another factor that will force maintenance operations on log files is a change in the database workload. If transaction volume in the database increases, generally the result of adding new applications or simply more users, the online redo log files will probably be required to change as well.

Adding online redo log groups

If the current online redo log groups cannot support the transaction demand, either because checkpoints are happening too frequently or the log groups are too small, you need to add new groups. If your goal is to simply increase the size of the online redo log groups, you must first add new groups and then drop the smaller groups. Oracle does not allow modifications to the size of the online redo log files. The only course of action is to add larger groups and then drop the smaller ones.

The following ALTER DATABASE command will add a new online redo log group:

```
ALTER DATABASE ADD LOGFILE  
GROUP 4 'C:\CERTDB\DISK3\log4a.rdo' SIZE 1M
```

In this example, the online redo log group is being assigned the group number of 4. If the group number assignment is omitted, Oracle will automatically assign it a number. In this example, the one member belonging to this group will be one megabyte in size. Groups must have a minimum of one member. You should note that this group is not multiplexed. There is only one member. As you recall, I recommend multiplexing online redo log files. If you decide to multiplex the online redo log files, then you need to add a member to this group. Instead of a two-step operation — creating the groups and then adding additional members — groups can be multiplexed when they are created. The following command will create GROUP 4 with 3 members. Note that multiplexing is not limited only to two files.

```
ALTER DATABASE ADD LOGFILE  
GROUP 4 ('C:\CERTDB\DISK3\log4a.rdo',  
         'C:\CERTDB\DISK4\log4b.rdo',  
         'C:\CERTDB\DISK5\log4c.rdo') SIZE 1M
```

Note that in this example the size for the online redo log files is only specified once. All files for a given group must be the same size, therefore, the size of the files can only be specified at the GROUP level.



Tip

You should always specify the full pathname of the files. If omitted then they are created in the default directory of the database server. It is unlikely that this is the same directory as where you desire the files to be created.

Adding online redo log members

Adding online redo log members is done to either multiplex an existing online redo log group or to relocate online redo log members to different disks or directories as part of a database reorganization. Issue the following command to add an online redo log member to GROUP 4:

```
ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK6\log4d.rdo' TO GROUP 4
```

Note that in this example, the size of the member is not specified. The size is inherited from the group it is being assigned to. You can also use the ALTER DATABASE ADD LOGFILE MEMBER command to add members to all groups at the same time. Issue the following command to add a member to all of the online redo log groups at the same time.

```
ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK6\log1c.rdo' TO GROUP 1,
'C:\CERTDB\DISK6\log2c.rdo' TO GROUP 2,
'C:\CERTDB\DISK6\log3c.rdo' TO GROUP 3,
'C:\CERTDB\DISK6\log4c.rdo' TO GROUP 4
```



Tip

If the file already exists, then it must be the same size as the file that is being added, and you must add the REUSE option, as shown in the following code:

```
ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK6\log1c.rdo' REUSE TO GROUP 1,
'C:\CERTDB\DISK6\log2c.rdo' REUSE TO GROUP 2,
'C:\CERTDB\DISK6\log3c.rdo' REUSE TO GROUP 3,
'C:\CERTDB\DISK6\log4c.rdo' REUSE TO GROUP 4
```

Dropping online redo log groups

Online redo log groups are usually dropped as part of a resizing or reorganizing exercise. If you determine that the online redo log groups are too small and wish to increase them, you must first create the new larger groups and then drop the smaller ones. Figure 7-4 illustrates this scenario. The DBA has determined that the online redo log files are too small. Checkpoints are happening too frequently, resulting in poor database performance. The DBA first adds two new larger groups, 3 and 4, before removing the smaller groups, 1 and 2.

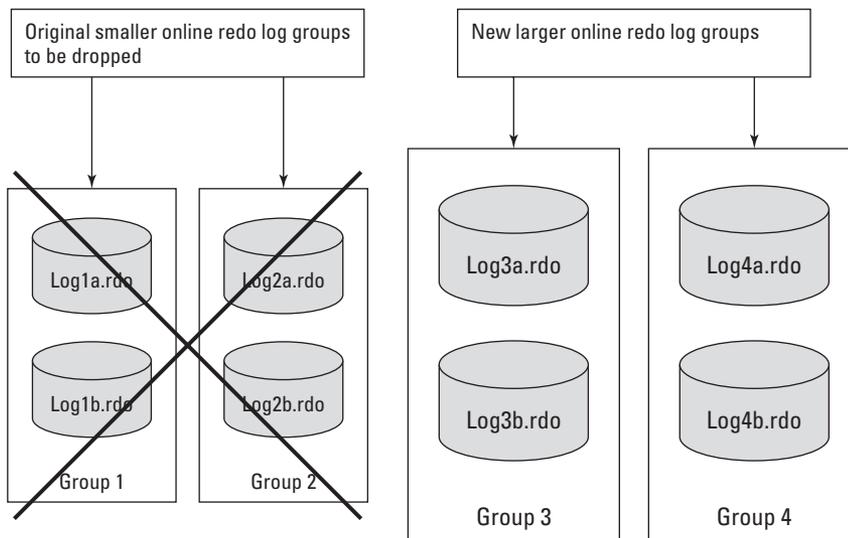


Figure 7-4: Dropping online redo log groups

The `ALTER DATABASE DROP LOGFILE` command is used to drop online redo log groups. The command to drop groups 1 and 2 in Figure 7-4 would be as follows:

```
ALTER DATABASE DROP LOGFILE GROUP 1, GROUP 2
```

When dropping an online redo log group, there are several things that you must be aware of:

- ♦ There must be at least two groups of online redo log files. Oracle will return an error if the `ALTER DATABASE DROP LOGFILE` command leaves the database with only one group. A new group must be added before another can be dropped.
- ♦ Groups with a status of either `ACTIVE` or `CURRENT`, taken from the `V$LOG` dynamic performance cannot be dropped. If the group is the current group, an `ALTER SYSTEM SWITCH LOGFILE` command can be issued to force a log switch. The group will no longer be the current group and can be dropped. Note that the status will change from `CURRENT` to `ACTIVE`. When the status is `ACTIVE`, Oracle is waiting for the checkpoint and/or the archive event to complete. When completed, the status will be changed to `INACTIVE` and the group can be dropped.
- ♦ When dropping groups, all members of the group will also be dropped.
- ♦ When online groups are dropped, the files are not removed from the OS. The files will need to be removed manually by issuing an OS command.

Dropping online redo log members

Online redo log members are usually dropped when the DBA is relocating files to new disks or when the status of the member is INVALID. The V\$LOGFILE view contains the status of log members. The status is set to invalid when LGWR can no longer write to the file. This can happen either through corruption, loss of a disk, or if the file is accidentally or intentionally removed at the OS.



Caution

Database files, including log files, must always be dropped in Oracle prior to being removed at the OS. If a file is removed at the OS first, the OS does not notify Oracle of the action. Therefore, when Oracle attempts to use the removed file, it will report an error. The LGWR process will stop writing to the file and place an error in the alert file as well as generate a trace file.

Issue the following command to remove one or more members of a specified group or groups:

```
ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK6\LOG4C.RDO'
```

If you want to drop multiple members, just separate the files with a comma:

```
ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK6\LOG3C.RDO', 'C:\CERTDB\DISK6\LOG4C.RDO'
```

Note that when dropping members, the group does not need to be specified.

As with dropping online redo log groups, there are restrictions to dropping online redo log members:

- ♦ If there is only one online redo log member left in a group, the member cannot be dropped. You must drop the group. If attempted, Oracle returns the following error:

```
ORA-00361: cannot remove last log member C:\CERTDB\DISK3\RED003.LOG for group 3
```

- ♦ If the member you are dropping belongs to the current group, members cannot be dropped. If attempted, Oracle returns the following error:

```
ORA-01609: log 3 is the current log for thread 1 - cannot drop members
```



Tip

If you want to drop a member of the current group, issue the ALTER SYSTEM SWITCH LOGFILE command to force a log switch. It will no longer be the current group.

- ♦ If the database is running in ARCHIVELOG mode, the group the member belongs to must be archived before it can be dropped.
- ♦ When removing members, the file at the OS will not be removed. It must be removed at the OS after being dropped in Oracle.

Moving or renaming online redo log files

Moving or renaming online redo log files can be performed two ways. The easier of the two methods is to create the new members with the new name or in the new location by issuing the `ALTER DATABASE ADD LOGFILE MEMBER` command followed by the `ALTER DATABASE DROP LOGFILE MEMBER` command to drop the old members. The other option involves issuing the `ALTER DATABASE RENAME FILE` command. Before issuing this command, the files in the new location must first exist. This is accomplished by copying the file at the OS to the new location prior to issuing the command.



Prior to copying the source file at the OS to the target file and location, ensure that the files do not belong to the current group. If they do, issue the `ALTER SYSTEM SWITCH LOGFILE` command prior to performing the copy. This will ensure that the file is not being written to by LGWR when attempting to perform the copy.

If renaming or relocating online members with this method, follow these steps. These steps will rename the member for group 3. Note that the steps would be the same if relocating the files to a new directory.

STEP BY STEP: Renaming an Online Redo Log File

1. Ensure that the member does not belong to the current group. Issue the following query against the `V$THREAD` view to determine the current group number.

```
SELECT CURRENT_GROUP# FROM V$THREAD
```

```
CURRENT_GROUP#
-----
3
```

2. Since group 3 is the current group and we are relocating the member from group 3, an `ALTER SYSTEM SWITCH LOGFILE` command must be issued. The command can be skipped if the member does not belong to the current group.
3. Copy the file at the OS to the new location and name. This can be done from SQL PLUS.

```
SQL> HOST COPY C:\CERTDB\DISK3\REDO03.LOG C:\CERTDB\DISK3\LOG3A.RD0
```

4. Issue the `ALTER DATABASE RENAME FILE` command. This command does not actually rename the file at the OS; it just simply changes the pointer in the control files.

```
ALTER DATABASE RENAME FILE 'C:\CERTDB\DISK3\REDO03.LOG' to
'C:\CERTDB\DISK3\LOG3A.RD0';
```

5. Remove the old file from the OS.

```
SQL> HOST DEL C:\CERTDB\DISK3\REDO03.LOG
```

Clearing online redo log files

When the status of an online redo log member is `INVALID`, there are two options for resolving the problem. The first option involves adding new members using the `ALTER DATABASE ADD LOGFILE MEMBER` command followed by the `ALTER DATABASE DROP LOGFILE MEMBER` command to drop the corrupted member. The second option involves using the `ALTER DATABASE CLEAR LOGFILE` command. The command has two options — clearing individual members of a group or clearing all members of a group. The choice depends on the severity of the corruption. To clear individual members, issue the following command:

```
ALTER DATABASE CLEAR LOGFILE 'C:\CERTDB\DISK3\LOG3A.RD0'
```

The following command will work to clear all the members of a group:

```
ALTER DATABASE CLEAR LOGFILE GROUP 3
```

If the database is running in `ARCHIVELOG` mode, a special option of the command, `UNARCHIVED`, is required to clear unarchived log files. This applies to log files with a status of `ACTIVE`.

```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 3;
```

Using Oracle Enterprise Manager

The Oracle Enterprise Manager (OEM) utility can be used for performing almost all the maintenance operations on redo log files.

STEP BY STEP: Managing Online Redo Log Files with DBA Studio

1. Launch DBA Studio and select the Launch DBA Studio in Stand Alone option. Start ⇨ Programs ⇨ Oracle OEM (This name may be different on your installation) ⇨ Database Administration ⇨ DBA Studio
2. Select the CERTDB database and enter the login information if required.
3. Select Storage.
4. Select Redo Log Groups.
5. Right-click on the Redo Log Groups and then select the desired option for adding log groups or switching log files. To drop a group, member, or add members to existing groups, right-click on the group number and select the desired option.

Troubleshooting LGWR Problems

Objective

Troubleshoot common redo log file problems

Checking the ALERT file should be done on a regular basis. It reports error messages generated from LGWR. The ALERT file will also report error messages from other processes as well as general database activity, which is why it is so important to check.

One important point to remember about redo log files is that as long as there is one valid member in the current group, LGWR will not cause the database to fail. If there are no members available in the current group, the database will fail and data loss may result. In the perfect world, this scenario should never happen, because redo log members should always be correctly multiplexed. But since this is not a perfect world, it is important to understand what happens when LGWR fails. The following table shows what you need to do to fix the problems.

<i>Problem</i>	<i>Solution</i>
Losing a member of a multiplexed group. There are still valid members in the group. Database will not fail and LGWR will continue processing.	First determine if the member belonged to the current group. If so, for a manual log switch, when it is no longer the current group you can simply add a new member and drop the invalid one. (These steps were covered in the section on Maintaining Redo Log Files.)
All members from the next group are unavailable and a log switch occurs. This will cause the database to fail because LGWR has nowhere to record transaction information. It is important to note that Oracle will not continue to switch log files until a valid group is found. The database will fail.	Since it was the group that Oracle was trying to switch into and not the current group, you will have zero data loss. Get the database to a mount state and then simply create a new group, replacing the bad one, or fix or replace the members in the invalid group.
All the members of the current group are invalid and the database does not initially fail.	Since the database did not initially fail, the only recourse for the DBA is to attempt to force a manual checkpoint. If this succeeds, then there will be no data loss. The checkpoint will write all dirty buffers from the database buffer cache to the data files. Once the checkpoint completes, the database will need to be recovered from the loss of the current online redo log group. If the checkpoint does not succeed, there may be data loss and the database will need to be recovered from the loss of the current online redo log group.
All the members of the current group are invalid and the database failed.	Any transaction written to the current group by LGWR will be lost since the database failed. The database will need to be recovered from the loss of the current online redo log group.

Using LogMiner

Objective

Analyze online and archived redo logs

The LogMiner utility is a new feature for Oracle8i. It provides for the interpretation of the online redo log or archive log files for any Oracle 8.0 database or greater. The utility translates the contents of the files into SQL statements that represent the logical operations performed against the database. This functionality has dramatically improved the DBA's toolbox. Prior to LogMiner, mapping data access patterns, dropped objects, or busy periods in the day, could only be done through database auditing. The problem with auditing was the tremendous overhead it added to a database. The overhead usually prevented DBAs from using it. Also, if the DBA needed to recover a committed transaction, the only option was point-in-time-recovery. This type of recovery is complicated and risky and it also requires the database to be closed. LogMiner can be used to read through log files and extract both the committed transaction to be recovered as well as the undo statement for this transaction. No recovery is required and the database need not be closed.

LogMiner adds no overhead to a production system because the analysis is being performed against transactions that have already occurred and against files that are not currently being used by Oracle.

How LogMiner Works

Before you can use LogMiner to analyze online redo or archive log files, you first need to create a dictionary file. The dictionary file allows Oracle to map transactions in a log file to an object in the database. The dictionary file contains the database it was created for as well as a list of all the objects. While the dictionary file is not required, interpreting the output without it is very difficult. The dictionary file only needs to be created once, and then again as new objects, like tables, are added to the database. The `DBMS_LOGMNR_D.BUILD` procedure is used to create the dictionary file. The `DBMS_LOGMNR` package is created when the `CATPROC.SQL` script is run as part of the database creation. A directory must be identified in the `UTL_FILE_DIR` database initialization parameter where the dictionary file will be created. The `UTL_FILE_DIR` parameter specifies a directory for PL/SQL I/O.

Below is an example of the `UTL_FILE_DIR` in an initialization parameter file. Once added, the database will need to be restarted for the parameter to take effect.

```
DB_NAME = "CERTDB"

UTL_FILE_DIR = 'C:\CERTDB\LOGMINER'

DB_FILES = 1024
CONTROL_FILES = ("C:\CERTDB\DISK1\control01.ct1",
"C:\CERTDB\DISK2\control02.ct1",
"C:\CERTDB\DISK3\control03.ct1")
```

In this example, the `UTL_FILE_DIR` parameter is set to `'C:\CERTDB\LOGMINER'`. Setting this parameter enables the dictionary file to be created in this directory. Note that multiple directories can be specified by a comma delimiting the directory names.

```
UTL_FILE_DIR = ('C:\CERTDB\LOGMINER', 'C:\CERTDB\DICT')
```

Creating the LogMiner dictionary file

Executing the following command creates a dictionary file called `CERTDBDICT.ORA` in the directory `C:\CERTDB\LOGMINER`.

```
EXECUTE DBMS_LOGMNR_D.BUILD('CERTDBDICT.ORA', 'C:\CERTDB\LOGMINER');
```

Specifying log and archive log files to be analyzed

Once the dictionary file has been created, you can now start identifying the files you wish to analyze. The `DBMS_LOGMNR.ADD_LOGFILE` procedure is used to specify the files. The procedure has three constants that can be used.

- ♦ `DBMS_LOGMNR.NEW` creates a new list for analysis and specifies the first file in that list.
- ♦ `DBMS_LOGMNR.ADDFILE` adds additional files to the list.
- ♦ `DBMS_LOGMNR.REMOVEFILE` removes files from the list.

Execute the following command to create a new list and add the first file.

```
EXECUTE
DBMS_LOGMNR.ADD_LOGFILE('C:\CERTDB\DISK3\LOG4A.RDO', DBMS_LOGMNR.NEW)
```

Execute the following command to add log files to the list.

```
EXECUTE
DBMS_LOGMNR.ADD_LOGFILE('C:\CERTDB\DISK3\LOG3A.RDO', DBMS_LOGMNR.ADDFILE)
```

This command adds online log file `LOG3A.RDO` to the list to be analyzed.

Performing the analysis

With the dictionary file created and redo or archive log files added to the list, you can now perform the analysis. The analysis goes through the files, extracting the SQL statements. The `DBMS_LOGMNR.START_LOGMNR` package is used to perform the analysis. The package has several options for narrowing the search:

<code>StartScn</code>	The starting system change number (SCN) in the range
<code>EndScn</code>	The ending SCN in the range
<code>StartTime</code>	The start time
<code>EndTime</code>	The end time
<code>DictFileName</code>	The dictionary file to use for the analysis

The following command is executed to analyze all transactions in the files:

```
EXECUTE DBMS_LOGMNR.START_LOGMNR(DICTFILENAME=>
  'C:\CERTDB\LOGMINER\CERTDBDICT.ORA')
```

To perform the analysis on a range of SCN numbers, execute the command this way:

```
EXECUTE DBMS_LOGMNR.START_LOGMNR(DICTFILENAME=> 'C:\CERTDB\
LOGMINER\CERTDBDICT.ORA', STARTSCN=>'1234567',
ENDSCN=>'1234589')
```

Viewing the analysis results

The analysis loads the transactions from the log files into the tables used by the V\$LOGMNR_CONTENTS view. Queries can be written against this table to extract information about all transactions or specific ones. Here is a partial list of the columns in the V\$LOGMNR_CONTENTS view.

<i>Column</i>	<i>Description</i>
SCN	System Change Number
TIMESTAMP	Time of transaction
THREAD#	Redo Log Group Number
LOG_ID	Log Sequence Number
XIDUSN	Rollback Segment Number of Transaction
ABS_FILE#	Absolute File Number for changed row
REL_FILE#	Relative File Number for changed row
DATA_BLK#	Data Block Number
DATA_OBJ#	Data Block Object Number
SEG_OWNER	Segment Owner
SEG_NAME	Segment Name
SEG_TYPE	Type of Segment (table, view, and so on)
TABLE_SPACE_NAME	Tablespace
ROW_ID	Rowid
SESSION#	Session number that performed action
SERIAL#	Serial number that performed action
USER_NAME	Username that performed action
ROLLBACK	Rollback request
OPERATION	Type of operation performed

<i>Column</i>	<i>Description</i>
SQL_REDO	The SQL of the operation performed
SQL_UNDO	The undo of the SQL operation performed
INFO	Informational Message
STATUS	Status

The key elements of the view are:

- ♦ The SCN number, which can be used to perform recovery.
- ♦ The TIMESTAMP, which indicates when the transaction occurred.
- ♦ The SEG_OWNER identifies the owner of the segment or table.
- ♦ The SEG_NAME identifies the segment or table the transaction was performed against.
- ♦ The USERNAME indicates the user that performed the transaction.
- ♦ The SQL_REDO, which is the statement executed. This can be used to duplicate transactions on other systems.
- ♦ The SQL_UNDO, which is the undo of the statement executed. This can be used to reverse committed transactions.

The following command could be issued to extract the both the SQL_REDO and SQL_UNDO for all statements made against the COURSES table:

```
SELECT SQL_REDO, SQL_UNDO
FROM V$LOGMNR_CONTENTS
WHERE seg_name = 'COURSES'
```

Finishing LogMiner sessions

The following command should be issued when you are done using LogMiner. This will clear out all the tables used by LogMiner:

```
EXECUTE DBMS_LOGMNR.END_LOGMNR
```

Obtaining information about the LogMiner analysis

There are three primary views that contain information about the LogMiner session and the logs being analyzed:

- ♦ V\$LOGMNR_DICTIONARY contains the LogMiner dictionary file being used.
- ♦ V\$LOGMNR_PARAMETERS contains the parameters used for the session.
- ♦ V\$LOGMNR_CONTENTS contains the contents of the log files being analyzed.

Features and restrictions of LogMiner

The initial release of LogMiner has some restrictions that you need to be aware of. Some of these restrictions will be lifted in future releases of the utility.

- ♦ The contents of the V\$LOGMNR_CONTENTS view is only visible to the session performing the analysis. If you want to keep a record of this information, create a permanent table to store the information and then select the contents of the V\$LOGMNR_CONTENTS view into this permanent table before completing the LogMiner session.

```
CREATE TABLE LOG_CONTENTS as
SELECT * FROM V$LOGMNR_CONTENTS
```

- ♦ Only one row is recorded per record.
- ♦ DML statements are only supported against scalar datatypes.
- ♦ DDL statements are recorded as DML statements against the data dictionary.
- ♦ SQL statements on chained rows cannot be reconstructed.
- ♦ Objects not listed in the LogMiner dictionary file display HEX values in the V\$LOGMNR_CONTENTS views and not the segment names.

Key Point Summary

In preparation for the “Oracle8i: Architecture and Administration” exam, keep these points regarding Redo Log files and recoverability in mind:

- ♦ The online and archive redo log files are critical for both the operation and recoverability of a database. They are primarily used by Oracle for recovery. Without at least two valid groups and one valid member in each of the groups, the database will fail.
- ♦ Multiplexing online redo log groups protects the database from a single point of failure. Try to ensure that when multiplexing, members from the same group are placed on different disks (see Figure 7-1). This practice will also protect the database against a single point of failure, the disk. As long as LGWR can access one valid member per group, the database will continue processing.
- ♦ The online redo log groups should all be the same size, as this promotes consistency in database performance. The size is a critical factor in the database because the frequency of checkpoints is directly correlated.
- ♦ When the current online redo log group fills, a log switch occurs. At every log switch, a checkpoint occurs.

- ♦ If the log files are too small, there will be frequent checkpoints. The inverse is true when the files are too big. Information about online redo log groups and members is available in three primary dynamic performance views, V\$THREAD, V\$LOG, and V\$LOGFILE. When correctly configured, the files need little maintenance and the database performs consistently and efficiently.
- ♦ A checkpoint is the synchronization event for the database. It ensures that all changes made against the database have been written to disk. It also updates all database files (control files, data files, and the current redo log files) with a log sequence number.
- ♦ The LSN is assigned to every file in a redo log group when it becomes the current group. The LSN number is used by Oracle during recovery. This is how it knows when files are not synchronized with the database.
- ♦ Three database initialization parameters can trigger checkpoint events before a log switch. They are LOG_CHECKPOINT_INTERVAL, LOG_CHECKPOINT_TIMEOUT, and FAST_START_IO_TARGET.
- ♦ If management can tolerate zero data loss, it is critical that the database run in ARCHIVELOG mode. When running in ARCHIVELOG mode, Oracle preserves a copy of the redo log files at every log switch. Backed-up online redo log files are called archive redo log files.
- ♦ The archiving can be done automatically by specifying the database initialization parameter LOG_ARCHIVE_START to true. When the database is running in NOARCHIVELOG mode, online redo log files will be overwritten when a log switch makes them the current group again.
- ♦ A new feature for Oracle8i is LogMiner. This utility analyzes either online redo or archive log files. Once analyzed, all statements, the undo operations for those statements, the time the statements were executed, as well as the user who performed the execution, can be extracted. The utility can be used for recovery purposes as well as compiling statistical information for the database.



STUDY GUIDE

Now that you have learned about the log files, you should test your understanding by reviewing the assessment questions and performing the exercises below.

Assessment Questions

1. What is the primary use for Log Files in Oracle?
 - A. Read consistency
 - B. Performing transaction rollbacks
 - C. Performing recovery operations
 - D. Tracking changes to the database
 - E. Logging user activity
2. What is the minimum number of redo log groups required by Oracle?
 - A. They are not required for all databases.
 - B. 1
 - C. 2
 - D. 3
 - E. 4
3. What is the minimum number of redo log members required per group?
 - A. 1
 - B. 2
 - C. 3
 - D. 4
 - E. Groups do not require members.
4. LGWR will always write to the current redo log files after every
 - A. Update Statement
 - B. Insert Statement
 - C. Delete Statement
 - D. Any DML or DDL statement
 - E. Commit

5. Which database creation parameter limits the maximum number of redo log groups per database?
 - A. There is no limit.
 - B. MAXLOGFILES
 - C. MAXLOGGROUPS
 - D. MAXGROUPS
 - E. LOG_FILES
6. What database creation parameter limits the maximum number of redo log members per group?
 - A. MAXLOGMEMBERS
 - B. MAXLOGFILES
 - C. LOG_FILES
 - D. MAXMEMBERS
 - E. There is no limit.
7. Which of the following is not true about the LGWR process?
 - A. LGWR always performs a write after any DML or DDL statement.
 - B. LGWR always performs a write after a commit.
 - C. LGWR always writes when the redo log buffer becomes one third full from a single transaction.
 - D. LGWR always performs a write prior to DBWn writing out dirty buffers from the database buffer cache.
 - E. LGWR always performs a write when a timeout occurs.
8. Which of the following is not true about checkpoints?
 - A. Checkpoints occur at every log switch.
 - B. Checkpoints occur during database shutdown except shutdown abort.
 - C. Checkpoints write all dirty buffers from the database buffer cache to the data files.
 - D. Checkpoints occur at every commit.
 - E. Checkpoints can be controlled through database initialization parameters.
9. Which statement is not true about a database running in ARCHIVELOG mode?
 - A. Log files are backed up at every log switch.
 - B. Complete database recovery is possible.
 - C. Redo Log files cannot be overwritten until they have been archived.
 - D. Checkpoints are no longer required.
 - E. The archive log files can be used by the LogMiner utility.

10. What statement is true about the LOG_ARCHIVE_START database initialization parameter?
- A. When set to TRUE, the database is running in ARCHIVELOG mode.
 - B. When set to TRUE, checkpoints are no longer required.
 - C. When set to FALSE, the database is switched to NOARCHIVELOG mode.
 - D. When set to TRUE, the archiver (ARCn) process automatically copies online redo log files at every log switch.
 - E. When set to FALSE, the archiver (ARCn) process automatically copies online redo log files at every log switch.
11. Which dynamic performance view would you query to obtain the name and location of redo log files?
- A. V\$LOG
 - B. DBA_LOG_FILES
 - C. V\$THREAD
 - D. V\$LOGFILE
 - E. V\$LOGMEMBER
12. What does a status of ACTIVE in the V\$LOG view mean about that group?
- A. The group is currently being written to by LGWR.
 - B. It is the current group.
 - C. It is being used for recovery.
 - D. The group is needed for crash recovery and cannot be overwritten.
 - E. The members of the group have been archived.
13. Which command can be used to force manual checkpoints?
- A. ALTER SYSTEM FORCE CHECKPOINT
 - B. ALTER DATABASE FORCE CHECKPOINT
 - C. ALTER SYSTEM CHECKPOINT FORCE
 - D. ALTER DATABASE CHECKPOINT
 - E. ALTER SYSTEM FORCE LOG SWITCH
14. Which command can be used for force manual log switches?
- A. ALTER DATABASE SWITCH LOGFILE
 - B. ALTER SYSTEM SWITCH LOGFILE
 - C. ALTER DATABASE LOGSWITCH
 - D. ALTER SYSTEM LOGSWITCH
 - E. ALTER SYSTEM FORCE LOG SWITCH

15. Which command is used to add a new online redo log group to the database?
- A. ALTER DATABASE ADD LOGGROUP
 - B. ALTER SYSTEM ADD LOGGROUP
 - C. ALTER DATABASE ADD LOGFILE
 - D. ALTER SYSTEM ADD LOGFILE
 - E. ALTER DATABASE ADD LOGMEMBER
16. Which of the following statements is true about the size of redo log files?
- A. The size must be specified when adding log files to existing groups.
 - B. The size of the log files does not need to be the same for all files in the same log group.
 - C. The size of the files can be changed dynamically with an ALTER DATABASE LOGFILE RESIZE.
 - D. The size for the files can only be specified at the group level.
 - E. The size of the files has no impact on the frequency of checkpoints.
17. Which two statements are not true about multiplexed online redo log files.
- 1. The members belonging to the same group should reside on separate physical disks.
 - 2. You eliminate a single point of failure in the database.
 - 3. There must only be two members for each group.
 - 4. Members of the same group must all be the same size.
 - 5. Multiplexing eliminates the need to run the database in ARCHIVELOG mode.
- A. 3 and 5
 - B. 3 and 4
 - C. 4 and 5
 - D. 1 and 3
 - E. 2 and 5
18. Which statement is true about dropping online redo log groups?
- A. The current group cannot be dropped.
 - B. Only the current group can be dropped.
 - C. Dropping groups drops all members of the group.
 - D. Dropping groups removes the files from the OS.
 - E. Both A and C.

19. Which statement is true about dropping online redo log members?
- A. Members from the current group cannot be dropped.
 - B. Dropping a member marks the GROUP invalid.
 - C. Dropping a member forces a log switch.
 - D. Members cannot be dropped, only groups.
 - E. A member can only be dropped if there are two valid members for the group.
 - F. Both A and E.
20. Which statement is true about the LogMiner utility?
- A. LogMiner can be used to analyze only redo log files.
 - B. LogMiner can be used to analyze both redo log and archive log files.
 - C. LogMiner only works with Oracle 8.1 log and archive log files.
 - D. LogMiner can be used only with archive log files.
 - E. LogMiner is used to analyze data files.
21. Which dynamic performance view contains the contents of the redo and archive log files being analyzed?
- A. V\$LOGMNR
 - B. V\$LOGMINER
 - C. V\$LOGMNR_ANALYSIS
 - D. V\$LOGMINER_CONTENTS
 - E. V\$LOGMNR_CONTENTS

Scenarios

1. You have just been hired as a consultant by BUY IT ON THE WEB Ltd. The company sells products over the Internet. The company is just starting out, and they want to ensure the database is protected against failures. They have state-of-the-art hardware with a 2 RAID 5 configuration, a mirrored system drive, as well as four other nonmirrored drives. Drive C is the mirrored system drive, drives D and E are the RAID 5 devices, and drives F through I are the nonmirrored drives. The company has placed the operating system, Oracle, and its applications on drive C. All their data files are on drive D and the indexes are on drive E. The database is running in NOARCHIVELOG mode, and there are two redo log groups on the system drive C with one redo log member per group. The company has little Oracle knowledge and is concerned about data loss. Since they are Web-based, transactions cannot be re-entered. What recommendations would you make to them, if any, about the configuration of the redo log files and about archiving?

- A. What recommendations would you make about the configuration of the redo log files? Be sure to include the location of the files in your answer.
- B. What recommendations would you make about archiving? Be sure to give reasons for your answer.
2. Your company has just acquired another one with 200 employees. This is going to double the users and the volume of transactions in your database. Management has agreed to upgrade the server, doubling the memory, disk, and CPU power. There are currently no complaints about performance and log switches are happening every ten minutes on average. Management is concerned about maintaining the current performance levels with the new users. They figure that this will help smooth the transition phase for both the current and new employees.
- A. What factors should you consider about the redo log files when doing the database upgrade?
- B. What factors should you consider about checkpoints?
3. You are hired as a consultant by Guru Database Marketing Ltd. The database is suffering from poor performance. After some investigation, you discover that there are several LGWR error messages in the ALERT file. The error message indicates that LGWR is waiting for checkpoints to complete. They also ask you to ensure that the database is protected against failure. This is the information that you derived from the various dynamic performance views:

```
SQL> SELECT * FROM V$LOGFILE;
```

GROUP#	STATUS	MEMBER
3		C:\CERTDB\DISK3\LOG3A.RDO
4		C:\CERTDB\DISK3\LOG4A.RDO

```
SQL> SELECT group#,bytes,members,status FROM V$LOG;
```

GROUP#	BYTES	MEMBERS	STATUS
3	1048576	1	CURRENT
4	1048576	1	INACTIVE

```
SQL> ARCHIVE LOG LIST
```

Database log mode	Archive Mode
Automatic archival	Enabled
Archive destination	C:\CERTDB\DISK6\archive
Oldest online log sequence	210
Next log sequence to archive	211
Current log sequence	211

- A. What recommendations would you make to reduce the waits by the LGWR process?
- B. What recommendations would you make about the company's concerns about protecting against failure?

Lab Exercises

Lab 7-1 Identifying the Log File and Archiving Configuration

1. Connect to your instance using Server Manager line mode (svrmgrl) as a user with SYSDBA privileges.
2. Determine the number of groups and names and locations of members that belong to each one of the groups. Query the V\$LOG and V\$LOGFILE views for this information. Record the results in the table that follows.

<i>Group Number</i>	<i>Group Status</i>	<i>Member Name and Location</i>
---------------------	---------------------	---------------------------------

-
3. How many groups do you have?
 4. Are the groups multiplexed?
 5. The next lab is going to have you add additional groups and multiplex the members. There are going to be four groups each with three members. Members for the same group are going to be spread across DISK3, DISK4, and DISK5. Add the group and member information to the table above. You can omit the Group Status for the groups you wish to add.
 6. Determine if the database is running in ARCHIVELOG mode and whether automatic archiving is enabled.

Lab 7-2 Adding Online Redo Log Groups and Members

1. Using Server Manager line mode (svrmgrl), connect to your instance as a user with SYSDBA privileges.
2. Ensure that GROUP 3 is the current group. This can be done by manually forcing log switches.
3. Create a new group, GROUP 4, with one member. The size for all members in this group should be one megabyte. Locate the member on DISK3. If you received the message, “Database altered,” then the command was successful. If it failed, ensure the pathnames are correct and that your NT or UNIX account has privileges on the directory.
4. Query the appropriate dynamic performance view to ensure their creation.
5. Multiplex all the groups. Place one member called REDO0?B.LOG, where the ? is substituted for the GROUP NUMBER, on DISK4 and the other called REDO0?C.LOG on DISK5 again where the ? is substituted for the GROUP number. The files for GROUP 1 will be called REDO01B.LOG and REDO01C.LOG. If you received the message, “Database altered,” then the command was successful. If it failed, ensure the pathnames are correct and that your NT or UNIX account has privileges on the directory.
6. Query the dynamic performance views to ensure their creation. The status for the new members will be STALE. This is normal and will change when the GROUP becomes the current group or — in the case of the current group — when that group becomes the current group again.

Lab 7-3 Dropping Log Groups and Members

1. Connect to your instance using Server Manager line mode (svrmgrl) as a user with SYSDBA privileges.
2. Ensure that GROUP 1 is the current group. If it is not, issue the ALTER SYSTEM SWITCH LOGFILE command enough times to make it the current group.
3. Try to drop GROUP 1.
4. What error message did you receive and why?
5. Issue the ALTER SYSTEM SWITCH LOGFILE command once so that GROUP 2 is now the current group and try to drop GROUP 1 again.
6. Why did it succeed this time?
7. Try to drop member ‘C:\CERTDB\DISK4\REDO02b.LOG’.
8. What error message did you get and why?
9. Issue the ALTER SYSTEM SWITCH LOGFILE command once and try to drop the member again.

10. Why did it succeed this time?
11. Ensure that GROUP 3 is the current group. If not issue the appropriate commands to make it the current group.
12. Relocate all members named REDO0?B.LOG where the ? is the GROUP number to DISK6. (There are two possible ways to accomplish this.)


 Tip

Since GROUP 3 is the current group you should do this one last. After you have moved the other three, force a log switch so that GROUP 3 is no longer the current group before attempting to relocate its members.

Lab 7-4 Putting the Database in ARCHIVELOG Mode and Configuring for Automatic Archiving

1. Connect to your instance using Server Manager line mode (svrmgrl) as a user with SYSDBA privileges.
2. Shutdown the database using the immediate option and be patient as this may take some time.
3. Modify the database initialization parameter file and add the following lines:

```
# ARCHIVE PARAMETERS SECTION
LOG_ARCHIVE_START = TRUE
LOG_ARCHIVE_DEST = C:\CERTDB\DISK6
LOG_ARCHIVE_FORMAT = ARCH%s.ARC
# END ARCHIVE PARAMETERS SECTION
```



You should create sections in the parameter file so that it is easy to identify what the parameters reference. If a line in the file starts with a #, it is treated as a comment line during startup.

4. Get the database to a mount state by issuing the startup pfile=???? Command. If the database does not start, then either the parameters were not specified correctly or the name for the pfile did not point to the correct file. Fix the problem and attempt the startup command again.
5. Verify that the database is at the mount state by running the following command:

```
SELECT STATUS FROM V$INSTANCE;
STATUS
-----
MOUNTED
```

6. Issue the following command to place the database in ARCHIVELOG mode:

```
ALTER DATABASE ARCHIVELOG
```

7. Open the database using the following command:

```
ALTER DATABASE OPEN
```



After placing the database in ARCHIVELOG mode, you must perform a full database backup. This insures future recoverability if required.



8. Verify that the database is now running in ARCHIVELOG mode.
9. Shutdown the database using the immediate option.
10. Modify the database initialization parameter file and comment out the lines pertaining to archiving.

```
# ARCHIVE PARAMETERS SECTION
# LOG_ARCHIVE_START = TRUE
# LOG_ARCHIVE_DEST = C:\CERTDB\DISK6
# LOG_ARCHIVE_FORMAT = ARCH%s.ARC
# END ARCHIVE PARAMETERS SECTION
```
11. Start the database.
12. Verify that it is not in ARCHIVELOG mode.



Placing the database in ARCHIVELOG mode is not covered on the exam.

Answers to Chapter Questions

Chapter Pre-Test

1. The online redo log files keep a chronological copy of all changes made to the database. Any DML and most DDL operations are logged in the online redo log files in the order in which they occurred.
2. Redo log groups include redo log members. Redo log members are the physical files on disk that belong to a redo log group. Oracle writes to all members of a redo log group at the same time.
3. The redo log files are written to by the LGWR process when a transaction commits, the redo log buffers is one-third filled, when there is more than one megabyte of changed records in the redo log buffer, every three seconds when a timeout occurs, and before the DBWn process writes changed blocks to the data files.
4. If your database is configured to run in ARCHIVELOG mode, when a redo log file group is switched from, the archiver process copies the redo log file in one or more archive log locations. In this way, should a recovery be required, you will have a chronological record of changes made to the database and can recover up to the point of failure.

5. To multiplex your redo log files ensure that more than one member exists for each redo log group. Members of the same group should be on separate physical disks to ensure that a disk loss will not prevent writes to the redo log files. You can multiplex redo log files when the database is created by specifying more than one file for each redo log group, or after the database is created by issuing the ALTER DATABASE ADD LOGFILE MEMBER command.
6. The three INIT.ORA parameters that control checkpoints are LOG_CHECKPOINT_INTERVAL specifying the maximum number of O/S blocks that can be written to a redo log file before forcing a checkpoint, LOG_CHECKPOINT_TIMEOUT, specifying the maximum number of seconds that can elapse between checkpoints, and FAST_START_IO_TARGET, specifying the maximum number of blocks that should be read to perform recovery on an instance restart.
7. Oracle must perform a checkpoint whenever a log switch takes place. Therefore, smaller logfiles will cause Oracle to perform more frequent log switches that may increase the frequency of checkpoints.
8. The three dynamic performance views that include information about logfiles are V\$THREAD, indicating which logfile group is currently being written to, V\$LOG, which shows which groups exist in the database and when and what was last written to them, and V\$LOGFILE, which shows the status of and other information about each member in each group.
9. To determine whether the database is in ARCHIVELOG mode, query the V\$INSTANCE view for the value of the ARCHIVER column — STARTED means that the archiver process is started. You can also query the V\$DATABASE view and verify the value of the ARCHIVE_MODE column, which will read ARCHIVELOG if the mode is set, otherwise NOARCHIVELOG. You can also issue the command “ARCHIVE LOG LIST” in SQL*Plus or Server Manage line mode.
10. The LOGMINER utility can be used to extract the redo and undo statements to reapply changes found in the log files or to reverse those changes.

Assessment Questions

1. C. Performing database recovery is the primary use for redo log files. Both instance and database. For information see the “Overview of Online Redo Log Files” section.
2. C. The minimum number of redo log files required by Oracle is two. Refer to the “Planning Redo Log Files” section for more information.
3. A. The minimum number of redo log files per group is one. You should, however, have at least two members per group to protect against failures. For more information, refer to the section on “Planning Redo Log Files” section.

4. **E.** LGWR will always write to the current redo log file(s) during a commit. **D** is partially correct as LGWR will perform a write after every DDL statement, but the reason for write is because Oracle also performs implicit commits after all DDL and DCL statements. Therefore, it is still caused by the commit.
5. **B.** The MAXLOGFILES parameter controls the maximum number of redo log groups per database. The parameter can only be specified at database creation time or when control files are rebuilt. For more information, refer to section “Planning Redo Log Files.”
6. **A.** The MAXLOGMEMBERS parameter controls the maximum number of redo log files per group. The parameter can only be specified at database creation time or when control files are rebuilt. For more information, refer to section “Planning Redo Log Files.”
7. **A.** Oracle does not guarantee that redo log information will be written after DML statements. For more information, refer to section “Using Online Redo Log Files”
8. **D.** Checkpoints do not happen at commits. For more information, refer to section “Using Online Redo Log Files.”
9. **D.** Checkpoints are required for all Oracle databases, regardless of the ARCHIVELOG mode the database is run in. Refer to the section on “Using Online Redo Log Files” for more information.
10. **D.** The LOG_ARCHIVE_START parameter starts the archiver process. If the parameter is not set to TRUE then the DBA must do manual archiving. It has no impact on archiving or checkpoints. For more information, refer to section “Using Online Redo Log Files.”
11. **D.** The V\$LOGFILE dynamic performance view contains the name of the log files, the group the files belong to as well as the status of the files. The section on “Maintaining Redo Log Files” contains information about the V\$LOGFILE, V\$LOG and V\$THREAD views.
12. **D.** The status of ACTIVE indicates that the group is still required for crash recovery. The files cannot be overwritten because Oracle could not guarantee that all committed transactions could be recovered in the event of a failure. Refer to the section on “Maintaining the Redo Log Files” for more information.
13. **A.** The ALTER SYSTEM CHECKPOINT forces manual checkpoints. More information about checkpoints can be found in the “Using Online Redo Log Files.”
14. **B.** The ALTER SYSTEM SWITCH LOGFILE forces manual log switches. The “Using Online Redo Log Files” section has more information about log switches.
15. **C.** You use the ALTER DATABASE ADD LOGFILE command to add new online redo log groups. Refer to the “Managing Redo Log Files” section for more information.

16. **D.** The size of redo log files can only be specified at the group level. The files are critical to several components of the database including controlling checkpoints. Section “Using Online Redo Log Files” has more information about the size of the redo log files.
17. **A.** The deceiving part to this question is number 3. Multiplexing implies that there is more than one file per group not that there is only two files per group. It is common to have 3 to 4 members per redo log group. Number 5 is also not true as multiplexing online redo log files does not eliminate the need to run a database in ARCHIVELOG mode. ARCHIVELOG mode protects the database from media or database failure. For more information, refer to section “Managing Redo Log Files.”
18. **E.** Both A and C are true as you cannot drop the current online redo log group and dropping redo log groups or members in Oracle does not remove the files from the OS. The files would need to be removed at the OS level. Refer to the section on “Maintaining Redo Log Files” for more information.
19. **F.** Both A and E are correct. A is true because dropping online redo log files can only be performed against the non-current group. E is true as well because there must always be one member per group, therefore, if there is only one member for a group, dropping that member would leave none. Refer to the section on “Maintaining Redo Log Files” for more information.
20. **B.** The LogMiner utility can be used to analyze both redo and archive log files. This is why A and D are not valid. Answer C is not correct as the LogMiner utility can be used on both Oracle 8.0 and 8.1 files. Refer to the section on “Using LogMiner” for more information.
21. **E.** The V\$LOGMNR_CONTENTS view contains the data from the redo and archive log files being analyzed. Refer to the section on “Using LogMiner” for more information.

Scenarios

1. The first recommendation about the redo log files should be that the groups are not mirrored or multiplexed. The groups should be multiplexed to prevent against the loss of a redo log file, which could result in the loss of data. When multiplexing the files, the members should be placed on separate drives so that the loss of a drive does not result in the loss of all members from the same group. Drives F and G could be used for the redo log files. Place one member from each of the groups on drive F and the other members on drive G. This multiplexes the files and also protects against the loss of a disk. By placing the files on drives F and G, the LGWR process will not be competing against the DBWn process writing to the data files on drive E.
2. The database should be running in ARCHIVELOG mode. Since the database is Web based, making it impossible to reenter data, this must be protected by archiving the redo log files. If the database is not running in ARCHIVELOG mode, a media or database failure will result in data loss. The archive files should be placed on drive H, this will prevent the ARCn process from competing with the LGWR process when archiving is happening.

3. The additional users and transactions are going to increase the transaction volume. This means that the redo log files are going to fill much faster. When the log files fill, log switches and checkpoints occur. Checkpoints that occur too frequently can result in poor database performance. Checkpoints are generally resource intensive, and if they happen too frequently the checkpoint resources take resources away from other Oracle processes, causing the potential performance problems. You should also recommend adding more redo log groups to give the checkpoint process more time to complete. If a checkpoint is triggered and the checkpoint started on the redo log group that is being switched to has not completed, then Oracle will wait for that checkpoint to complete before it continues processing.
4. The information from the dynamic performance views reveals that the database only has two redo log groups. The groups are both one megabyte in size. The waits by the LGWR process indicate that the LGWR is waiting for a checkpoint to complete. When Group 3 fills and forces a log switch, a checkpoint is started. If that checkpoint does not complete before Group 4 fills and switches back to Group 3, then Oracle must wait. The easiest fix for this problem is to add additional log groups. This should give the checkpoints more time to complete as now three log switches will happen before the original checkpoint must have completed. You may also consider increasing the size of the redo log groups, as this will also increase the time between checkpoints. The company's concerns about protecting the database from failure should also be addressed. The database is running in ARCHIVELOG mode and the archiver process is running, so the database is protected against media and database failure, however, the redo log groups should be multiplexed. If the files are multiplexed then the redo log groups are protected against failure. If the files are not multiplexed then losing a log file could result in the loss of data.

Lab Solutions

Lab 7-1

2.

```
SELECT * FROM V$LOGFILE;  
SELECT * FROM V$LOG;
```

3. There are 3 groups.
4. The groups are not multiplexed.
5. ARCHIVE LOG LIST

Lab 7-2

2.

```
SELECT CURRENT_GROUP# FROM V$THREAD;
```

If GROUP 3 is not the current group, use the ALTER SYSTEM SWITCH LOGFILE command until it is the current group. Recheck the V\$THREAD view to ensure that GROUP 3 is the current.

3.

```
ALTER DATABASE ADD LOGFILE GROUP 4 'C:\CERTDB\DISK3\
REDO01A.LOG' SIZE 1M;
```

4.

```
SELECT GROUP#,BYTES,MEMBERS FROM V$LOG WHERE GROUP#=4;
SELECT GROUP#,MEMBER FROM V$LOGFILE WHERE GROUP#=4;
```

5.

```
ALTER DATABASE ADD LOGFILE MEMBER 'C:\CERTDB\DISK4\
REDO01B.LOG', 'C:\CERTDB\DISK5\REDO01C.LOG' TO GROUP 1 ;
ALTER DATABASE ADD LOGFILE MEMBER 'C:\CERTDB\DISK4\
REDO02B.LOG', 'C:\CERTDB\DISK5\REDO02C.LOG' TO GROUP 2 ;
ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK4\REDO03B.LOG', 'C:\CERTDB\DISK5\REDO03C.
LOG' TO GROUP 3 ;
ALTER DATABASE ADD LOGFILE MEMBER 'C:\CERTDB\DISK4\
REDO04B.LOG', 'C:\CERTDB\DISK5\REDO04C.LOG' TO GROUP 4 ;
```

6.

```
SELECT GROUP#,STATUS,MEMBER FROM V$LOGFILE;
```

Lab 7-3

2.

```
SELECT CURRENT_GROUP# FROM V$THREAD;
```

If GROUP 3 is not the current group, use the ALTER SYSTEM SWITCH LOGFILE command until it is the current group. Recheck the V\$THREAD view to ensure that GROUP 3 is the current.

3.

```
ALTER DATABASE DROP LOGFILE GROUP 1;
```

```
ALTER DATABASE DROP LOGFILE GROUP 1
```

*

```
ERROR at line 1:
ORA-01623: log 1 is current log for thread 1 - cannot drop
ORA-00312: online log 1 thread 1: 'C:\CERTDB\DISK3\
REDO01A.LOG'
```

4. The error is raised because you cannot drop the current group.

5.

```
ALTER SYSTEM SWITCH LOGFILE;
ALTER DATABASE DROP LOGFILE GROUP 1;
```

6. It succeeds now because GROUP 1 is no longer the current group.

7.

```
ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK4\
REDO02B.LOG';
```

8.

```
ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK4\
RED002B.LOG'
*
ERROR at line 1:
ORA-01609: log 2 is the current log for thread 2 - cannot
drop members
ORA-00312: online log 2 thread 2: 'C:\CERTDB\DISK3\
RED002A.LOG'
ORA-00312: online log 2 thread 2: 'C:\CERTDB\DISK4\
RED002B.LOG'
ORA-00312: online log 2 thread 2:
'C:\CERTDB\DISK5\RED002C.LOG'
```

The error is received because you cannot drop members of the current group.

9.

```
ALTER SYSTEM SWITCH LOGFILE;
ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK4\
RED002B.LOG';
```

10. It succeeds because GROUP 2 is no longer the current group.

11.

```
ALTER SYSTEM SWITCH LOGFILE;
SELECT CURRENT_GROUP# FROM V$THREAD;
```

12. Solution (1)

```
ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK4\
RED001B.LOG';

ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK4\
RED002B.LOG';

ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK4\
RED004B.LOG';

ALTER SYSTEM SWITCH LOGFILE;

ALTER DATABASE DROP LOGFILE MEMBER 'C:\CERTDB\DISK4\
RED003B.LOG';

ALTER DATABASE ADD LOGFILE MEMBER
C:\CERTDB\DISK6\RED001B.LOG' TO GROUP 1;

ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK6\RED002B.LOG' TO GROUP 2;

ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK6\RED003B.LOG' TO GROUP 3;

ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK6\RED004B.LOG' TO GROUP 4;
```


Tip

The previous four commands could have been written as follows:

```
ALTER DATABASE ADD LOGFILE MEMBER
'C:\CERTDB\DISK6\RED001B.LOG' TO GROUP 1,
'C:\CERTDB\DISK6\RED002B.LOG' TO GROUP 2;
'C:\CERTDB\DISK6\RED003B.LOG' TO GROUP 3;
'C:\CERTDB\DISK6\RED004B.LOG' TO GROUP 4;
```

Solution (2) This is just another method for performing this task.

```
HOST COPY C:\CERTDB\DISK4\RED001B.LOG C:\CERTDB\DISK6\
RED001B.LOG
```

```
HOST COPY C:\CERTDB\DISK4\RED002B.LOG C:\CERTDB\DISK6\
RED002B.LOG
```

```
HOST COPY C:\CERTDB\DISK4\RED002B.LOG C:\CERTDB\DISK6\
RED002B.LOG
```

```
ALTER DATABASE RENAME FILE 'C:\CERTDB\DISK4\RED001B.LOG' TO
'C:\CERTDB\DISK6\RED001B.LOG';
```

```
ALTER DATABASE RENAME FILE 'C:\CERTDB\DISK4\RED002B.LOG' TO
'C:\CERTDB\DISK6\RED002B.LOG';
```

```
ALTER DATABASE RENAME FILE 'C:\CERTDB\DISK4\RED004B.LOG' TO
'C:\CERTDB\DISK6\RED004B.LOG';
```

```
ALTER SYSTEM SWITCH LOGFILE;
```

```
HOST COPY C:\CERTDB\DISK4\RED003B.LOG C:\CERTDB\DISK6\
RED003B.LOG
```

```
ALTER DATABASE RENAME FILE 'C:\CERTDB\DISK4\RED003B.LOG' TO
'C:\CERTDB\DISK6\RED003B.LOG';
```

Lab 7-4

2.

```
SHUTDOWN IMMEDIATE
```

3. Edit the parameter file.

4. STARTUP MOUNT PFILE=C:\CERTDB\INITCERTDB.ORA

5.

```
SELECT LOG_MODE FROM V$DATABASE;
```

6.

```
ALTER DATABASE ARCHIVELOG;
```

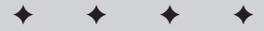
7.

```
ALTER DATABASE OPEN;
```

Managing Tablespaces and Datafiles

EXAM OBJECTIVES

- ◆ Managing Tablespaces and Datafiles
 - Distinguish the different types of temporary segments
 - Create tablespaces
 - Change the size of tablespaces
 - Allocate space for temporary segments
 - Change the status of tablespaces
 - Change the storage settings of tablespaces
 - Relocate tablespaces



CHAPTER PRE-TEST

1. What is the role of a tablespace in the database?
2. What is the physical structure of a database?
3. What is the logical structure of a database?
4. How can tablespaces be classified?
5. How can you create a tablespace?
6. How can you make a tablespace temporarily unavailable?
7. How can you prevent changes to data in a tablespace?
8. How can you increase the size of a tablespace?
9. How can you change the storage settings for a tablespace?
10. How can you enforce a minimum size for all extents in a tablespace?
11. How can you relocate a non-SYSTEM tablespace?
12. How can you relocate the SYSTEM tablespace?
13. What are temporary segments?

A database is not what it seems. When you look at a database, you see tables, columns, and rows. This is not at all the way data is actually stored—the real, physical structure of the database is quite different. In this chapter, you will examine the physical and logical structure of the database and especially the concept that ties the two together—the tablespace. You will learn about how to create different types of tablespaces and how to modify them to suit your needs. You will find out about the different types of segments that can “live” in different types of tablespaces, and what they can be used for. At the end of the chapter, as usual, you will get to practice what you learned with hands-on lab exercises, answer a few exam-style questions, and examine some scenarios that will require a good understanding of the topic.

The Oracle Database Storage Hierarchy

The Oracle database has two distinct sides: physical and logical. The hierarchy can be expressed as a model, as shown in Figure 8-1.

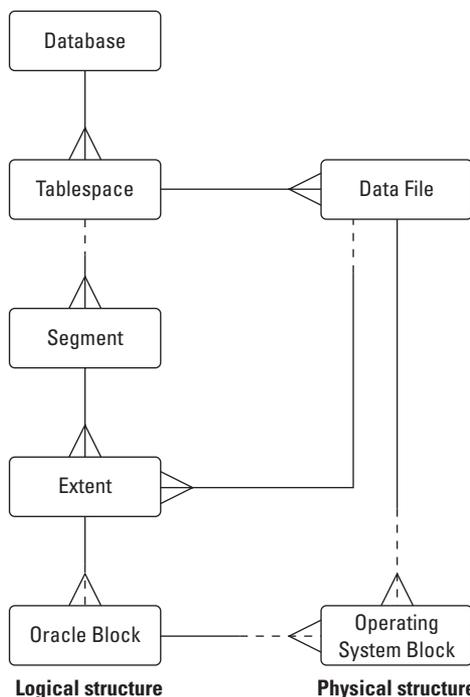


Figure 8-1: The entity relational model of an Oracle database

You can read the model like this: Every box is a component of the database (except the database itself). Every line is a relationship. If the line is broken, the relationship is optional; if it is solid, the relationship is mandatory. The “crow’s foot” means “one or more”; otherwise, it is “one.”

Here is how this can be read:

- ◆ Every database must consist of one or more tablespaces. Every tablespace must belong to one and only one database.
- ◆ Every tablespace must consist of one or more datafiles. Each datafile must belong to one and only one tablespace.
- ◆ Every datafile must consist of one or more operating system blocks. Each operating system block may belong to one and only one datafile.
- ◆ Every tablespace may contain one or more segments. Every segment must exist in one and only one tablespace.
- ◆ Every segment must consist of one or more extents. Each extent must belong to one and only one segment.
- ◆ Every extent must consist of one or more Oracle blocks. Each Oracle block may belong to one and only one extent.
- ◆ Every extent must be located in one and only one datafile. The space in the datafile may be allocated as one or more extents.
- ◆ Every Oracle block must consist of one or more operating system blocks. Every operating system block may be part of one and only one Oracle block.

The left side of the diagram illustrates the logical structure of the database; the right side shows the physical structure. Let us examine the physical structure first.

Physical components

The physical structure is what the operating system sees when “looking” at an Oracle database. It is really quite simple: one or more datafiles. Each file stored in a file system is composed of operating system blocks, or “clusters,” or “allocation units” — every operating system will call them something different. In order to keep the name the same, Oracle chose the term “operating system block” and is sticking with it. So, the physical structure of the database is datafiles composed of operating system blocks.

Datafiles

The datafiles are the actual operating system files that hold data. The first one got created when you created the database. The database can have more than one tablespace, each tablespace consisting of one or more datafiles. In this way, you can spread the load between different disk drives, as well as separate segments that have different characteristics — extent sizes, activity levels, and administrative requirements.

The datafiles consist of the header and the space that can be used to allocate extents to segments. In effect, most datafiles have three parts: the header, the extents (allocated space), and free (unallocated) space.

The header of a datafile identifies it as part of a database and stores specifics of that datafile: which tablespace it belongs to and the last checkpoint performed. This way, Oracle can check that all files are synchronized when it starts up. If it detects that one of the files is older than the rest (or all files are older than the control file), it assumes that the file was restored from a backup and prompts you to perform recovery.

The rest of the file is extents and free space. You normally manage both through the tablespace and not directly through the datafile itself.

Characteristics of datafiles include:

- ♦ A datafile can belong to one and only one tablespace.
- ♦ A datafile consists of operating system blocks.
- ♦ A datafile can be resized manually or automatically.
- ♦ A datafile can be taken offline, as long as it does not belong to the SYSTEM tablespace.

Operating system blocks

The operating system blocks (don't confuse them with the database blocks!) are the minimum allocation unit for the file system. Each file system has its own minimum and default size, and it often varies with the hard disk geometry and size—as well as the file system used to format the drive. Most systems will allow you to configure the block size when formatting the drive. It is a good idea to keep the operating system block size the same as the Oracle block size. So, if you are planning to add a drive in order to keep Oracle datafiles on it, and your database block size is 8KB, you may want to format the drive using 8KB blocks. This way, for every Oracle IO request, the operating system only needs to retrieve one block. You may want to look at your operating system documentation for information on the possible block sizes and how to format a drive using a specific block size.

Logical components

Most of the time, you will be looking at the database from the logical point of view. You will see tablespaces, segments, extents, and Oracle blocks, and only rarely worry about datafiles and operating system blocks. So, let's examine the logical structure of the database in a little more detail.

The logical structure of the database is the left side of the model in Figure 8-1. To remind you: The database is composed of tablespaces, which contain segments of different types. Each segment consists of extents. An extent is a group of contiguous Oracle blocks.

Database

The database is the highest and final unit of organization for Oracle. It is self-contained and consists of at least one (preferably more) tablespace. In order to access it, it must be opened by an instance.

Tablespace

The tablespace is the link between the logical and the physical structures of the database. Tablespaces are used to control the size of the database, the location of data in it, and other physical characteristics. On the other hand, tablespaces also organize purely logical structures — segments. The operating system is only aware of datafiles, where the tablespace data is stored. For this reason tablespaces are considered part of the logical, not physical, structure of the Oracle database.

A database must have at least one tablespace — SYSTEM. As you already know, it gets created during the creation of the database itself. You can, if you like, keep running the database with just the SYSTEM tablespace and create all other segments on it. However, for a database to run efficiently, and to even out IO, and to make it easier to manage your database, you should create other tablespaces.

Perhaps the first tablespace that needs to be created is a temporary tablespace. Temporary tablespaces are used when Oracle needs disk space for a sorting operation that exceeds the amount of memory allocated to a session by the `SORT_AREA_SIZE` `INIT.ORA` parameter. All such sorts need to be performed in a dedicated tablespace in order to avoid fragmentation. Temporary tablespaces will be discussed later in this chapter.

The next tablespace to be created is a dedicated rollback segment tablespace. You will use it to create all of your rollback segments in — and nothing else. Rollback segments behave very differently than other segment types and impose quite a few restrictions on managing the tablespace they are in, so for higher performance, efficient space use, and your sanity (Oracle calls this “ease of administration”), you should create a dedicated tablespace for all of your rollback segments.



Rollback segments will be discussed in detail in Chapter 10.

Other tablespaces that you may create vary, but usually include a Data tablespace for tables, an Index tablespace for indexes, and a Tools tablespace for application objects. You may also create a User tablespace for user objects, if your users are allowed to create them.

**Tip**

When creating a database using the Database Configuration Assistant, the Optimal Flexible Architecture will create the SYSTEM, TEMP (for sorting), USER (for data), INDX (for indexes), TOOLS (for application support), and RBS (for rollback segments) tablespaces. This is to get you in the habit of creating multiple tablespaces and having them associated with the type of data they will be used to store. The Database Configuration Assistant requires that all of these tablespaces be created and will not let you optionally remove one from the list to be created. If you do not want to create all six tablespaces, modify the database creation scripts that the Database Configuration Assistant creates.

Some things to keep in mind regarding tablespaces include

- ♦ A tablespace can belong to one and only one database.
- ♦ A tablespace can consist of one or more datafiles.
- ♦ A tablespace can contain one or more segments.
- ♦ The size of a tablespace is the sum of the sizes of datafiles it consists of.
- ♦ You can resize the datafiles or allow them to grow automatically.
- ♦ Most tablespaces (except SYSTEM) can be taken offline and brought back online.
- ♦ Most tablespaces can be made read-only or read-write.

Segment

Within a tablespace, space can be allocated to segments. A segment is an object in the database, but not every object is a segment. Segments have space allocated to them directly and can store data. For example, a table stores rows, an index stores pointers to rows in a table, a rollback segment stores undo information—they are all segments. A view, on the other hand, is not a segment since it does not store data—it is just a prewritten query that enables easy access to data stored in tables.

**Exam Tip**

Oracle8i has created a new type of view called a “materialized view.” Materialized views store data, such as summary information in a data warehousing environment, and can be used with another new Oracle object called a “dimension” to speed up processing of queries. You should know that materialized views do exist, but they are not covered on the exam.

There are different kinds of segments in an Oracle database. The easiest way to get a list of segment types present in your database is by using the following query:

```
SVRMGR> SELECT DISTINCT segment_type
2> FROM DBA_SEGMENTS;
SEGMENT_TYPE
-----
```

```
CACHE  
CLUSTER  
INDEX  
LOBINDEX  
LOBSEGMENT  
ROLLBACK  
TABLE  
7 rows selected.
```

This query only shows the segment types that exist in your database, not all possible types. Most will be discussed in other chapters.

General information you should know about segments includes

- ♦ A segment is an object that holds data.
- ♦ A segment can exist in only one tablespace, but its extents may be spread across multiple files belonging to that tablespace.
- ♦ Segments can be of different types, each having a different structure, behavior, and purpose.

Extent

In order to store data, segments are allocated space in a tablespace. The space is allocated in contiguous ranges known as extents. An extent is one or more contiguous database blocks within a tablespace and datafile. Each extent is recorded in the data dictionary, together with the segment it belongs to.



Tip

Oracle8i has introduced the capability to manage extents locally within a tablespace instead of going to the data dictionary to determine where in a datafile an extent should start. This is covered later in the “Creating Tablespaces” section of this chapter.

When a segment is created, it is allocated at least one extent. The DBA can control the size and location of the initial and future extents using various methods. Some information on extent sizing and placement will be covered in this chapter; you will see more throughout this book.

You should keep the following in mind regarding extents:

- ♦ An extent is a unit of space allocation in a tablespace and datafile.
- ♦ All segments are allocated at least one extent when they are created.
- ♦ Each extent can only be located in one datafile; however, multiple extents belonging to the same segment can be located in different datafiles that belong to the same tablespace.
- ♦ Some segments, such as partitioned tables, may have different extents in different tablespaces.

- ◆ A DBA can manually allocate an extent to a segment or allow Oracle to allocate them as needed.
- ◆ A DBA can manually specify the size of extents for each segment or allow Oracle to adjust it as needed.
- ◆ An extent has to be contiguous. Even if you have enough free space in the tablespace to create an extent, you may be unable to do so if the free space is not contiguous.
- ◆ An extent cannot be shared by multiple segments. It always belongs to one and only one segment.

Database block

The Oracle block is the minimum unit of IO in the database. When the database needs to read data, it cannot read just one row—it must read the entire block. The same happens with writing: Even if only one row in a block is changed, the DBWn process has to write the entire block during the checkpoint.

A DBA can control the size of the database block only during the creation of the database. Once set, the size cannot be changed. A DBA can, however, control how the space inside the block is used.



Block space utilization is covered in chapters 9 and 11.

- ◆ The database block is the minimum unit of IO.
- ◆ The database block consists of operating system blocks.
- ◆ The database block size is set by the `DB_BLOCK_SIZE` parameter during creation of the database and cannot be changed. In all current versions of Oracle, all blocks in the database are of the same size.
- ◆ A database block can belong to only one extent, or it can be part of free space.

Tablespaces

Tablespaces are provided by Oracle in order to enable logical divisions within the database. In addition, since tablespaces are based on specific datafiles, they also have a physical aspect. By using tablespaces, you can separate segments of different types, or you can locate segments that get accessed simultaneously (for example, a table and its indexes) on different disks. You can make part of your data read-only, or prevent access to it altogether by taking the tablespace offline.

Tablespace types

Tablespaces can be classified using many different criteria. The first is simple: the SYSTEM tablespace and everything else. Another way to classify tablespaces is by the type of objects they contain: permanent or temporary. Yet another way is by the space allocation mechanism used by the tablespace: dictionary managed or locally managed (a new feature in Oracle8i). All of these types will be discussed later in this chapter; first, let's get the SYSTEM tablespace "out of the way."

SYSTEM tablespace

The SYSTEM tablespace contains the data dictionary—internal tables that describe the structure of the database itself, all of the objects in it, users, roles, and privileges. When a user issues a query or a DML statement, the data dictionary is used in order to verify the user's permissions and find the data that belongs to the segment being queried or changed. When a DDL statement is processed, the internal tables are modified in order to reflect the change—an object created, altered, or dropped.

As a matter of fact, there really is no such thing as DDL as far as data is concerned. For example, when you issue a CREATE TABLE statement, the TAB\$ internal table has a row inserted into it with the definition of the table, the COL\$ table gets a row for each column in the new table, the UET\$ table gets a row describing the location and size of the initial extent allocated to the new table, and the FET\$ table is updated to reflect the fact that some of the space that used to be free is now not available. As you can see, a CREATE statement (DDL) is translated into a series of DML statements affecting the internal tables.

The SYSTEM tablespace also contains one rollback segment, SYSTEM, which can only be used for operations on objects stored in the SYSTEM tablespace. The internal tables are created by the SQL.BSQ script during creation of the database; the views and stored procedures are created immediately after that by running the CATALOG.SQL and CATPROC.SQL scripts. Other objects should be kept out of the SYSTEM tablespace in order to keep it operating efficiently.

Non-SYSTEM tablespaces

After the database has been created, you need to create additional tablespaces. There are many reasons to do this, but they all boil down to management and performance.

For performance reasons, it is best to separate data that will participate in resource contention. For example, a table and its indexes are normally accessed simultaneously during inserts and updates. In order to prevent a bottleneck, it is best to place them on different hard drives. In order to achieve that, you create different tablespaces—DATA and INDEX—and locate their datafiles on different hard drives. This way, you can update both the table and its index simultaneously and not risk resource contention.

It is also better to separate objects that have different storage requirements. For example, if you allow small tables and large tables to coexist in one tablespace, you simply cannot win: Allocating big extents to a small table is a waste of space, and building a large table out of small extents is very inefficient (the fewer extents, the better). Mixing large and small extents in the same tablespace is generally not recommended.



For more information on sizing extents for maximum performance, see Chapter 9.

Objects of different types are generally stored in separate tablespaces, as well. For example, you usually create a dedicated tablespace for rollback segments, another for temporary segments, and another one for indexes. This is mostly due to their behavior. If objects that acquire and deallocate space differently are placed in the same tablespace, the space becomes very fragmented, affecting performance of the database.

Another performance-related feature that requires multiple tablespaces is partitioning. Using partitioning, you can spread a table across multiple tablespaces — and therefore, multiple disks. For example, this is useful when a table contains historical data spanning many years. Queries of the past years will not affect UPDATE and INSERT activity on the current range of dates. As a matter of fact, you can even make some of the partitions read-only to prevent changes, or even take the tablespaces containing them offline if necessary — all without affecting other partitions.

In addition to the performance considerations, you generally take administrative considerations into account. For example, an administrative reason for putting rollback segments into a dedicated tablespace is that you cannot take a tablespace containing active rollback segments offline. In order to be able to take nonessential tablespaces offline, you need to make sure they do not contain any rollback segments. Also, if you want to make a tablespace read-only, you have to be certain it does not contain any objects that need to be modified.

As a rule, a database should contain at least the following non-SYSTEM tablespaces:

- ♦ **TEMPORARY** — for sorting activity
- ♦ **ROLLBACK** — for rollback segments
- ♦ **INDEX** — for indexes
- ♦ **DATA** — for tables
- ♦ **USER** — for user objects, if any

Many databases will also have other non-SYSTEM tablespaces, or more than one of the types mentioned above.

Tablespace contents

A tablespace exists in order to store segments. The types of segments a tablespace can store are determined at creation and can sometimes be changed. In essence, there are two types of tablespaces if you classify them by their contents: permanent and temporary.

Permanent objects

Permanent objects are the ones you are most familiar with. They include tables, indexes, rollback segments, and other segments that retain their data, even when the database is closed and the instance is shut down. Most of this book will discuss permanent objects.

Temporary objects

Objective

Distinguish the different types of temporary segments

Temporary objects in Oracle are segments that are created as necessary and only maintained until the database is closed and the instance is shut down. There are two kinds of temporary segments: those used for sorts and those for temporary tables (a new feature in Oracle8i).

Cross-Reference

Temporary tables will be discussed in Chapter 11.

The segments that fall within this exam objective are the ones used for sorting operations when there is not enough memory allocated in the PGA to complete the sort. In those cases, Oracle needs to allocate some space on disk in order to store the necessary data.

The temporary segment that Oracle uses is created in the user's temporary tablespace. By default, this tablespace is SYSTEM. You can find out which tablespace is a user's temporary tablespace by querying the DBA_USERS view like this:

```
SQL> SELECT username, temporary_tablespace
       2 FROM DBA_USERS;
```

USERNAME	TEMPORARY_TABLESPACE
SYS	TEMP
SYSTEM	TEMP
STUDENT	TEMP
OUTLN	SYSTEM
DBSNMP	SYSTEM

As you can see, there are users who still have SYSTEM as their temporary tablespace. In order to change that, you can use the ALTER USER command:

```
SQL> ALTER USER outln TEMPORARY TABLESPACE TEMP;  
User altered.  
SQL> ALTER USER dbsnmp TEMPORARY TABLESPACE TEMP;  
User altered.
```



The ALTER USER command is covered in detail in Chapter 17.

So, why is it not a good idea to make SYSTEM a user's temporary tablespace? The reason is that if a temporary segment is created in a permanent tablespace, it gets deallocated as soon as it is no longer necessary, releasing space. Every user who needs temporary space gets his or her own segment, which disappears as soon as the sort is over. As you will learn in Chapter 9, this leads to fragmentation of free space and very inefficient operation of that tablespace. If it is SYSTEM that is fragmented and inefficient, the entire database suffers because the SYSTEM tablespace stores the data dictionary and fragmentation on the SYSTEM tablespace may affect queries and modifications to the data dictionary.

You can create a tablespace specifically for temporary objects. These tablespaces cannot contain any permanent segments. Sort segments in these tablespaces behave very differently: Instead of being deallocated immediately after the sort is over, the segment is retained in the tablespace. If another session needs space, it will be allocated an extent out of the existing segment. In this way, there is only one sort segment in the temporary tablespace. If you are running Oracle Parallel Server, there is one sort segment per temporary tablespace per instance. Also, space allocation in these segments is not recorded in the data dictionary — instead, it is stored in the sort extent pool in the SGA. Because the SGA is destroyed when the instance is shut down, the space is released. When the instance starts and the database is opened again, a new segment is created as soon as it becomes necessary.

When creating temporary tablespaces, you should keep in mind how space in them is allocated. Because they are most often used for sorting, a session requests space in a temporary tablespace when its PGA can no longer accept any sort-related data. When space in a sort segment is allocated, the session writes out the data in the amount set by the SORT_AREA_SIZE initialization parameter. Because of this, you need to ensure that the extents in the temporary tablespace are sized according to the formula: (Multiple of SORT_AREA_SIZE) + 1 block for overhead. If the tablespace is dictionary managed, you can use the DEFAULT STORAGE clause; if the tablespace is locally managed, you need to use the UNIFORM SIZE clause.

The major differences between temporary segments created in permanent tablespaces and those created in temporary tablespaces are outlined in Table 8-1.

Table 8-1
**Comparison of Temporary Segments in Permanent
 and Temporary Tablespaces**

<i>Temporary Segments in Permanent Tablespaces</i>	<i>Temporary Segments in Temporary Tablespaces</i>
A segment is created every time a user session requires disk space in order to perform a sort.	One segment is created when the first user session needs disk space in order to perform a sort.
The segment is only used by one session.	The segment is used by all sessions requiring temporary space.
If multiple users are performing sort operations, multiple segments exist simultaneously.	Only one sort segment exists per temporary tablespace, per instance.
Extent allocations are recorded in the data dictionary.	Extent allocations in the temporary tablespace are recorded in the sort extent pool in the SGA.
After the sort is over, extents allocated to the temporary segments are released and the segment is dropped.	After the sort is over, the extents that were used by the session remain in the sort segment and can be reused by other sessions. The segment remains in the tablespace until the database is shut down.
Because of frequent allocation and deallocation of space, temporary segments in permanent tablespaces cause extreme fragmentation of the tablespace.	Temporary segments in temporary tablespaces cause no fragmentation at all.

If a tablespace was created as a permanent tablespace, it can be made temporary by using the following statement, as long as it does not contain any permanent objects:

```
SVRMGR> ALTER TABLESPACE temp TEMPORARY;  
Statement processed.
```

If a tablespace was created as a temporary tablespace, you may be able to make it permanent. It depends on what statement you used in order to create the tablespace. If you originally used the CREATE TEMPORARY TABLESPACE statement, the tablespace cannot be made permanent. If you used the CREATE TABLESPACE ... TEMPORARY statement to create the tablespace, you can make it permanent by using this command:

```
SVRMGR> ALTER TABLESPACE temp PERMANENT;  
Statement processed.
```

Creating tablespaces

Objective

Create tablespaces

There are two ways to create a tablespace in Oracle. You can use the CREATE TABLESPACE or CREATE TEMPORARY TABLESPACE command, or you can use the Storage Manager component of the Oracle Enterprise Manager.

The CREATE TABLESPACE command

The CREATE TABLESPACE and CREATE TEMPORARY TABLESPACE commands enable you to create a tablespace from the Server Manager or SQL*Plus interface quickly and easily. The main components of the command are the name of the tablespace to be created, the datafiles that will belong to it, and the default storage settings for the segments that are created in it.

The overall syntax of the command is somewhat complex because of the number of options available, as follows:

```
CREATE TABLESPACE tablespace name
  DATAFILE data file spec
  [MINIMUM EXTENT minimum extent size]
  [DEFAULT STORAGE (default storage clause)]
  [LOGGING|NOLOGGING]
  [ONLINE|OFFLINE]
  [PERMANENT|TEMPORARY]
  [EXTENT MANAGEMENT DICTIONARY|LOCAL]
```

Actually, all you need is the name and the datafile specification. Any settings not specified inherit the Oracle built-in default values. For example, create a tablespace by using the following statement:

```
SQL> CREATE TABLESPACE chapter_data
      2 DATAFILE 'C:\CERTDB\DISK4\chapter01.dbf' SIZE 10M;
Tablespace created.
```

This one command did everything: physically created the datafile, created the tablespace, updated the control file and the data dictionary, and set all the defaults for the new tablespace. Unfortunately, some of the defaults may not be the settings you want.

First, let us discuss the DATAFILE clause. Using this clause, you can configure the size of the datafile and its location, as you saw in the command above. You can also enable it to extend automatically if data in the tablespace grows beyond the current size of the datafiles. In order to configure automatic extension of the datafiles, you use three settings: AUTOEXTEND ON, NEXT, and MAXSIZE.

By specifying `AUTOEXTEND ON` in the `DATAFILE` clause, you are allowing the datafile to increase in size as necessary. However, you are not setting the maximum size (by default, `UNLIMITED`), nor are you specifying how much it should extend by. As a result, you can get a datafile that extends a little bit at a time — fragmenting the disk — and keeps extending until it runs out of disk space.

`AUTOEXTEND`, if you choose to allow it, should always be specified together with the `NEXT` and `MAXSIZE` parameters, like this:

```
SQL> CREATE TABLESPACE chapter_data
  2 DATAFILE 'C:\CERTDB\DISK4\chapter01.dbf' SIZE 10M
  3 AUTOEXTEND ON NEXT 10M MAXSIZE 50M;
Tablespace created.
```

This creates the tablespace with one datafile that has the initial size of 10MB and increases, if necessary, by 10MB until it reaches 50MB.

You can also create a tablespace with more than one datafile:

```
SQL> CREATE TABLESPACE chapter_data
  2 DATAFILE 'C:\CERTDB\DISK4\chapter01.dbf' SIZE 10M
  3 AUTOEXTEND ON NEXT 10M MAXSIZE 50M,
  4 'C:\CERTDB\DISK5\chapter02.dbf' SIZE 10M
  5 AUTOEXTEND ON NEXT 10M MAXSIZE 50M;
Tablespace created.
```

As you can see, this statement includes two complete datafile specifications, each with its own `AUTOEXTEND` clause. Also, the datafiles are on two different disks, which enables Oracle to spread the load when possible.

If the datafile already exists (for example, it used to belong to a tablespace that has been dropped), you can use that datafile by specifying the keyword `REUSE` instead of `SIZE`:

```
SQL> CREATE TABLESPACE chapter_data
  2 DATAFILE 'C:\CERTDB\DISK4\chapter01.dbf' REUSE;
Tablespace created.
```



In the good old days, when disk space was scarce and expensive, DBAs used to have a trick for reserving disk space for the database: They would create a tablespace with multiple datafiles and immediately drop it. When a tablespace is dropped, the datafiles are not deleted. When the DBA actually needed the space, he or she would add the datafiles to a real tablespace by using the `REUSE` keyword.

The next clause to discuss is the `DEFAULT STORAGE` clause. This part of the `CREATE TABLESPACE` statement enables us to provide defaults for all segments created in the tablespace. The actual value for each setting can be overridden when the

segment is created, with a few exceptions. If you do not specify all of the options for `DEFAULT STORAGE`, any setting not specified when the segment is created will inherit Oracle's built-in default settings.

The `DEFAULT STORAGE` clause consists of the key settings listed in Table 8-2 (and a few other settings).

Table 8-2
DEFAULT STORAGE Clause Components

<i>Setting</i>	<i>Description</i>
<code>MINEXTENTS</code>	The number of extents each segment will be created with. The default and minimum value is 1.
<code>MAXEXTENTS</code>	The maximum number of extents a segment will be allocated if necessary. The default for <code>MAXEXTENTS</code> depends on the <code>DB_BLOCK_SIZE</code> setting and the maximum value is <code>UNLIMITED</code> .
<code>INITIAL</code>	Set in bytes, or you can use <code>K</code> or <code>M</code> to specify <code>KB</code> or <code>MB</code> , respectively. Enables you to specify the size of the first extent allocated to a segment. Default: <code>5*DB_BLOCK_SIZE</code> . Minimum: <code>2*DB_BLOCK_SIZE</code> .
<code>NEXT</code>	Also set in bytes, or <code>K</code> (<code>KB</code>) or <code>M</code> (<code>MB</code>). Specifies the size of the second extent. The default is the same value as <code>INITIAL</code> .
<code>PCTINCREASE</code>	Set in percent. Specifies how much bigger (rounded up to the next multiple of <code>DB_BLOCK_SIZE</code>) to make the third extent compared to the second extent, the fourth compared to the third, and so forth.

For example, if `PCTINCREASE` is set to 20, the `INITIAL` and `NEXT` are set to 100K, the third extent will be 120K, the fourth 144K, and so on. Also, if the `PCTINCREASE` is set to 0, `SMON` won't automatically coalesce the free space in the tablespace.

The default for `PCTINCREASE` is 50, but should always be changed to 0 (or 1 if you want automatic coalescing of free space by `SMON`).

Avoid setting `MAXEXTENTS` to `UNLIMITED`. This enables a segment to take all the available space in the tablespace and disrupt normal operation of the database.

Coalescing of free space is covered in Chapter 9.

Keep in mind that all of the components of the `DEFAULT STORAGE` clause are just that—defaults. They can be overridden when you create the segments themselves, as you will find out in later chapters.



The one parameter that has to do with space allocation to segments in the new tablespace and cannot be overridden is `MINIMUM EXTENT`. This setting is not a default, but a limit. If you set `MINIMUM EXTENT`, Oracle will round up the size of each extent required for a segment to the nearest multiple of your setting. For example, if `MINIMUM EXTENT` is set to 500KB, extents can be created with 500KB, 1,000KB, 1,500KB, and so forth allocated to them. You use this parameter to make sure that all extents in the tablespace are as uniform as possible.

The next option is `LOGGING` or `NOLOGGING`. This simply specifies the default setting for direct loads into tables for this tablespace. By default, all data being loaded is also logged in the redo logs. If the load is run as `NOLOGGING` (this can be specified on the tablespace or table level as a default), the data is not recorded in the redo. This setting does not affect conventional loads.

**Cross-Reference**

Conventional and direct loads are covered in Chapter 14.

By default, a tablespace is available immediately after it is created. However, you can create it `OFFLINE` if you do not want it to be available. This option is rarely used.

Oracle8i introduced a new feature: locally managed tablespaces. You can create one by specifying `EXTENT MANAGEMENT LOCAL` as part of the `CREATE TABLESPACE` statements. This feature allows the tablespace to manage its own space by using a bitmap instead of involving internal tables in the data dictionary. The result is more efficient utilization of space with less overhead. You can create a permanent or temporary tablespace with local extent management. For these tablespaces, you cannot specify `DEFAULT STORAGE` or `MINIMUM EXTENT`. Also, you cannot use the `TEMPORARY` option—you need to use the `CREATE TEMPORARY TABLESPACE` statement instead.

**Objective****Allocate space for temporary segments**

Finally, the `TEMPORARY` option itself. In older versions of Oracle, this was the only way to create a temporary tablespace. Temporary tablespaces, as you remember, are used for sorting activity. With Oracle8i, there is a new `CREATE TEMPORARY TABLESPACE` command that should be used instead, because it enables you to create a locally managed temporary tablespace. In addition, it enables you to use so-called “tempfiles.” These are datafiles that can only be used for temporary tablespaces. Tempfiles can be created and used even when the database is in read-only mode. As a matter of fact, you cannot use the `DATAFILE` clause with the `CREATE TEMPORARY TABLESPACE` command—you need to use the `TEMPFILE CLAUSE` instead, like this:

```
SVRMGR> CREATE TEMPORARY TABLESPACE TEMP
        2> TEMPFILE 'C:\CERTDB\YURY.DBF' SIZE 10M
        3> EXTENT MANAGEMENT LOCAL;
Statement processed.
```

Using Oracle Storage Manager from OEM

You can also create (and manage) tablespaces with the Storage Manager component of the OEM. In order to create a tablespace, follow these steps:

Creating a Tablespace Using Storage Manager

1. Start the Oracle Enterprise Manager console and connect to the database you are going to create the new tablespace in.
2. Expand the Storage node, then the Tablespaces node. You should see a screen that looks like Figure 8-2.

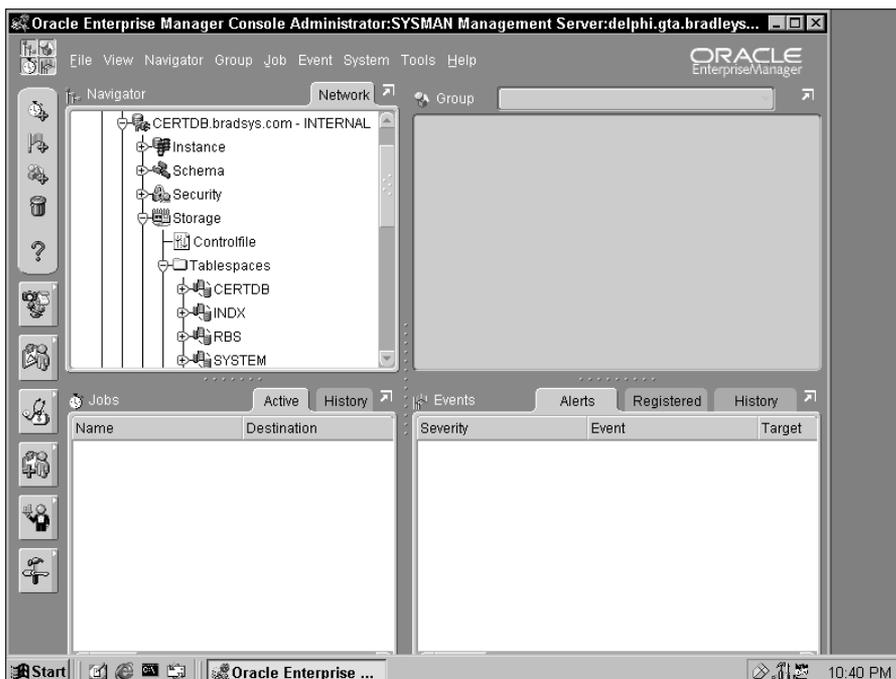


Figure 8-2: The Tablespaces node of Storage Manager

3. Right-click the Tablespaces node and select Create. In the screen that opens, type in the name for the new tablespace. You can also change the name and size of the datafiles that are created and assigned to the tablespace. The Show SQL button on the bottom enables you to see the statement that Storage Manager sends to the server in order to create the tablespace. The screen should now look like the one in Figure 8-3.

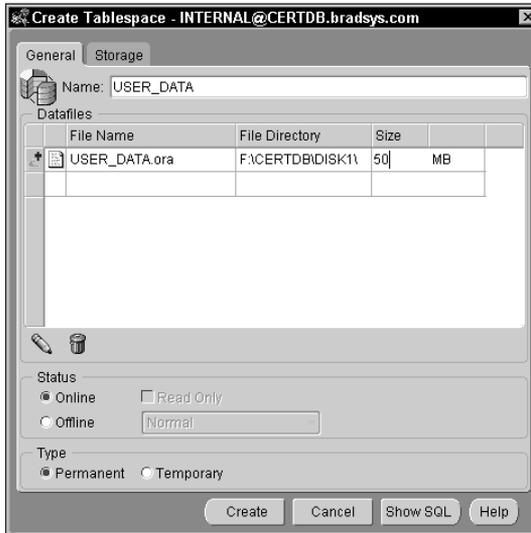


Figure 8-3: The Create Tablespace screen of Storage Manager

4. Double-click the datafile name. You will see a screen that enables you to set the file properties, such as its size, name, and automatic extension. Click OK or Cancel after making the changes.
5. Click the Storage tab. On this tab, you can configure the extent management settings (Local or Dictionary) and the storage settings. Depending on whether you choose “Locally managed” or “Managed in the dictionary,” you have different options for the storage settings. For a change, I clicked on the Show SQL button to show you the resulting statement in Figure 8-4.
6. Click the Create button, then click OK on the dialog box telling you that the tablespace was created successfully. The new tablespace should appear in the list.

You can modify the tablespace you have just created by right-clicking it and selecting Edit. If you already have a tablespace with specific settings and you want to create another one similar to it, you can right-click the existing tablespace and click Create Like on the pop-up menu. You can also modify the properties and settings of the new tablespace’s datafiles by expanding the tablespace, expanding the Datafiles node, right-clicking the datafile you need to modify, and clicking Edit.

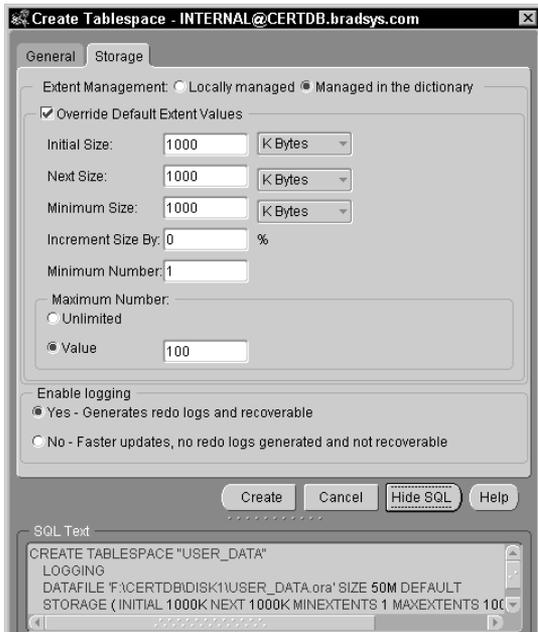


Figure 8-4: Configuring storage settings for a new tablespace

Maintaining tablespaces

Once a tablespace has been created, you can modify most of its properties. You can change the storage settings, make the tablespace temporarily unavailable or read-only, or allocate additional space to the tablespace. You can also relocate datafiles belonging to the tablespace, be it in order to improve performance or to utilize additional disk space available on another drive.

Taking tablespaces offline

Objective

Change the status of tablespaces

You can make most tablespaces unavailable to users by taking them offline. An offline tablespace is not accessed by Oracle and is available for maintenance. You can perform recovery operations on an offline tablespace while allowing users to continue accessing the rest of the database.

There are tablespaces that cannot be taken offline. The first is, of course, SYSTEM because the data dictionary must be accessible at all times. The other type of tablespace that cannot be taken offline is any tablespace that contains active (online) rollback segments. This is one of the reasons to keep rollback segments in dedicated tablespaces, so that you can take other tablespaces offline if necessary.

In order to take a tablespace offline, you use the ALTER TABLESPACE command. Here is how it looks:

```
SVRMGR> ALTER TABLESPACE chapter_data OFFLINE;  
Statement processed.
```

There are very few options for this command. They have to do with whether Oracle forces a checkpoint before taking the tablespace offline. In most cases, you want the checkpoint performed so that all data belonging to the tablespace is written to its datafiles before they become unavailable. This is known as OFFLINE NORMAL and is the default behavior. However, sometimes you need to take the tablespace offline when some of its datafiles are missing—for example, in case of a hard drive failure. In those cases, you can use the TEMPORARY or IMMEDIATE option. TEMPORARY performs a checkpoint on the datafiles that are available, but if some are missing they will require recovery. IMMEDIATE does not attempt a checkpoint, and the tablespace requires recovery before it is brought back online.



You should be aware that recovery is one of the reasons to take a tablespace offline. However, actual recovery scenarios are not covered on the Oracle8i DBA: Architecture and Administration exam.

If you take a tablespace offline, it remains offline until you bring it back online. Even restarting the database will not bring an offline tablespace back online automatically. If a user tries to access an offline tablespace, they get an error:

```
SVRMGR> SELECT num, title FROM chapters;  
NUM          TITLE  
-----  
ORA-00376: file 7 cannot be read at this time  
ORA-01110: data file 7: 'C:\CERTDB\DISK5\CHAPTER02.DBF'
```

It is interesting to note that the column headings do appear. This is because the table and column definitions are stored in the data dictionary, which remains online. Only the actual rows are stored in the CHAPTER_DATA tablespace.

In order to make the tablespace available again, you use the ALTER TABLESPACE command to bring it back online:

```
SVRMGR> ALTER TABLESPACE chapter_data ONLINE;  
Statement processed.
```

This is a very useful command for performing tablespace maintenance (as you will see later in this chapter) and recovery. If recovery is needed on the datafiles, you will not be able to bring the tablespace online, and must recover all datafiles before the tablespace can once again be available to users.

Using the Storage Manager, you take a tablespace offline or bring it back online by following these steps:

STEP BY STEP: Taking a Tablespace Offline or Bringing It Online with Storage Manager

1. Start Oracle Enterprise Manager and connect to the instance.
2. Expand the database, then the Storage node and the Tablespaces node.
3. Right-click the tablespace and select Place Online or Take Offline, as necessary. If you are taking the tablespace offline, you will need to select the mode—you usually select Normal.

Making tablespaces read-only

You can also keep a tablespace available for queries but prohibit changes. This is known as making the tablespace read-only. You may want to make a tablespace read-only if it contains historical data that should not be modified. Read-only tablespaces can also be moved to read-only media, such as a CD-ROM.

One of the advantages of read-only tablespaces is that they only need to be backed up once. This way, large amounts of static data can be excluded from the backup rotation. If the tablespace is not read-only, it needs to be backed up regularly even if there are no changes to data.

When you make a tablespace read-only, Oracle8i waits until all current transactions affecting the tablespace are complete, then performs a checkpoint and places the tablespace in read-only mode. For earlier releases (that is, prior to Oracle8i), Oracle recommends that the database be in RESTRICTED mode during the change, because no active transactions on the tablespace are allowed while it is being made read-only.

The only change that can be made to data in a read-only tablespace is dropping a segment. Because segment definitions are stored in the data dictionary and dropping a segment simply deletes its definition, a DROP statement on a table or index located in a read-only tablespace will succeed.

In order to make a tablespace read-only, you use the ALTER TABLESPACE command:

```
SVRMGR> ALTER TABLESPACE chapter_data READ ONLY;  
Statement processed.
```

The restrictions are

- ◆ All datafiles belonging to the tablespace must be online before the tablespace can be made read-only or read-write.
- ◆ The tablespace must not be involved in an online backup when it is made read-only.
- ◆ The tablespace cannot contain any active rollback segments.

In order to make the tablespace available for DDL and DML again, you use the ALTER TABLESPACE ... READ WRITE command:

```
SVRMGR> ALTER TABLESPACE chapter_data READ WRITE;  
Statement processed.
```

In order to make a tablespace read-only or writable with Storage Manager, follow these steps:

STEP BY STEP: Making a Tablespace Read-Only or Read-Write with Storage Manager

1. Start Oracle Enterprise Manager and connect to the instance.
2. Expand the database, then the Storage node and the Tablespaces node.
3. Right-click the tablespace and select Make Read-Only or Make Writable.

The Storage Manager tablespace menu is shown in Figure 8-5.

Changing the storage settings

Objective

Change the storage settings of tablespaces

You can change most of the settings defined during the creation of the tablespace by using the ALTER TABLESPACE command. In order to change the storage settings, use a command like this:

```
SVRMGR> ALTER TABLESPACE chapter_data  
2> DEFAULT STORAGE (  
3> INITIAL 200K NEXT 200K PCTINCREASE 10);  
Statement processed.
```

Or, you can change the MINIMUM EXTENT setting:

```
SVRMGR> ALTER TABLESPACE chapter_data  
2> MINIMUM EXTENT 200k;  
Statement processed.
```

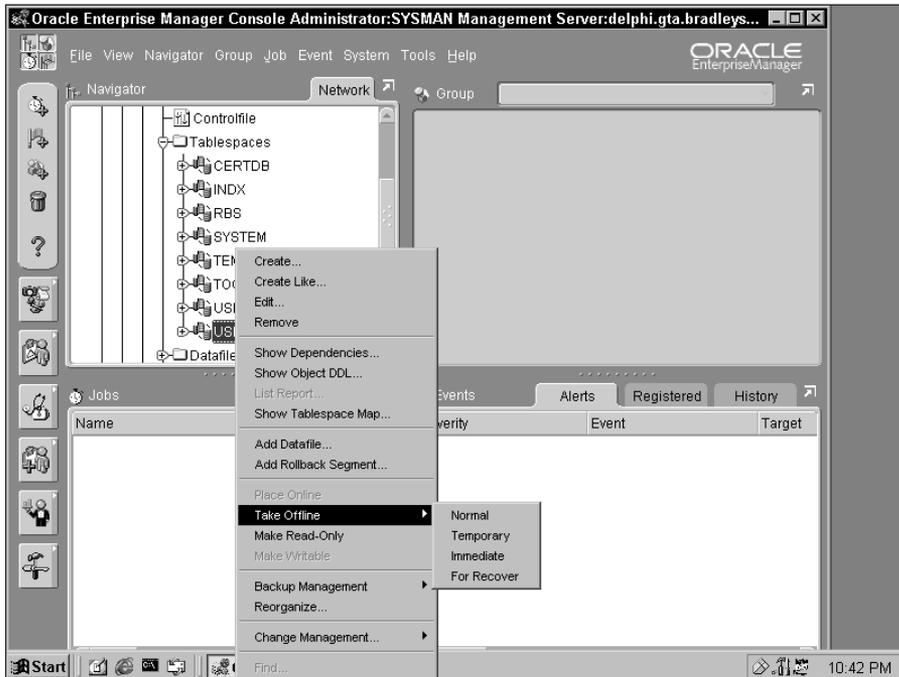


Figure 8-5: Changing the status of a tablespace in Storage Manager

When storage settings for the tablespace change, the space currently allocated to segments is not restructured. Each modification to a setting only affects future space allocations. For example, `MINIMUM EXTENT` will control the minimum extent size of future extents created in the tablespace; the new `INITIAL` setting will only apply to new segments created from now on.

In order to change the storage settings for a tablespace with `STORAGE MANAGER`, follow these steps:

STEP BY STEP: Changing Storage Settings of a Tablespace in Storage Manager

1. Start Oracle Enterprise Manager and connect to the instance.
2. Expand the database, then the Storage node and the Tablespaces node.
3. Right-click the tablespace and select Edit.
4. Click on the Storage tab. You get the screen shown in Figure 8-4 earlier.
5. Make the necessary changes and click OK.

Resizing tablespaces

Objective

Change the size of tablespaces

A tablespace is created with a specific amount of disk space allocated to it. The total size of a tablespace is the sum of the sizes of all datafiles that belong to it. Sometimes you need to increase the size of the tablespace in order to accommodate more data than you originally anticipated. This is done in one of the following three ways.

Automatic resizing

Perhaps the easiest way to give the tablespace more disk space is to allow its files to grow automatically. You do that by using the automatic extension feature of Oracle datafiles. This enables you to minimize administrative load while ensuring sufficient space for the data.

**Caution**

Allowing files to extend automatically is good in moderation. If automatic growth is not restricted by using the `NEXT` and `MAXSIZE` parameters, it can cause heavy disk fragmentation and may cause the system to run out of disk space. In some cases, this can cause the Oracle instance — or, even worse, the server — to crash.

You have already seen how to enable the `AUTOEXTEND` feature on a datafile when the tablespace is created. In order to enable an existing datafile to extend automatically, use the `ALTER DATABASE` command:

```
SVRMGR> ALTER DATABASE
        2> DATAFILE 'C:\CERTDB\DISK5\CHAPTER02.DBF'
        3> AUTOEXTEND ON NEXT 10M MAXSIZE 60M;
Statement processed.
```

You can also turn the automatic extension off with this command:

```
SVRMGR> ALTER DATABASE
        2> DATAFILE 'C:\CERTDB\DISK5\CHAPTER02.DBF'
        3> AUTOEXTEND OFF;
Statement processed.
```

Oracle is very flexible when it comes to automatically extending the datafiles. You do not need to enable `AUTOEXTEND` on all of the files that belong to a tablespace — it can be just one. Also, Oracle will choose which file to extend based on which file would need to extend the least in order to end up with the required amount of space. For example, I am trying to allocate a 200KB extent in the `CHAPTER_DATA` tablespace. The `CHAPTER01.DBF` file has 150KB available while the `CHAPTER02.DBF` has 50KB available. Both files are enabled for automatic extension. Oracle extends the `CHAPTER01.DBF` datafile, because it requires the least extension, and creates the new extent there.

By default, Oracle only extends the datafile by as little as necessary to accommodate the new extent. Because of that, the space on the hard drive can become very fragmented. In order to minimize this fragmentation, you need to use the `NEXT`

setting in the AUTOEXTEND clause. Also, you should use the MAXSIZE parameter in order to prevent the file from occupying the entire disk and preventing other files or applications from acquiring space on it.

Manual resizing

Many DBAs prefer to resize datafiles manually rather than rely on automatic extension. In order to do that, you need to use the ALTER DATABASE command:

```
SVRMGR> ALTER DATABASE
      2> DATAFILE 'C:\CERTDB\DISK5\CHAPTER02.DBF'
      3> RESIZE 60M;
Statement processed.
```

Normally, you use this command in order to increase the size of the datafile. However, you can also decrease its size if the space in the end of the file has never been used. One thing to remember is that you cannot use this method to resize a read-only tablespace without making it read-write first.

Adding datafiles to tablespaces

The third way to increase the size of a tablespace is to add datafiles to it. You normally perform this operation when you run out of space on a drive but need more space for the tablespace.

In order to add a datafile to an existing tablespace, you use the ALTER TABLESPACE command:

```
SVRMGR> ALTER TABLESPACE CHAPTER_DATA
      2> ADD DATAFILE 'C:\CERTDB\DISK6\CHAPTER03.DBF'
      3> SIZE 20M;
Statement processed.
```

This command creates the new datafile and adds it to the existing tablespace. You can also add an existing datafile by using the REUSE keyword.

In order to resize a tablespace with Storage Manager, you can use one of the following procedures.

STEP BY STEP: Resizing a Tablespace with Storage Manager (Option 1)

1. Start Oracle Enterprise Manager and connect to the instance.
2. Expand the database, then the Storage node and the Tablespaces node.
3. Expand the tablespace you need to modify, then the Datafiles node.
4. Right-click the datafile you want to change settings for and select Edit.

5. On the General tab of the Edit Datafile screen, change the size of the datafile. On the Storage tab (shown in Figure 8-6), enable or disable automatic extension and configure the NEXT and MAXSIZE parameters.

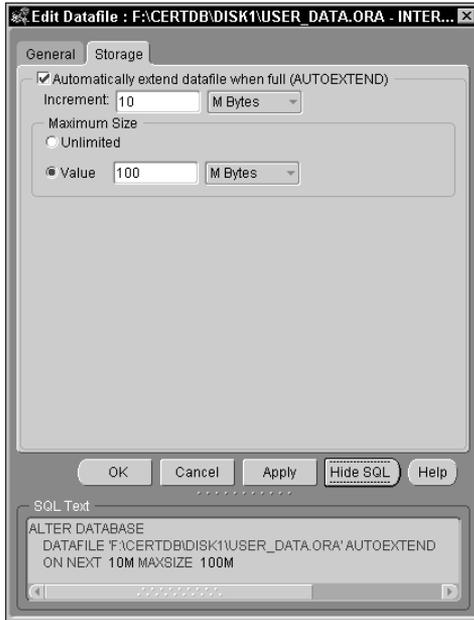


Figure 8-6: Configuring automatic extension settings for a datafile

6. Click OK to apply the changes and close the screen.

STEP BY STEP: Resizing a Tablespace with Storage Manager (Option 2)

1. Start Oracle Enterprise Manager and connect to the instance.
2. Expand the database, then the Storage node and the Datafiles node.
3. Follow steps 4–6 in Option 1.

Moving datafiles



Relocate tablespaces

Sometimes you need to relocate a datafile to a new disk. Maybe the old one failed, or maybe you need more space and do not want to add another datafile. Perhaps you want to move the datafile to a RAID array, or simply to a faster drive. Maybe you inherited the database from another DBA, and the structure is light-years from being OFA-compliant. There are countless reasons to relocate datafiles.

There are two ways to move datafiles from one location to another. You can shut the database instance down, restart it in a MOUNT state, and use the ALTER DATABASE RENAME FILE command, or you can take the affected tablespace offline and use the ALTER TABLESPACE ... RENAME DATAFILE command.

When the tablespace can be taken offline, you should avoid shutting down the entire database. Simply follow these steps:

STEP BY STEP: Relocating a Datafile Using the ALTER TABLESPACE ... RENAME DATAFILE Command

1. Place the tablespace offline:

```
SVRMGR> ALTER TABLESPACE CHAPTER_DATA OFFLINE;  
Statement processed.
```

2. Using the operating system commands or utilities, copy or move (copying is safer as the old file remains on disk) the datafile to the new location:

```
C:\>copy C:\CERTDB\DISK6\CHAPTER03.DBF C:\CERTDB\DISK3  
1 file(s) copied.
```

If you are using a UNIX system, use the cp command:

```
cp /CERTDB/DISK6/CHAPTER03.DBF /CERTDB/DISK3
```

You can also use the GUI file management application (like Windows Explorer) to copy or move the file.

3. Issue the ALTER TABLESPACE ... RENAME DATAFILE command:

```
SVRMGR> ALTER TABLESPACE CHAPTER_DATA  
2> RENAME DATAFILE 'C:\CERTDB\DISK6\CHAPTER03.DBF' TO  
3> 'C:\CERTDB\DISK3\CHAPTER03.DBF';  
Statement processed.
```

4. Place the tablespace online:

```
SVRMGR> ALTER TABLESPACE CHAPTER_DATA ONLINE;  
Statement processed.
```

5. If you copied the file to the new location, you can delete the old copy if the tablespace came back online normally. To confirm that everything worked, query the DBA_DATA_FILES view:

```
SVRMGR> SELECT file_name FROM DBA_DATA_FILES  
2> WHERE tablespace_name = 'CHAPTER_DATA';
```

```
FILE_NAME
-----
C:\CERTDB\DISK4\CHAPTER01.DBF
C:\CERTDB\DISK5\CHAPTER02.DBF
C:\CERTDB\DISK3\CHAPTER03.DBF
3 rows selected.
```

As you can see, the file is now located in the DISK3 directory. The old copy in DISK6 can be deleted.

If you get an error placing the tablespace online, check that the path is correct, that the permissions are set up correctly and Oracle can access the datafile, and that you did not forget to copy the file — the ALTER TABLESPACE command only changes the location of the datafile in the control file and in the data dictionary.

The other way to move a datafile is through the use of the ALTER DATABASE command. This method is used when the file to be moved belongs to a tablespace that cannot be taken offline — for example, the SYSTEM tablespace.

In order to relocate a non-SYSTEM datafile using the Storage Manager, follow these steps:

STEP BY STEP: Relocating a non-SYSTEM Datafile Using the Storage Manager

1. Start Oracle Enterprise Manager and connect to the instance.
2. Expand the database, then the Storage node and the Tablespaces node.
3. Right-click the tablespace and select Take Offline, then Normal.
4. Using the operating system command or utility, copy the datafile to the new location.
5. Right-click the datafile being relocated and select Edit.
6. On the General tab of the Edit Datafile screen, modify the file name and/or path. Click OK to save changes and close the screen.
7. Right-click the tablespace and select Place Online.
8. After checking that the file has been relocated successfully, you can delete the old copy.

In order to move a tablespace to a read-only media, you must make it read-only first. Follow these steps:

STEP BY STEP: Moving a Tablespace to Read-Only Media (CD-ROM)

1. Make the tablespace read-only by using the ALTER TABLESPACE command or Storage Manager.
2. Take the tablespace offline by using the ALTER TABLESPACE command or Storage Manager.
3. Using a third-party utility, copy the datafiles to the read-only media.
4. Using the ALTER TABLESPACE ... RENAME DATAFILE command or Storage Manager, change the path to the datafiles.
5. Using the ALTER TABLESPACE command or Storage Manager, bring the tablespace online.

In order to move a datafile that belongs to the SYSTEM tablespace, you need to mount the database and use the ALTER DATABASE RENAME FILE command. Follow these steps:

STEP BY STEP: Relocating a Datafile Using the ALTER DATABASE RENAME FILE Command

1. Shut down the database:

```
SVRMGR> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

2. Using an operating system command or utility, move or copy the datafile to the new location:

```
C:\>copy C:\CERTDB\DISK1\SYSTEM01.DBF C:\CERTDB\DISK2
1 file(s) copied.
```

If using a UNIX system, use the cp command:

```
cp /CERTDB/DISK1/SYSTEM01.DBF /CERTDB/DISK2
```

3. Mount the database:

```
SVRMGR> STARTUP MOUNT PFILE=C:\CERTDB\INITCERT.ORA
ORACLE instance started.
Total System Global Area                4818188 bytes
Fixed Size                               70924 bytes
Variable Size                            3850240 bytes
Database Buffers                         819200 bytes
```

```
Redo Buffers                          77824 bytes
Database mounted.
```

4. Use the `ALTER DATABASE` command in order to update the control file with the new location of the datafile:

```
SVRMGR> ALTER DATABASE RENAME FILE
        2> 'C:\CERTDB\DISK1\SYSTEM01.DBF' TO
        3> 'C:\CERTDB\DISK2\SYSTEM01.DBF';
Statement processed.
```

5. Open the database:

```
SVRMGR> ALTER DATABASE OPEN;
Statement processed.
```

6. Use the `DBA_DATA_FILES` view to check that the file has been moved:

```
SVRMGR> SELECT file_name FROM DBA_DATA_FILES
        2> WHERE tablespace_name = 'SYSTEM';
FILE_NAME
-----
C:\CERTDB\DISK2\SYSTEM01.DBF
1 row selected.
```

Now you can delete the old copy of the datafile.

Another time you would use the `ALTER DATABASE RENAME FILE` command is when you are relocating the rollback tablespace. Although it can be taken offline, it is faster and much easier to shut down the database.



You need to know very well when to use the `ALTER TABLESPACE` and when to use the `ALTER DATABASE` command in order to manage the datafiles.

In order to perform the same operation using Oracle Enterprise Manager, use this procedure:

STEP BY STEP: Relocating a SYSTEM Datafile Using the Oracle Enterprise Manager

1. Start Oracle Enterprise Manager and connect to the instance.
2. Right-click the database and select Shutdown.
3. On the Shutdown Options screen, select the shutdown mode—you usually choose Immediate or Transactional. Click OK.
4. When the shutdown is complete, click Close to close the message box. Right-click the database again and choose Startup.

5. On the Startup Options screen, shown in Figure 8-7, select Mount and choose the stored configuration or the parameter file to be used. Click OK to mount the database.

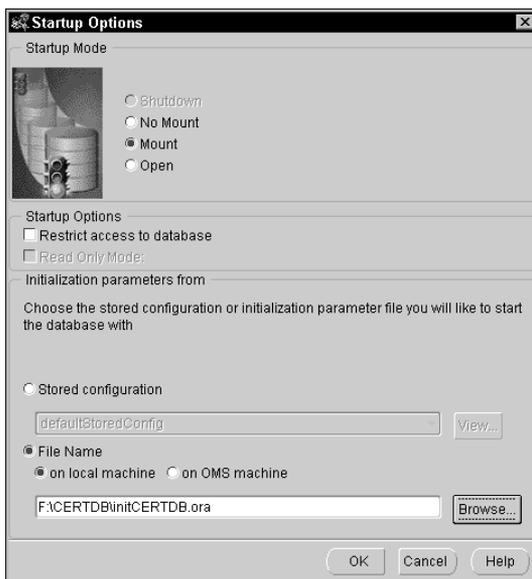


Figure 8-7: Configuring startup options

6. When the database is mounted, click Close to dismiss the message box.
7. Using operating system commands or utilities, copy the datafile to the new location.
8. In Oracle Enterprise Manager, right-click the database and select Related Tools, then SQL*Plus Worksheet. You cannot edit the properties of a datafile in Storage Manager unless the database is open.
9. Issue the ALTER DATABASE RENAME FILE command, as shown in Figure 8-8, and click the Execute button on the left to run the command.
10. Close SQL*Plus Worksheet.
11. In Oracle Enterprise Manager, right-click the database and select Startup. Confirm that Open is selected, click OK, and then (once the database is open) click Close on the message box.
12. Check that the datafile has been relocated successfully by using the Storage Manager. If everything is OK, delete the old copy of the datafile.



If you choose to copy the files rather than move them, you can use the old copy as a backup of the datafile. In order for this copy to be useable, your database needs to be in archive log mode.

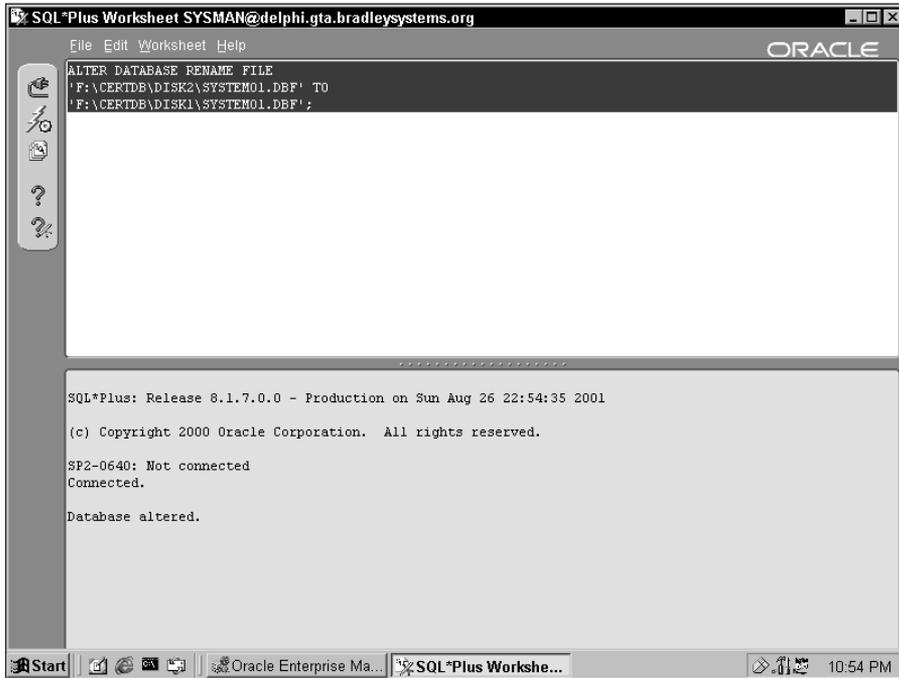


Figure 8-8: SQL*Plus Worksheet

Dropping tablespaces

This has to be the easiest part:

```
SVRMGR> DROP TABLESPACE CHAPTER_DATA;
Statement processed.
```

If there are objects in the tablespace, you get the following error:

```
SVRMGR> DROP TABLESPACE CHAPTER_DATA;
DROP TABLESPACE CHAPTER_DATA
*
ORA-01549: tablespace not empty, use INCLUDING CONTENTS option
```

What Oracle is trying to tell you is that you need to issue the following command:

```
SVRMGR> DROP TABLESPACE CHAPTER_DATA INCLUDING CONTENTS;
Statement processed.
```

If this generates an error, you may need to use the `CASCADE CONSTRAINTS` option. For example, you have a parent table in the `CHAPTER_DATA` tablespace, and there is a child table in another tablespace. The following command drops the `CHAPTER_DATA` tablespace, all of its contents, and all of the constraints referring to it:

```
SVRMGR> DROP TABLESPACE CHAPTER_DATA
          2>INCLUDING CONTENTS CASCADE CONSTRAINTS;
Statement processed.
```



Dropping a tablespace including contents and cascading constraints can have a domino effect on the database and may make it unusable. This is because, unless you have a good idea of what the tablespace contents are and why there are constraints depending on the objects in the tablespace you are dropping, you may inadvertently drop objects and constraints that your data finds critical. It is a good idea to drop all objects in a tablespace before dropping the tablespace itself. The only exception might be testing or development environments where the data is not critical.

The result of dropping a tablespace is that all mention of it is erased from the data dictionary and the control file of the database. The datafile is not actually deleted, but it may as well be—there is no metadata that allows us to access its contents. If you need to delete the datafile, use the operating system commands or utilities.

You can also drop a tablespace using the Storage Manager:

STEP BY STEP: Dropping a Tablespace with Storage Manager

1. Start Oracle Enterprise Manager and connect to the instance.
2. Expand the database, then the Storage node and the Tablespaces node.
3. Right-click the tablespace and select Remove. Click Yes to confirm that you want to remove the tablespace.



Storage Manager automatically uses the `INCLUDING CONTENTS` option when dropping tablespaces.

Getting information about tablespaces

There are quite a few views that deal with tablespaces and datafiles. The main ones are `DBA_TABLESPACES`, `DBA_DATA_FILES`, `V$DATAFILE` and `V$TABLESPACE`. Table 8-3 describes the most commonly used columns in `DBA_TABLESPACES`.

Table 8-3
DBA_TABLESPACES View

<i>Column Name</i>	<i>Description and Possible Values</i>
TABLESPACE_NAME	Self-explanatory: the name of the tablespace. The view has one row per tablespace.
INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE	The DEFAULT STORAGE settings specified for the tablespace. These can be modified by using the ALTER TABLESPACE command.
MIN_EXTLEN	The MINIMUM EXTENT setting for the tablespace. All extents created in the tablespace will be a multiple of this size.
STATUS	Whether the tablespace is ONLINE, OFFLINE or READ ONLY.
CONTENTS	PERMANENT or TEMPORARY.
LOGGING	LOGGING or NOLOGGING.
EXTENT_MANAGEMENT	DICTIONARY for dictionary-managed tablespaces, LOCAL for locally managed ones.

The DBA_DATA_FILES view is just as straightforward, as detailed in Table 8-4.

Table 8-4
DBA_DATA_FILES View

<i>Column Name</i>	<i>Description and Possible Values</i>
FILE_NAME	The operating system path and name of the datafile.
FILE_ID	The numeric file ID. Can be very useful – in many cases, you can refer to the files by their ID's, especially during database recovery.
TABLESPACE_NAME	The name of the tablespace the datafile belongs to.
BYTES, BLOCKS	The size of the datafile in bytes and database blocks.
USER_BYTES, USER_BLOCKS	The amount of useable space in the datafile, in bytes and database blocks.
STATUS	AVAILABLE or, in case of errors, INVALID.
AUTOEXTENSIBLE	Whether the automatic extension has been enabled for the datafile. The possible values are YES and NO.

<i>Column Name</i>	<i>Description and Possible Values</i>
MAXBYTES	If the file is not allowed to extend automatically, the value is 0. Otherwise, this is the value set by the MAXSIZE parameter when automatic extension was enabled. The file will grow to this size in bytes, if necessary.
MAXBLOCKS	Same as MAXBYTES, but in database blocks.
INCREMENT_BY	Only for files that can extend automatically: the number of blocks it will extend by. Equal to the value of the NEXT parameter set when automatic extension was enabled expressed in database blocks.

As you know, some of the operations on datafiles are performed when the database is in MOUNT, not OPEN. In those cases, you cannot use the DBA_ views because they are only available when the database is OPEN. However, V\$ views can still be used. The view you need the most is V\$DATAFILE. This view is described in Table 8-5.

Table 8-5
V\$DATAFILE View

<i>Column Name</i>	<i>Description and Possible Values</i>
NAME	The physical path and name of the datafile.
FILE#	The numeric file ID.
TS#	The numeric tablespace ID. If you need the tablespace name, you can join this view to V\$TABLESPACE using the TS# column in both views.
BYTES	The current size of the datafile in bytes.
BLOCKS	The current size of the datafile in database blocks.
CREATE_BYTES	The size of the datafile at creation. For files that have never been extended, it will be the same as current size.
STATUS	ONLINE, OFFLINE, or SYSTEM. Interesting to note that datafiles belonging to the SYSTEM tablespace have their own status.
BLOCK_SIZE	Currently, it is the same for all datafiles – the database block size. However, it is an indication that Oracle is actively working on allowing different block sizes to coexist in the same database. According to Oracle, this feature will be available in Oracle 9i, although not in the initial release.

You may also need to monitor the activity in the temporary tablespaces. The views you would need for that are V\$SORT_SEGMENT and V\$SORT_USAGE.

The `V$SORT_SEGMENT` shows you the details on the segments in the temporary tablespaces. For example, you can examine the current size of the temporary segments in each temporary tablespace by using the `V$SORT_SEGMENT`:

```
SVRMGR> SELECT tablespace_name, current_users,
              2> total_extents, used_extents
              3> FROM V$SORT_SEGMENT;
TABLESPACE_NAME          CURRENT_US TOTAL_EXTE USED_EXTEN
-----
TEMP                      2          14         14
1 row selected.
```

However, this view does not show temporary segments in permanent tablespaces. In order to see those, you need the `V$SORT_USAGE` view.

The `V$SORT_USAGE` view enables you to see a row per session that is doing a sort on disk. You can tell how big the sort is and where the segment is allocated, even if the segment is in a permanent tablespace. As soon as the sort is over, the row for that session disappears. Here is an example showing that one of the users is currently sorting in the `SYSTEM` tablespace:

```
SVRMGR> SELECT user, tablespace, contents, extents
              2> FROM V$SORT_USAGE;
USER          TABLESPACE          CONTENTS  EXTENTS
-----
SYS           TEMP                TEMPORARY    13
SYS           TEMP                TEMPORARY     1
STUDENT      SYSTEM              PERMANENT     1
3 rows selected.
```

You can use these views to detect potential problems, determine the need to add more space to the temporary tablespaces, or solve performance problems caused by excessive sorting.

Guidelines for tablespaces

In order to keep the database manageable and performing well, the database needs to have more than one tablespace. As a minimum, you should have the following:

- ♦ **SYSTEM** — Should be reserved strictly for the data dictionary structures. In order to achieve this, make sure that all users have been assigned a default and a temporary tablespace and that you do not create user objects while logged in as `SYS` or `INTERNAL`.
- ♦ **TEMP** — A temporary tablespace. This is the tablespace that holds temporary tables and sorts segments for users. In order to prevent unnecessary allocation and deallocation of space by these segments, every user should have a temporary tablespace assigned.

- ♦ **ROLLBACK**—A dedicated tablespace for rollback segments. This tablespace is necessary both for performance and manageability. In order to further even out disk IO, you may want to create multiple tablespaces to hold rollback segments, with datafiles on multiple disks.
- ♦ **DATA**—For storing tables. You may very well have multiple data tablespaces in order to separate segments of different sizes, or to partition large tables, or simply to be able to make only part of the data unavailable by taking a tablespace offline. A database will typically have several tablespaces to hold segment data.
- ♦ **INDEX**—A dedicated tablespace on a separate disk is necessary for good DML performance. This way, the DML activity on the table and corresponding activity on its indexes can happen without affecting each other. If you have an INDEX tablespace, it is also easier to take the tablespace containing parent tables offline. This is discussed in more detail in Chapter 13. A database will typically have several tablespaces to hold index segments.
- ♦ **USER**—If users are going to be allowed to create objects, you need this tablespace. You should make sure that it is the users' default tablespace, and you need to ensure that this is the only tablespace for which most users have any quota.



Quotas on tablespaces are discussed in Chapter 17.

You may need to create others, as well. For example, you may need a dedicated read-only tablespace for historical data. Or you may need a TOOLS tablespace for tables supporting an application. When you need to administer a subset of the database individually, you may want to consider a dedicated tablespace.

When you create a tablespace, make sure that you explicitly specify the `DEFAULT STORAGE` clause as well as the `MINIMUM EXTENT` clause for data dictionary-managed tablespaces. Locally managed tablespaces should have a `UNIFORM SIZE` parameter specified. The extents in a tablespace should be as uniform as possible in order to minimize fragmentation.

There is a big difference between a tablespace that contains static data and a read-only tablespace. If you can make a tablespace containing historical data read-only, you will only need to back it up once. Also, Oracle will prevent any accidental changes to the data.

Key Point Summary

In this chapter, we discussed the physical and the logical structure of the database. In order to continue on to the following chapters, you have to have a thorough understanding of the concepts summarized here.

- ♦ The database has a physical structure and a logical structure. From the physical point of view, the database consists of datafiles built out of operating system blocks. The logical view of the database is that it consists of tablespaces, which contain segments built out of extents made up of database blocks. The smallest logical unit in the database is the database block.
- ♦ A tablespace is the link between the physical and logical structures of a database. By using tablespaces, you can control the location of data and administer parts of the databases separately.
- ♦ A segment is an object that is allocated space in datafiles. There are different types of segments — tables, indexes, partitions, rollback segments, temporary or sort segments, and many others. A segment can only be located in one tablespace.
- ♦ An extent is the unit of space allocation in a tablespace. An extent is a group of contiguous blocks and must exist in one datafile. Different extents belonging to the same segment can be located in different datafiles belonging to the same tablespace. Multiple segments cannot share an extent in Oracle.
- ♦ A database block is the minimum unit of IO within the database. The size of all blocks in the database is set at its creation and cannot be changed. The database block size can be anywhere from 2,048 bytes to 65,536 bytes, depending on the operating system. Valid settings are 2,048, 4,096, 8,192, 16,384, 32,768, and 65,536 bytes.
- ♦ The tablespaces can be of different types. First, there is SYSTEM — the tablespace that contains the data dictionary. The rest of the tablespaces can be classified according to their contents — permanent or temporary — or according to the space management mechanism used — locally managed or dictionary-managed. All tablespaces besides SYSTEM, are called non-SYSTEM tablespaces.
- ♦ A tablespace can contain permanent objects or temporary objects. Temporary objects are segments created for sort operations and temporary tables. These objects cause very high fragmentation if allowed to exist in a permanent tablespace, so a temporary tablespace should always be assigned to users with the CREATE USER or ALTER USER command. You should always create at least one temporary tablespace.
- ♦ A tablespace can be created using the CREATE TABLESPACE statement, or you can use the Storage Manager component of the Oracle Enterprise Manager.
- ♦ When creating a tablespace, you need to specify the datafiles it will use, the default storage settings for the segments created in it, and restrictions such as MINIMUM EXTENT or UNIFORM SIZE.
- ♦ You can allow the datafiles to extend automatically if necessary by using the AUTOEXTEND clause in the datafile definition. If you do, make sure you specify NEXT (how much the file will grow by) and MAXSIZE (the maximum size it will grow to) to prevent problems.

- ♦ If you need to increase the size of an existing tablespace, you can use one of three methods. You can add a datafile, resize a datafile, or enable one or more datafiles to extend automatically.
- ♦ Storage settings for an existing tablespace can be changed dynamically by using the ALTER TABLESPACE command. The new settings won't affect currently allocated extents, only the future space allocations.
- ♦ A tablespace can be made unavailable by taking it offline using the ALTER TABLESPACE command. You may need to take a tablespace offline in order to perform maintenance or recovery on it.
- ♦ A tablespace that contains static or historical data should be made read-only by using the ALTER TABLESPACE command. Read-only tablespaces do not need to be backed up regularly, and most changes to data are prevented. You can, however, drop an object located in a read-only tablespace.
- ♦ If you need to relocate a datafile belonging to a tablespace, you can either take the tablespace offline and use the ALTER TABLESPACE ... RENAME DATAFILE command, or you can shut down and mount the database and use the ALTER DATABASE ... RENAME FILE command. The first method is the preferred method for tablespaces that can be taken offline, the second is necessary when you are relocating the SYSTEM tablespace or a tablespace containing rollback segments.
- ♦ If you no longer need a tablespace, it can be dropped. Dropping the tablespace does not delete the operating system files, but it does remove the tablespace from the control file and from the data dictionary. If the tablespace contains objects, you need to use the DROP TABLESPACE ... INCLUDING CONTENTS command.
- ♦ In order to view information on your tablespaces and datafiles, you can use the DBA_TABLESPACES and DBA_DATA_FILES views. When the database is not open, you can query the V\$DATAFILE and V\$TABLESPACE views.
- ♦ In order to monitor sorting activity in the database, you can use the V\$SORT_USAGE view. For details on temporary segments in temporary tablespaces, query the V\$SORT_SEGMENT view.



STUDY GUIDE

Use the multiple-choice questions and scenarios below to test your knowledge. The lab exercises are designed to practice creating, managing, and dropping tablespaces.

Assessment Questions

1. How can you relocate a datafile belonging to the SYSTEM tablespace?
 - A. By using the ALTER DATABASE RENAME DATAFILE command
 - B. By using the ALTER DATABASE RENAME FILE command
 - C. By using the ALTER TABLESPACE ... RENAME FILE command
 - D. By using the ALTER TABLESPACE ... RENAME DATAFILE command
2. Which of the following tablespaces cannot be taken offline?
 - A. A read-only tablespace
 - B. Any tablespace containing rollback segments
 - C. A temporary tablespace
 - D. Any tablespace containing active rollback segments
3. When must you make the tablespace read-only in order to move it to a CD-ROM?
 - A. Before taking it offline
 - B. After taking it offline but before copying the files
 - C. After copying the files but before bringing it online
 - D. After bringing the tablespace online
4. You dropped a tablespace and are now trying to re-create it using the original datafiles, but you are getting an error that the files cannot be created. What must you do for the creation to succeed?
 - A. Use the REUSE option instead of SIZE in the file specification.
 - B. Use the OVERWRITE option instead of SIZE in the file specification.
 - C. You cannot reuse the existing files. They must be deleted before re-creating the tablespace.
 - D. You did not drop the tablespace before trying to re-create it. Dropping the tablespace will delete the files.

5. You change the `MINIMUM EXTENT` setting using the `ALTER TABLESPACE` command. What will the new setting apply to?
 - A. All existing extents and new extents in the tablespace.
 - B. New extents for new and existing segments.
 - C. Extents for new segments only.
 - D. The `MINIMUM EXTENT` setting cannot be modified for an existing tablespace.

6. Which of the following statements is true?
 - A. A segment must be located in only one tablespace.
 - B. A segment must be located in only one datafile.
 - C. An extent can be split between multiple datafiles.
 - D. An extent can be shared by multiple segments.

7. How can you configure how much new space will be allocated to a datafile every time it is extended automatically?
 - A. By using the `MAXSIZE` setting
 - B. By using the `NEXT` setting
 - C. By using the `PCTINCREASE` setting
 - D. By using the `CHUNKSIZE` setting

8. Which of the following statements can be used in order to change the size of a tablespace?
 - A. `ALTER TABLESPACE ...ADD FILE`
 - B. `ALTER DATABASE ADD DATAFILE`
 - C. `ALTER DATABASE RESIZE DATAFILE`
 - D. `ALTER DATABASE DATAFILE ... RESIZE`

9. You have mounted the database but did not open it. Which two views do you need to query if you need the locations of all datafiles and the names of the tablespaces they belong to? (Choose two.)
 - A. `V$DATAFILE`
 - B. `DBA_DATA_FILES`
 - C. `V$TABLESPACE`
 - D. `V$CONTROLFILE`

10. Which of the following views can you query if you need to know whether a datafile has been extended after creation?
- A. DBA_DATA_FILES
 - B. DBA_TABLESPACES
 - C. V\$DATAFILE
 - D. V\$TABLESPACE

Scenarios

1. You just received a call from your former employer. After you left (on good terms, of course), they made the decision to use Oracle for their customer information database. Their current system administrator installed and configured Oracle to the best of his ability, but he does not have any Oracle-specific training. Oracle is installed on a small server with one hard drive, but the administrator feels that they have plenty of free space. The database has been created with default settings, then the tables and views were created and data loaded. They are having performance problems and need you to help.
- A. What do you suspect is the main cause of the performance problems?
 - B. Are there any changes in server hardware that you would recommend?
 - C. Which changes in the database configuration would you recommend?
 - D. Are there any settings that cannot be changed now without rebuilding the database?
2. Your database is having problems with disk space. The company is willing to add more drives and asked you how many more they should buy and what data you are planning to use each one for. The database is used for an application with heavy update activity and occasional reporting.
- A. Which tablespaces would you consider putting on separate disks?
 - B. You are asked to keep the database available while you are moving the tablespaces to their new locations. How will you accomplish this?
 - C. Assuming that the new drives are sufficiently big and will not be used for other purposes, how can you make sure that the datafiles get the space they need automatically?
 - D. How can you make sure that all of the extents in the tablespaces are the same size or multiples of each other?
3. A company contacted you for help. They need to decide how to increase the space available to their data. At present, they store all data in the same tablespace that has one datafile. There is plenty of space left on the drive. They need to decide the best way to increase the size of the tablespace.

- A. If they decide not to invest in another drive, should they add a new datafile on the same disk or should they increase the size of the existing datafile?
 - B. If they decide to enable automatic extension for the existing datafile, how can they make sure that it only grows to a specific maximum size?
 - C. Are there any advantages to purchasing another disk and locating a new datafile on it?
4. You get a call from a colleague who suspects that the poor performance of her Oracle database is caused by sorting activity. Which views should she look at in order to monitor the sorting activity on her server? How can she make sure that none of the users sort in a permanent tablespace?

Lab Exercises

Lab 8-1 Creating Tablespaces

1. Create the following tablespaces in your CERTDB database:
 - A. RBS — For rollback segments.
Datafile (NT): C:\CERTDB\DISK3\RBS01.DBF size 10MB
Datafile (UNIX): /CERTDB/DISK3/RBS01.DBF size 10 MB
Make sure that the default storage settings are such that all extents will be created equal by default.
 - B. DATA — For tables.
Datafile (NT): C:\CERTDB\DISK4\DATA01.DBF size 10MB
Datafile (UNIX): /CERTDB/DISK4/DATA01.DBF size 10MB
Make sure that the smallest extent that can be created in this tablespace is 100K.
 - C. INDX — For indexes.
Datafile (NT): C:\CERTDB\DISK2\INDX01.DBF size 15MB
Datafile (UNIX): /CERTDB/DISK2/INDX01.DBF size 15MB
 - D. TEMP — Temporary.
Datafile (NT): C:\CERTDB\DISK5\TEMP01.DBF size 20MB
Datafile (UNIX): /CERTDB/DISK5/TEMP01.DBF size 20MB
Make sure that the TEMP tablespace can only contain temporary objects.

E. RONLY — You will make this tablespace read-only later.

Datafile (NT): C:\CERTDB\DISK6\RONLY01.DBF size 10MB

Datafile (UNIX): /CERTDB/DISK6/RONLY01.DBF size 10MB

- The SYSTEM rollback segment cannot be used to work with data outside of the SYSTEM tablespace. In order to complete the following exercises, you need at least one other rollback segment. Use the following statements in order to create one and bring it online:

```
SVRMGR> CREATE PUBLIC ROLLBACK SEGMENT rbs_01
        2> TABLESPACE RBS;
Statement processed.
SVRMGR> ALTER ROLLBACK SEGMENT rbs_01 ONLINE;
Statement processed.
```



Rollback segments are covered in Chapter 10. Right now, you just need one in order to do the labs.

- Refer to Appendix E for instructions on creating the CERTDB tablespace and populating it with sample data. Using the instructions, execute the CREATEUSER.SQL, CERTDBOBJ.SQL and INSERT_DATA.SQL scripts.

Lab 8-2 Managing the Status of Tablespaces

- Connect to the database as SYSDBA. Use the following statements to create a table in the RONLY tablespace:

```
SVRMGR> CREATE TABLE lab82 TABLESPACE ronly AS
        2> SELECT locationid, locationname
        3> FROM student.locations;
```



Creating tables is covered in Chapter 11. For now, you just need one for this lab.

- Query the LAB82 table to make sure it got created correctly. It should have three rows and two columns.
- Take the RONLY tablespace offline.
- Attempt to query the LAB82 table again. What is the result?
- Bring the RONLY tablespace online again and make it read-only. Check that you can query the LAB82 table.
- Try to perform the following operations on the table:
 - Insert a row.
 - Delete all rows.
 - Update a row.

- D. Truncate the table.
- E. Drop the table.

Which of the above operations succeeded? Why?

7. Make the tablespace read-write and rerun the statement in step 1 to re-create the table.
8. Attempt to take the RBS tablespace offline. Attempt to make it read-only. What happened? Why?
9. Attempt to take the SYSTEM tablespace offline. Attempt to make it read-only. What happened? Why?

Lab 8-3 Relocating, Resizing, and Dropping Tablespaces

1. Relocate the SYSTEM tablespace to the DISK2 directory. Can it be done without shutting down the database? Why?
2. Relocate the RONLY tablespace to the DISK1 directory. Can it be done without shutting down the database? How?
3. Check that the LAB82 table is intact and delete the old copy of the datafile.
4. Allocate additional space to the RONLY tablespace by increasing the size of its datafile to 12MB. Can it be done while the tablespace is online? If necessary, could you resize a datafile that belongs to the SYSTEM tablespace without shutting down the database?
5. Make the RONLY tablespace read-only and try to resize its datafile to 15MB. What happens? Why?
6. Allocate additional space to the INDX01 tablespace by allowing its datafile to increase automatically. Ensure that it does not grow beyond 20MB and that it grows 1MB at a time.
7. Allocate additional space to the DATA tablespace by adding a datafile to it. Name the new datafile DATA02.dbf, make its size 10MB, and place it in DISK6. What is the total size of the DATA tablespace now?
8. Using dynamic performance (V\$) and data dictionary (DBA_) views, find out which datafiles can extend automatically and which datafiles have been extended after their creation. Have any extended automatically?
9. Drop the RONLY tablespace. Which option do you need to use? Why?
10. Check whether the RONLY01 data file has been deleted. Has it?
11. Re-create the RONLY tablespace using the original datafile. Which keyword do you need to use? Is the LAB82 table still there?

Answers to Chapter Questions

Chapter Pre-Test

1. The tablespace is the link between the logical and physical structures of the database. Using tablespaces, you can specify the location of segments and manage subsets of the database. A tablespace can be made read-only or be taken offline altogether. You also use tablespaces in order to separate segments that have different requirements or those that cause resource contention or excess fragmentation.
2. The database physical structure contains two main elements: datafiles and operating system blocks. A database can contain more than one data file, but datafiles cannot belong to more than one database.
3. From the logical point of view, a database consists of tablespaces, segments, extents, and database blocks. A tablespace is the highest unit of logical organization within the database. It provides a location for segments, which are database objects that are allocated storage space. The segments consist of extents, which are contiguous groups of database blocks. The database block is the minimum unit of IO for Oracle.
4. There are many ways to classify tablespaces. First, there is SYSTEM and everything else. Then, tablespaces can be classified based on their contents. Temporary tablespaces cannot contain any permanent objects and are only used for temporary segments. Permanent tablespaces can contain permanent and temporary objects, but it is best to keep temporary segments in temporary tablespaces. Tablespaces can also be classified according to the space management mechanism used: locally managed or dictionary-managed.
5. You can create a tablespace by using the CREATE TABLESPACE or CREATE TEMPORARY TABLESPACE commands. When creating a tablespace, you will need to specify at least one datafile. You can also create a tablespace by using the Storage Manager component of Oracle Enterprise Manager.
6. In order to make a tablespace unavailable, it must be taken offline. You can use the ALTER TABLESPACE command or Storage Manager. The SYSTEM tablespace or a tablespace that contains active rollback segments cannot be taken offline.
7. You can prevent data in a tablespace from getting changed by making it read-only. The SYSTEM tablespace or a tablespace containing active rollback segments cannot be made read-only. A read-only tablespace does not have to be backed up as frequently as read-write tablespaces. You can use the ALTER TABLESPACE command or Storage Manager in order to make a tablespace read-only.
8. In order to increase the size of a tablespace, you need to increase the total size of the datafiles belonging to it. You can do this by adding a datafile to the tablespace, resizing a datafile, or allowing one or more datafiles to extend

automatically. You can use the ALTER TABLESPACE command to add a datafile and the ALTER DATABASE command to resize a datafile or change its AUTOEXTEND settings. You can also use Storage Manager.

9. You can change the storage settings for a tablespace by using the ALTER TABLESPACE command or Storage Manager. The new settings will affect only the future space allocations, not the existing data.
10. In order to enforce a minimum size for all extents in a tablespace, you need to use the MINIMUM EXTENT setting. Although you can use the DEFAULT STORAGE clause to specify default sizes, they can be overridden when a segment is created.
11. In order to relocate a non-SYSTEM tablespace, you should take it offline, copy the datafile(s), and use the ALTER TABLESPACE ... RENAME DATAFILE command. You can use Storage Manager to perform this operation, as well.
12. In order to relocate the SYSTEM tablespace, you need to mount the database, copy the datafile(s) to the new location, and use the ALTER DATABASE RENAME FILE command. You can do most of this with Oracle Enterprise Manager, as well.
13. Temporary segments are used in order to provide space for sorting and to temporarily store data in temporary tables. If temporary segments are allowed to exist in permanent tablespaces, they will fragment the space in that tablespace. In temporary tablespaces, temporary segments are shared, and space is not deallocated until the instance is shut down. You should always create a temporary tablespace and make sure that it is assigned to all users.

Assessment Questions

1. **B.** In order to move a datafile that belongs to the SYSTEM tablespace, you have to shut the database down, move or copy the file, then mount the database and use the ALTER DATABASE RENAME FILE command. The commands in answers A and C generate errors. The command in answer D can only be used when the tablespace is offline and the database is open and cannot be used in order to relocate the SYSTEM tablespace.
2. **D.** A tablespace cannot be taken offline if it contains active rollback segments. If you take all of the rollback segments in the tablespace offline, you can take the tablespace itself offline, so answer B is wrong. Answer A is incorrect because read-only tablespaces can be taken offline. Temporary tablespaces can also be taken offline, so answer C is wrong. The only tablespace that can never be taken offline is SYSTEM.
3. **A.** In order to move a tablespace to a CD_ROM, you need to make it read-only first. A tablespace cannot be made read-only while it is offline, so answers B and C are wrong. Answer D is impossible because Oracle needs to perform a checkpoint when a tablespace is made read-only. If the tablespace is already on the CD-ROM, Oracle will not be able to write to it, and you will receive an error.

4. **A.** When creating a tablespace when the datafiles already exist, you need to specify REUSE instead of SIZE in the datafile specification. Answer B mentions a nonexistent option. Answer C is incorrect—you can reuse existing files. You will not recover the data that used to be stored there, however. Answer D is incorrect because the question specifically stated that the tablespace has been dropped.
5. **B.** When you change the MINIMUM EXTENT for a tablespace that already contains segments, the existing extents are not resized, so answer A is incorrect. All new extents will have to conform to the new MINIMUM EXTENT, whether they belong to new or existing segments. This makes answer C incorrect. The MINIMUM EXTENT setting can be changed for an existing tablespace, so answer D is incorrect.
6. **A.** A segment can belong to only one tablespace. The extents that form it, however, can be located in different datafiles that belong to the same tablespace. This makes answer B incorrect. Each extent must be located within one datafile, so answer C is incorrect. Segments do not share extents in Oracle, so answer D is also wrong.
7. **B.** The NEXT setting is used when enabling a datafile for automatic extension in order to control how much space will be allocated to the datafile every time it is extended. The MAXSIZE setting (answer A) is used in order to specify the maximum size the datafile will be allowed to grow to. The PCTINCREASE and CHUNKSIZE do not apply to datafiles, so answers C and D are incorrect.
8. **D.** There are three ways to resize a tablespace: add a new datafile, resize an existing datafile, or enable a datafile for automatic extension. The only valid command out of the ones presented in the answers is the last one, ALTER DATABASE DATAFILE ... RESIZE. All other answers list invalid commands.
9. **A.** and **C.** The only views you can query when the database is only mounted are the V\$ views, because the DBA_ views are not available unless the database is open. This makes answer B incorrect. You can use the V\$DATAFILE in order to list the names and locations of the datafiles, but it won't show you the names of the tablespaces—only their numbers. In order to see the names, you can query the V\$TABLESPACE—or join the views on the TS# column. The V\$CONTROLFILE view does not contain any information on datafiles or tablespaces.
10. **C.** In order to see whether a datafile has been extended since its creation, you can use the V\$DATAFILE view: Compare the values in the BYTES and CREATE_BYTES columns for the same datafile to see if they are different. The DBA_DATA_FILES view (answer A) tells you whether the file is currently enabled for automatic extension, not whether its size has changed. The DBA_TABLESPACES view (Answer B) contains no information on datafile size, current or past. The V\$TABLESPACE view only shows the names and numbers of tablespaces.

Scenarios

1. Where do we start? Installing Oracle with defaults results in a database with a 2KB block size, one tablespace (SYSTEM), and all objects (including temporary ones) created in that tablespace.

- A. The main cause of performance problems is the single tablespace on a single disk. The problems themselves are numerous: inefficient IO, a fragmented SYSTEM tablespace, and insufficient rollback segments, just to name a few.
- B. The first change should be to invest in some hard drives. The actual number may vary depending on the workload type, but one is definitely not enough.
- C. Once the new hard drives are installed, you should decide on how to allocate space on them to the database. You should definitely create some new tablespaces: temporary, rollback, data, and index to begin with, provided there are no special requirements for anything else. Data and index tablespaces should be placed on different disks to improve IO. All tablespaces should be created with proper DEFAULT STORAGE and MINIMUM EXTENT settings. Location of other tablespaces can be based on activity. You can use the operating system utilities in order to measure disk IO and relocate the tablespaces to the least-busy disks.

Users should be assigned the temporary tablespace. Because the database was created with all defaults, all users are using SYSTEM for sorting activity. In order to improve sorting performance and minimize the fragmentation of space in the SYSTEM tablespace, you should use the ALTER USER command to assign them a proper temporary tablespace.

Another change will be to relocate existing segments to proper tablespaces once they are created.

- D. The settings that cannot be changed after the creation of the database are the block size and character sets. Since the company's concern is performance, not language-related problems, you can assume that they are fine with US7ASCII—the default character set. The block size, however, is likely part of the performance problems. As a rule, 2KB is too small for most databases. However, there is not much you can do about it now without re-creating the database from scratch.

All in all, this is a good example of what happens when a database is created without proper planning. Although it can be “fixed,” some of the settings can no longer be changed. Also, a lot of damage has already been done to the SYSTEM tablespace. It will also take a lot of time to relocate indexes and tables to new tablespaces. You may want to propose that the current data be exported from the database and the database be rebuilt from scratch. This may take less time than fixing it, and it will result in a better database.

2. They need to buy 21 new hard drives. Just kidding! A question like “How many drives do you need?” cannot be answered without a lot more information gathered through thorough monitoring of the server. There are, however, some questions you can answer right now.
 - A. The tablespaces that should be located on separate disks are the ones accessed simultaneously and the ones that see constant heavy activity. Usually, you will want to make sure that data and index tablespaces are separated, because they are often accessed at the same time.
 - B. Because you are unlikely to move the SYSTEM tablespace, you may be able to make most of the changes while the database is online. You need to take tablespaces offline one by one and use the ALTER TABLESPACE ... RENAME DATAFILE command in order to relocate them.
 - C. You may want to consider allowing the datafiles to extend automatically. Make sure you use the NEXT and MAXSIZE parameters in order to control the datafiles’ growth.
 - D. You can use the MINIMUM EXTENT setting in order to ensure that all extents in the tablespace are created equal or as multiples of each other.
3. This is an example where you need to step back and ask whether what your client is asking you to do is really the best solution. They are asking you how to increase the size of the tablespace. What they should be doing is creating another tablespace on another disk and at least moving the indexes there.
 - A. There is no reason to add a new datafile to a tablespace if the datafile is going to be located on the same drive. In this case, you should simply resize the existing datafile.
 - B. In order to prevent a datafile from growing indefinitely, use the MAXSIZE setting.
 - C. Purchasing a new disk will enable you to physically separate data. This has a number of positive effects, such as performance (data can now be accessed in parallel) and recoverability in case of a disk failure (you will only lose part of the database if the disk fails). Overall, this may be the course of action you should recommend to the company.
4. Your colleague should start by checking the TEMPORARY_TABLESPACE column in the DBA_USERS view to see if any of the users are not assigned a proper temporary tablespace. If any users need to be modified, she can use the ALTER USER command. After this, she can monitor the sorting activity by querying the V\$SORT_SEGMENT and V\$SORT_USAGE views.

Lab Exercises

Lab 8-1

1. To create tablespaces on Windows 2000 or NT, open a command prompt and start Server Manager (svrmgrl). Connect as a privileged user and issue the following commands:

```
SVRMGR> CREATE TABLESPACE RBS
      2> DATAFILE 'C:\CERTDB\DISK3\RBS01.DBF' SIZE 10M
      3> DEFAULT STORAGE (PCTINCREASE 0);
Statement processed.
SVRMGR> CREATE TABLESPACE DATA
      2> DATAFILE 'C:\CERTDB\DISK4\DATA01.DBF' SIZE 10M
      3> MINIMUM EXTENT 100K;
Statement processed.
SVRMGR> CREATE TABLESPACE INDX
      2> DATAFILE 'C:\CERTDB\DISK2\INDX01.DBF' SIZE 15M;
Statement processed.
SVRMGR> CREATE TEMPORARY TABLESPACE TEMP
      2> TEMPFILE 'C:\CERTDB\DISK5\TEMP01.DBF' SIZE 20M;
Statement processed.
SVRMGR> CREATE TABLESPACE RONLY
      2> DATAFILE 'C:\CERTDB\DISK6\RONLY01.DBF' SIZE 10M;
Statement processed.
```

To create tablespaces on a UNIX system, open a shell window and start Server Manager (svrmgrl). Connect as a privileged user and issue the following commands:

```
SVRMGR> CREATE TABLESPACE RBS
      2> DATAFILE '/CERTDB/DISK3/RBS01.DBF' SIZE 10M
      3> DEFAULT STORAGE (PCTINCREASE 0);
Statement processed.
SVRMGR> CREATE TABLESPACE DATA
      2> DATAFILE '/CERTDB/DISK4/DATA01.DBF' SIZE 10M
      3> MINIMUM EXTENT 100K;
Statement processed.
SVRMGR> CREATE TABLESPACE INDX
      2> DATAFILE '/CERTDB/DISK2/INDX01.DBF' SIZE 15M;
Statement processed.
SVRMGR> CREATE TEMPORARY TABLESPACE TEMP
      2> TEMPFILE '/CERTDB/DISK5/TEMP01.DBF' SIZE 20M;
Statement processed.
SVRMGR> CREATE TABLESPACE RONLY
      2> DATAFILE '/CERTDB/DISK6/RONLY01.DBF' SIZE 10M;
Statement processed.
```

2. In the Server Manager session, issue the commands to create the rollback segment and bring it online:

```
SVRMGR> CREATE PUBLIC ROLLBACK SEGMENT rbs_01
      2> TABLESPACE RBS;
```

```
Statement processed.
SVRMGR> ALTER ROLLBACK SEGMENT rbs_01 ONLINE;
Statement processed.
```

3. Follow the instructions in Appendix E to create the CERTDB tablespace, the STUDENT user, and sample data.

Lab 8-2

1. In the Server Manager session, connect as a SYSDBA because the scripts that created the sample data left you logged in as user STUDENT:

```
SVRMGR> CONNECT / AS SYSDBA;
Connected.
```

2. In the Server Manager session, issue the following query:

```
SVRMGR> SELECT * FROM lab82;
LOCATIONID LOCATIONNAME
-----
          100 New York Park Ave
          200 San Francisco Downtown
          300 Downtown Toronto
3 rows selected.
```

3. In order to take the tablespace offline, issue the following statement:

```
SVRMGR> ALTER TABLESPACE ronly OFFLINE;
Statement processed.
```

4. When you try to query the table, you get an error. The tablespace that contains the table is offline, and the data in it is unavailable:

```
SVRMGR> SELECT * FROM lab82;
LOCATIONID LOCATIONNAME
-----
ORA-00376: file 6 cannot be read at this time
ORA-01110: data file 6: 'C:\CERTDB\DISK6\RONLY01.DBF'
```

5. In order to bring the tablespace online, use the ALTER TABLESPACE statement:

```
SVRMGR> ALTER TABLESPACE ronly ONLINE;
Statement processed.
SVRMGR> ALTER TABLESPACE ronly READ ONLY;
Statement processed.
SVRMGR> SELECT * FROM lab82;
LOCATIONID LOCATIONNAME
-----
          100 New York Park Ave
          200 San Francisco Downtown
          300 Downtown Toronto
3 rows selected.
```

6. In the Server Manager session, try to execute the following statements:

```
SVRMGR> INSERT INTO lab82 VALUES (400, 'New West Downtown');
ORA-00372: file 6 cannot be modified at this time
ORA-01110: data file 6: 'C:\CERTDB\DISK6\RONLY01.DBF'
SVRMGR> DELETE FROM lab82;
ORA-00372: file 6 cannot be modified at this time
ORA-01110: data file 6: 'C:\CERTDB\DISK6\RONLY01.DBF'
SVRMGR> TRUNCATE TABLE lab82;
TRUNCATE TABLE lab82
          *
ORA-00372: file 6 cannot be modified at this time
ORA-01110: data file 6: 'C:\CERTDB\DISK6\RONLY01.DBF'
SVRMGR> UPDATE lab82 SET locationid = 400;
ORA-00372: file 6 cannot be modified at this time
ORA-01110: data file 6: 'C:\CERTDB\DISK6\RONLY01.DBF'
SVRMGR> DROP TABLE lab82;
Statement processed.
```

You cannot insert, update, or delete any data in a read-only tablespace. However, you can drop a table because the change actually happens in the data dictionary, which is always writable.

7. In your Server Manager window, issue these commands:

```
SVRMGR> ALTER TABLESPACE ronly READ WRITE;
Statement processed.
SVRMGR> CREATE TABLE lab82 TABLESPACE ronly AS
      2> SELECT locationid, locationname
      3> FROM student.locations;
```

8. Use the ALTER TABLESPACE command to try and take the RBS tablespace offline and then try to make it read-only:

```
SVRMGR> ALTER TABLESPACE rbs OFFLINE;
ALTER TABLESPACE rbs OFFLINE
*
ORA-01546: tablespace contains active rollback segment 'RBS0'
SVRMGR> ALTER TABLESPACE rbs READ ONLY;
ALTER TABLESPACE rbs READ ONLY
*
ORA-01546: tablespace contains active rollback segment 'RBS0'
```

The RBS tablespace cannot be taken offline at this time because it contains an active rollback segment.

9. Use the ALTER TABLESPACE command to try and take the RBS tablespace offline and then try to make it read-only:

```
SVRMGR> ALTER TABLESPACE system OFFLINE;
ALTER TABLESPACE system OFFLINE
*
ORA-01541: system tablespace cannot be brought offline; shut
down if necessary
```

```
SVRMGR> ALTER TABLESPACE system READ ONLY;
ALTER TABLESPACE system READ ONLY
*
ORA-01643: system tablespace can not be made read only
```

The SYSTEM tablespace cannot be taken offline or made read-only because it contains the data dictionary.

Lab 8-3

1. The SYSTEM tablespace cannot be relocated without shutting down the database because it cannot be taken offline. You must shut down the database, copy the file, mount the database, and use the ALTER DATABASE RENAME FILE command.

On a Windows 2000 or NT system, issue the following commands in the Server Manager session while logged in as a SYSDBA:

```
SVRMGR> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR> host
C:\>COPY C:\CERTDB\DISK1\SYSTEM01.DBF C:\CERTDB\DISK2
      1 file(s) copied.
C:\>exit
SVRMGR> STARTUP MOUNT PFILE=C:\CERTDB\INITCERT.ORA
ORACLE instance started.
Total System Global Area                4818188 bytes
Fixed Size                               70924 bytes
Variable Size                           3850240 bytes
Database Buffers                        819200 bytes
Redo Buffers                             77824 bytes
Database mounted.
SVRMGR> ALTER DATABASE RENAME FILE
      2> 'C:\CERTDB\DISK1\SYSTEM01.DBF' TO
      3> 'C:\CERTDB\DISK2\SYSTEM01.DBF';
Statement processed.
SVRMGR> ALTER DATABASE OPEN;
Statement processed.
```

On a UNIX-based system, use the following commands while logged in as a user with SYSDBA privileges:

```
SVRMGR> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR> host
# cp /CERTDB/DISK1/SYSTEM01.DBF /CERTDB/DISK2
# exit
```

```

SVRMGR> STARTUP MOUNT PFILE=/CERTDB/INITCERT.ORA
ORACLE instance started.
Total System Global Area                4818188 bytes
Fixed Size                               70924 bytes
Variable Size                           3850240 bytes
Database Buffers                        819200 bytes
Redo Buffers                             77824 bytes
Database mounted.
SVRMGR> ALTER DATABASE RENAME FILE
      2> '/CERTDB/DISK1/SYSTEM01.DBF' TO
      3> '/CERTDB/DISK2/SYSTEM01.DBF';
Statement processed.
SVRMGR> ALTER DATABASE OPEN;
Statement processed.

```

Once you are done and the database opened normally, you can delete the old file.

On Windows NT, the following command deletes the old datafile:

```
SVRMGR> host DEL C:\CERTDB\DISK1\SYSTEM01.DBF
```

On a UNIX-based system, use this command:

```
SVRMGR> host rm /CERTDB/DISK1/SYSTEM01.DBF
```

2. The RONLY tablespace can be relocated without shutting down the database. First, you will need to take it offline. Then, you can copy the file to the new location. Finally, you can bring the tablespace back online:

Here is the sequence of steps you should take if you are using a Windows 2000 or NT system:

```

SVRMGR> ALTER TABLESPACE ronly OFFLINE;
Statement processed.
SVRMGR> host
C:\>COPY C:\CERTDB\DISK6\RONLY01.DBF C:\CERTDB\DISK1
      1 file(s) copied.
C:\>exit
SVRMGR> ALTER TABLESPACE ronly RENAME DATAFILE
      2> 'C:\CERTDB\DISK6\RONLY01.DBF' TO
      3> 'C:\CERTDB\DISK1\RONLY01.DBF';
Statement processed.
SVRMGR> ALTER TABLESPACE ronly ONLINE;
Statement processed.

```

Here is the same procedure for a UNIX system:

```

SVRMGR> ALTER TABLESPACE ronly OFFLINE;
Statement processed.
SVRMGR> host
# cp /CERTDB/DISK6/RONLY01.DBF /CERTDB/DISK1

```

```
# exit
SVRMGR> ALTER TABLESPACE ronly RENAME DATAFILE
      2> '/CERTDB/DISK6/RONLY01.DBF' TO
      3> '/CERTDB/DISK1/RONLY01.DBF';
Statement processed.
SVRMGR> ALTER TABLESPACE ronly ONLINE;
Statement processed.
```

3. Although you could have just moved the datafile to the new location, I prefer copying the files first and deleting them later — this way, I am sure that the operation went well before I start deleting datafiles. As a rule of thumb, when I need to choose between different methods of doing the same job, I choose the one that lets me sleep better at night.

In your Server Manager session, run any query on the LAB82 table:

```
SVRMGR> SELECT * FROM lab82;
LOCATIONID LOCATIONNAME
-----
          100 New York Park Ave
          200 San Francisco Downtown
          300 Downtown Toronto
3 rows selected.
```

Now that you are sure everything is fine, you can delete the old datafile.

On a Windows 2000 or NT system, use this command:

```
SVRMGR> host DEL C:\CERTDB\DISK6\RONLY01.DBF
```

On a UNIX system, use this one:

```
SVRMGR> host rm /CERTDB/DISK6/RONLY01.DBF
```

4. Resizing a datafile can be done while the database is open and the tablespace is online. You can also resize datafiles that belong to the SYSTEM tablespace, even when the database is open.

On a Windows 2000 or NT system, use this statement in your Server Manager session:

```
SVRMGR> ALTER DATABASE DATAFILE
      2> 'C:\CERTDB\DISK1\RONLY01.DBF' RESIZE 12 M;
Statement processed.
```

In order to resize a datafile on a UNIX system, use this command:

```
SVRMGR> ALTER DATABASE DATAFILE
      2> '/CERTDB/DISK1/RONLY01.DBF' RESIZE 12 M;
Statement processed.
```

5. You cannot resize the datafiles belonging to read-only tablespaces.

To try to do it on a Windows 2000 or NT machine, issue this command in your Server Manager session:

```
SVRMGR> ALTER DATABASE DATAFILE
      2> 'C:\CERTDB\DISK1\RONLY01.DBF' RESIZE 15 M;
ALTER DATABASE DATAFILE
*
ORA-02495: cannot resize file C:\CERTDB\DISK1\RONLY01.DBF,
tablespace RONLY is read only
```

For a UNIX-based system, issue this command at the Server Manager prompt:

```
SVRMGR> ALTER DATABASE DATAFILE
      2> '/CERTDB/DISK1/RONLY01.DBF' RESIZE 15 M;
ALTER DATABASE DATAFILE
*
ORA-02495: cannot resize file /CERTDB/DISK1/RONLY01.DBF,
tablespace RONLY is read only
```

If you need to resize a datafile belonging to a read-only tablespace, make it read write first, then resize the datafile.

- 6. In order to change the datafile parameters, you need to use the ALTER DATABASE command.**

For a Windows 2000 or NT system, the command looks like this:

```
SVRMGR> ALTER DATABASE
      2> DATAFILE 'C:\CERTDB\DISK2\INDX01.DBF'
      3> AUTOEXTEND ON NEXT 1M MAXSIZE 20M;
Statement processed.
```

For a UNIX-based system, this is the command you can use:

```
SVRMGR> ALTER DATABASE
      2> DATAFILE '/CERTDB/DISK2/INDX01.DBF'
      3> AUTOEXTEND ON NEXT 1M MAXSIZE 20M;
Statement processed.
```

The NEXT setting ensures that the file extends by 1MB at a time, and the MAXSIZE settings limits its maximum size to 20MB.

- 7. You need to use the ALTER TABLESPACE ... ADD DATAFILE command in order to increase the size of a tablespace by adding a datafile to it.**

Here is the command for a Windows 2000 or NT system:

```
SVRMGR> ALTER TABLESPACE DATA ADD DATAFILE
      2> 'C:\CERTDB\DISK6\DAT02.DBF' SIZE 10M;
Statement processed.
```

This is the command that adds a datafile to an existing tablespace on a UNIX system:

```
SVRMGR> ALTER TABLESPACE DATA ADD DATAFILE
      2> '/CERTDB/DISK6/DATA02.DBF' SIZE 10M;
Statement processed.
```

The total size of the tablespace is the sum of the sizes of its datafiles. You can find this by querying the V\$DATAFILE or the DBA_DATA_FILES views:

```
SVRMGR> SELECT SUM(bytes) FROM V$DATAFILE
        2> WHERE ts# =
        3> (SELECT ts# FROM V$TABLESPACE WHERE
        4> name = 'DATA');
SUM(BYTES)
-----
20971520
1 row selected.
```

Or:

```
SVRMGR> SELECT SUM(bytes) FROM DBA_DATA_FILES WHERE
        2> tablespace_name = 'DATA';
SUM(BYTES)
-----
20971520
1 row selected.
```

The total size of the DATA tablespace is currently 20MB.

8. In order to find out which files can extend automatically, you can query the DBA_DATA_FILES view. You can even see the maximum size and the NEXT setting:

```
SVRMGR> select file_name, autoextensible, maxbytes,
        2> increment_by FROM DBA_DATA_FILES;
FILE_NAME                                     AUT MAXBYTES      INCREMENT_
-----
C:\CERTDB\DISK2\SYSTEM01.DBF                YES 1.7180E+10      1280
C:\CERTDB\DISK3\RBS01.DBF                   NO   0                0
C:\CERTDB\DISK5\TEMP01.DBF                  NO   0                0
C:\CERTDB\DISK2\INDX01.DBF                   YES  20971520        128
C:\CERTDB\DISK4\DATA01.DBF                  NO   0                0
C:\CERTDB\DISK1\RONLY01.DBF                 NO   0                0
C:\CERTDB\CERTDB01.DBF                      NO   0                0
C:\CERTDB\DISK6\DATA02.DBF                  NO   0                0
8 rows selected.
```

The value for the maximum size of the SYSTEM01.DBF file means that the maximum size for the datafile has not been set.

In order to find out which files have been extended since their creation, use the V\$DATAFILE view:

```
SVRMGR> select name, bytes, create_bytes
        2> from V$DATAFILE where
        3> bytes != create_bytes;
NAME                                     BYTES      CREATE_BYT
-----
C:\CERTDB\DISK1\RONLY01.DBF              12582912   10485760
1 row selected.
```

The only datafile that has been extended is the one that belongs to the RONLY tablespace. You did this manually in step 4 of this lab. The query shows that when the file was created, it was 10MB and is now 12MB.

9. In order to drop the RONLY tablespace, you need to use the INCLUDING CONTENTS option because it contains a table:

```
SVRMGR> DROP TABLESPACE ronly;
DROP TABLESPACE ronly
*
ORA-01549: tablespace not empty, use INCLUDING CONTENTS
option
SVRMGR> DROP TABLESPACE ronly INCLUDING CONTENTS;
Statement processed.
```

10. In order to re-create the tablespace and reuse an existing datafile, you need to use the CREATE TABLESPACE command with the REUSE keyword in the datafile specification:

For a Windows 2000 or NT system, issue this command in the Server Manager session:

```
SVRMGR> CREATE TABLESPACE ronly
2> DATAFILE 'C:\CERTDB\DISK1\RONLY01.DBF' REUSE;
Statement processed.
```

Using a UNIX-based system, the command looks like this:

```
SVRMGR> CREATE TABLESPACE ronly
2> DATAFILE '/CERTDB/DISK1/RONLY01.DBF' REUSE;
Statement processed.
```

The LAB82 table that used to exist in the original RONLY tablespace is gone, of course:

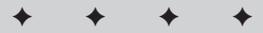
```
SVRMGR> SELECT * FROM lab82;
SELECT * FROM lab82
*
ORA-00942: table or view does not exist
```


Administering Storage for Oracle Objects

Knowing how to create a database, tablespaces, and datafiles is only one part of administering Oracle. You also need to be aware of issues surrounding the proper location of various objects in Oracle that require storage (segments). This part of the book is divided into five chapters that first introduce you to the different types of segments in an Oracle database, then follows with a discussion on each segment's particular storage needs, and finishes with a discussion on how to enforce data integrity in the database and the impact of constraints.

Chapter 9 provides a general overview of the logical storage structure of an Oracle database. It first lists and describes the types of segments that you can create in Oracle. This is followed by a discussion on how storage parameters are specified in Oracle as well as how they are inherited from parent objects. It concludes with a discussion on the management of extents, database blocks, and how to retrieve storage information about objects from the data dictionary.

In Chapter 10, you will learn how rollback segments play a critical role in the performance of transactional databases such as order entry systems. The different types of rollback segments will be explained, as well as how transactions make use of them. You will be shown how to create rollback segments and how to bring them online manually, or have them brought online automatically when an instance starts. The chapter ends with a discussion on how to properly plan rollback segments, after a discussion on troubleshooting any problems you may encounter.



In This Part

Chapter 9

Storage Structure and Relationships

Chapter 10

Managing Rollback Segments

Chapter 11

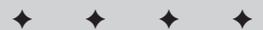
Managing Tables

Chapter 12

Managing Indexes

Chapter 13

Maintaining Data Integrity



The basic object in any database is a table. Chapter 11 starts by discussing how individual data is stored in Oracle, and the different data types that Oracle supports and their impact on storage. You will then learn how to create a table on a tablespace, as well as on a cluster. After a table is created, changes may need to be made — such as adding or removing columns — which is the subject of the next part of the chapter. Finally, you will learn how data dictionary views can be used to get information on your tables, their structure, and storage characteristics.

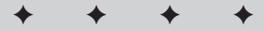
Tables alone may not provide the best performance when querying a database, so a discussion on indexes is required. In Chapter 12, you will learn about the different types of indexes available in Oracle, their uses, and their structure. The creation and modification of indexes will also be discussed, as well as good index maintenance practices. Finally, data dictionary views that provide information on indexes will be introduced and discussed.

Although a discussion on maintaining database integrity using constraints may not necessarily seem appropriate in a book for database administration, the topic is covered on the exam and is discussed in Chapter 13. Perhaps a key reason that data integrity is discussed in a book on database administration is that, as a DBA, this is your responsibility. Understanding the different constraints available is important information for a DBA. Furthermore, if you need to perform bulk modifications on the database, constraints may affect your ability to do so, in which case understanding how to enable and disable constraints, how to find rows that violate constraints, as well as being able to find constraint information using data dictionary views is important.

Storage Structure and Relationships

EXAM OBJECTIVES

- ◆ Storage Structure and Relationships
 - Describe the logical structure of the database
 - List the segment types and their uses
 - List the keywords that control block space usage
 - Obtain information about storage structures from the data dictionary
 - List the criteria for separating segments



CHAPTER PRE-TEST

1. What effect does specifying UNIFORM SIZE when creating a tablespace have on the storage clause of a segment?
2. To reduce the overhead of dynamic extent allocation, should you create a locally managed or dictionary-managed tablespace?
3. What is the meaning of the PCTUSED parameter of a table's storage clause?
4. What is the meaning of the PCTFREE parameter of an index's storage clause?
5. What kind of segments are supported by Oracle8i?
6. What storage clause settings will apply to a table when it is created?
7. What data dictionary view or views would you query to determine the space occupied by a segment?
8. What is a transaction slot and how is the minimum number of transaction slots for a segment specified?
9. What happens if more rows on a database block are being modified than transaction slots available?
10. When deciding which segments should coexist on which tablespace, what are some of the considerations to take into account?

One of the advantages of Oracle as a database management system is the ability for the DBA to be very precise regarding the storage structure of different segment types. In Chapter 8 you found out that Oracle storage is divided into physical and logical components. On the physical side, a database consists of one or more data files that are associated with a logical element — the tablespace. In this chapter you will find out how logical storage is organized within tablespaces and how to optimize the allocation of storage within the database.

Overview of Logical Storage Structure Components

Objective

Describe the logical structure of the database

In an Oracle database, a tablespace consists of one or more data files. At the operating system level, this is all that you see—a bunch of disk files. Like a good book, the real elegance and content is between the covers, that is in the data files themselves.

Figure 9-1 shows both the physical and logical structure of the database. As you can see, a tablespace itself is composed of segments, which are made up of extents, which are composed of one or more database blocks. A segment is any object in an Oracle database that requires physical storage, such as a table, index, and so on.

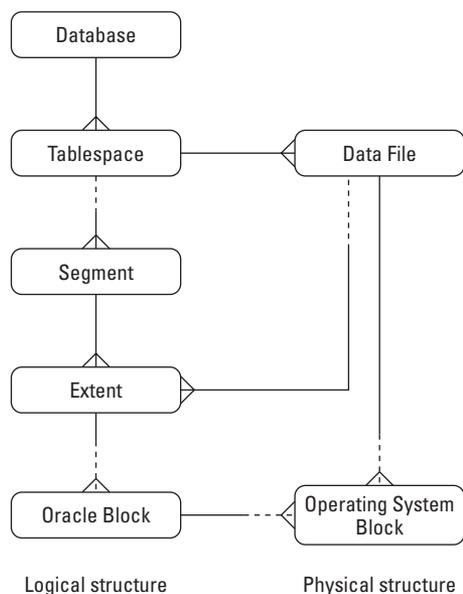


Figure 9-1: The entity relational model of Oracle database storage

When creating any segment, the developer or DBA specifies how big a chunk of disk space should be allocated to the segment. This allocation is known as an extent—a group of contiguous database blocks. Any object must contain at least one extent and may have multiple extents, the size of which you can control. Furthermore, as rows are added to a table or index entry to an index, they will occupy space on a data block. Oracle allows you to also control how space within individual blocks is managed.

Types of Segments

Objective

List the segment types and their uses

Any object that occupies space in an Oracle database is considered a segment. This broad definition includes a large number of Oracle objects. These include tables, indexes, table or index partitions, clusters, rollback segments, temporary segments, LOB segments, LOB indexes, nested tables, and the Oracle bootstrap segment. Many of these will be covered in detail in future chapters, but each is briefly discussed here as well.

Table

The *table* is the most common segment in any Oracle database. You will usually store your data in tables, which are really a collection of columns and rows. While tables can be clustered or partitioned, a table segment is neither. Any segment of table type stores all of its data in a single tablespace and in no particular order. Oracle itself manages the individual data on the blocks, and the DBA, aside from specifying some parameters for the size of extents and the allocation of free space on a block, has little control over where any particular row is stored.

Cross-Reference

Tables are covered in more detail in Chapter 11.

Index

An *index* segment is used to store physical entries associated with an index key defined on a column or columns of a table. The entries can be stored as a B-tree or a bitmap, and are always used to look up the corresponding rows of a table. An index segment is also stored in a single tablespace, and the DBA has relatively little control over how data at the block level is physically organized—Oracle maintains the index itself.

Cross-Reference

Indexes are covered in more detail in Chapter 12.

Cluster

A *cluster* is a segment on which rows from two or more tables are stored on the same database block. Using clusters enables a DBA to physically co-locate related rows. For example, if you had an Orders table that stored the bill to and ship to address, as well as order number and other information for a sales order, and you also had an OrderDetails table that stored the individual line items that made up the order, using a cluster you can have both tables' data stored on the same physical block. This would improve performance when retrieving data from both the Orders and OrderDetails tables since Oracle would only need to read the block once to get data from both tables.

Oracle supports *index clusters* and *hash clusters*. Data on an index cluster is physically stored according to the key value, such as an OrderNumber, that is shared by all tables defined on the cluster. Data on a hash cluster is stored according to the calculated hash value for the columns on which the hash key is defined.



Index and hash clusters, and creating tables on a cluster are covered in more detail in Chapter 11.

Index-organized table

Introduced in Oracle8, an index-organized table is a merger of an index and a table into a single segment. An index-organized table stores data in the order of the index key, much the same way as an index is organized. However, unlike an index where the leaf level blocks contain the key value and pointers to rows in a table, the leaf level of an index-organized table stores the key value and the values for all other columns in the row. A full table scan of an index-organized table always returns data in the index order, and also does not have to do a second IO operation to retrieve the rest of the columns' values — they will be stored in the index itself. Index-organized tables are ideally suited to be used for dimension tables in a data warehousing database.

Partitions

Oracle introduced partitions in Oracle8. The idea behind partitions was to allow the DBA to physically break up a single table and store part of the data on different tablespaces. In this way, it would be possible for a DBA to predict which rows would be stored on which tablespace, and, consequently, which data files and physical hard disks. This would then enable the DBA to balance out IO among multiple disk drives, as well as overcome any limitations on the maximum size of a hard disk partition or data file. Essentially, you can now have really big tables with many billions of rows, and predict where those rows will be stored.

The physical placement of rows and the division of data can be performed by specifying which range of values for a column of columns would be located on which partition, or by a hashing algorithm, or by a combination of the two. Oracle, therefore, supports range partitions, hash partitions, and composite partitioning—also known as sub-partitioning. Both indexes and tables can be partitioned using any of the methods available. Each partition is a separate segment and can have its own storage characteristics.



Partitioning and the proper definition and use of partitions is beyond the scope of the exam and this book. The information provided here is designed to introduce you to the concept of partitioning and provide some insight as to when and how to use it. For more information on partitioning, consult the *Oracle8i Administrator's Guide*.

Range partitions

When partitioning a table or an index using *range partitioning*, the DBA defines a partition key column or columns, and which values will be stored in each partition. For example, if you wanted to partition the ClassEnrollment table based upon the range of values in the EnrollmentDate column, you could issue the following command:

```
CREATE TABLE ClassEnrollment (
  ClassID int NOT NULL ,
  StudentNumber int NOT NULL ,
  Status char (10) NOT NULL ,
  EnrollmentDate date NOT NULL ,
  Price number (9,2) NOT NULL ,
  Grade char (4) NULL ,
  Comments varchar2 (2000) NULL
) PARTITION BY RANGE (EnrollmentDate)
( PARTITION Pre_2001 VALUES LESS THAN ('01-JAN-01')
  TABLESPACE OLD_DATA,
  PARTITION Q1_2001 VALUES LESS THAN ('01-APR-01')
  TABLESPACE Q1_2001_DATA,
  PARTITION Q2_2001 VALUES LESS THAN ('01-JUL-01')
  TABLESPACE Q2_2001_DATA,
  PARTITION Q3_2001 VALUES LESS THAN ('01-OCT-01')
  TABLESPACE Q3_2001_DATA,
  PARTITION Q4_2001 VALUES LESS THAN ('01-JAN-02')
  TABLESPACE Q4_2001_DATA);
```

The PARTITION BY RANGE keyword tells Oracle that the column or columns enclosed in parentheses will be used to determine which partition data will be placed—that is, you will be able to predict where each row will be physically placed based upon the value of the EnrollmentDate column.

The VALUES LESS THAN clauses tell Oracle where to place rows whose data is below a threshold value. It is important to note that all data whose values are less than but not equal to the value specified will be stored in the partition. In the preceding

example, it would also not be possible to store any rows where the EnrollmentDate was greater than December 31, 2001, since the upper boundry of the last partition is VALUES LESS THAN '01-JAN-02'. If you did not want to limit the upper boundary of the data in the table, you can also make use of the MAXVALUE keyword, which will allow any valid data of the same datatype as the partition key to be stored.

Hash partitions

While range partitions allow a DBA to have predictability over where a specific row is stored, *hash partitions* are designed primarily for speed. In Oracle, retrieving rows by a hash value of one or more of the columns is the second fastest way, after using the ROWID, to get data. By partitioning a table according to a hash performed on one or more of its columns tells Oracle to manage the physical location of the data itself, while ensuring that the data will be stored in a manner to optimize data retrieval. Combining hash partitioning with parallel query operations can make a database perform much better than using parallel query alone.

To hash partition a table, you issue a command similar to the following:

```
CREATE TABLE ClassEnrollment (
  ClassID int NOT NULL ,
  StudentNumber int NOT NULL ,
  Status char (10) NOT NULL ,
  EnrollmentDate date NOT NULL,
  Price number (9,2) NOT NULL ,
  Grade char (4) NULL ,
  Comments varchar2 (2000) NULL
) PARTITION BY HASH (ClassID)
PARTITIONS 16 STORE IN (TS1, TS2, TS3, TS4);
```

This command partitions the ClassEnrollment table according to the hash value of the ClassID column. The table is broken up into 16 partitions, which will be stored in tablespaces TS1, TS2, TS3, and TS4. Each tablespace has four partitions so that no one tablespace is more burdened than another. The location of any one row cannot be predicted as Oracle will manage data placement.

Sub-partitioning (composite partitioning)

For the best of both worlds where the DBA can predict placement of rows, as with range partitions, and also ensure that performance is maximized, Oracle8i provides support for *composite partitions*, or *sub-partitioning*. With composite partitioning, you specify the main partitions by range and then sub-partitions by hash, as in the following example:

```
CREATE TABLE ClassEnrollment (
  ClassID int NOT NULL ,
  StudentNumber int NOT NULL ,
  Status char (10) NOT NULL ,
  EnrollmentDate date NOT NULL,
```

```

    Price number (9,2) NOT NULL ,
    Grade char (4) NULL ,
    Comments varchar2 (2000) NULL
) PARTITION BY RANGE (EnrollmentDate)
  SUBPARTITION BY HASH (ClassID) SUBPARTITIONS 4
  STORE IN (TS1, TS2, TS3, TS4)
( PARTITION Pre_2001 VALUES LESS THAN ('01-JAN-01'),
  PARTITION Q1_2001 VALUES LESS THAN ('01-APR-01'),
  PARTITION Q2_2001 VALUES LESS THAN ('01-JUL-01'),
  PARTITION Q3_2001 VALUES LESS THAN ('01-OCT-01'),
  PARTITION Q4_2001 VALUES LESS THAN ('01-JAN-02'));

```

In this example, you create a total of 20 sub-partitions. This is because you created five range partitions, as in the previous range partitioning example, and then you specified that you wanted four sub-partitions per partition with the `SUBPARTITIONS 4` clause. Each of the four tablespaces (TS1, TS2, TS3, and TS4) that you specified are used to store the hash-based sub-partitions and will contain five segments (one sub-partition for each range of values in the partition definition).

Rollback segment

One of the great advantages of using Oracle over other databases is the capability for one user to read data that someone else is changing. Naturally, the user reading the data does not see what the other user is changing, but rather gets a copy of the way the data looked prior to the start of the modifications. Support for allowing users to see data that is in the process of being changed is handled by rollback segments.

Whenever a transaction is started, Oracle makes a copy of the data to be changed in the rollback segment. The rollback segment records information about the way the data looked before the change took place. This allows other users reading the row in flux to be able to get a consistent image of the data and also allows the user making changes to the data to change his or her mind. If the user issues a `ROLLBACK` statement indicating that the changes should not take place, Oracle reconstructs the original data from the rollback segment. In this way, the rollback segment plays a very important part in maintaining the consistency of the database.



Rollback segments, their uses and their proper configuration are covered in more detail in Chapter 10.

Temporary segment

In any database, when users retrieve data, they may want to have that data sorted in a particular way. Oracle performs a sort whenever an `ORDER BY`, `GROUP BY`, `SELECT ... DISTINCT`, or `UNION`, `INTERSECT`, or `MINUS` clause is used in a SQL statement. Sorts may also take place during other operations, such as creating indexes. Oracle always attempts to perform the sort in memory, but if there is not sufficient memory to perform the entire sort, it has the sort spill to disk into the user's temporary tablespace. Temporary segments are created whenever a sort spills to disk. These segments are also sometimes referred to as *sort segments*.

Temporary segments are also used to store data in a temporary table. Because temporary table data only persists for the length of a transaction or a session, there is no point keeping it around. Consequently, it is stored in a temporary segment.

LOB segment

If a table contains columns of BLOB, CLOB, or NCLOB datatype (also known as large objects, or LOBs), Oracle stores the data for those columns in a separate segment away from the rest of the data. Typically, columns of these datatypes store data that is quite large, such as images, MP3 files, or even movies. The storage of data in a separate LOB segment occurs if the DBA has configured Oracle to store all LOB data in a separate segment. Oracle also stores LOB data in a separate segment if the data exceeds 4000 bytes, whether or not the DBA, or table owner, has told it to do so. This is done to ensure that large objects do not decrease performance on full table scans and data reads if the LOB columns are not being referenced.

LOB segments are also created for columns of VARRAY datatype. This is because VARRAYs can hold multiple elements and can be quite large. They are of an object datatype and, because of their potential size, are considered a potential LOB by Oracle.



More information on LOB datatypes and how LOBs can be used is provided in Chapter 11.

For each LOB segment created Oracle also creates a LOB index. The LOB index is always stored with the LOB segment and cannot be separated from it. The LOB index is used to look up values in the LOB segment to ensure that they are returned in the proper order and that Oracle is able to locate the data for any LOB column.

Nested table

In Oracle8 and Oracle8i, one of the objects that can be defined in a column of a table is another table, which is known as a *nested table*. A nested table is only associated with its parent table and cannot be addressed separately. However, because it is a table, Oracle creates a separate segment to store the nested table data.



More information on objects and collections, including nested tables and VARRAYs, is provided in Chapter 11.

Bootstrap segment

Whenever you boot your computer, there is something called a boot loader that determines where the operating system is and then reads that portion to start the OS and get your computer up and running. Oracle's *bootstrap segment* serves a similar purpose in your database. In each database the bootstrap segment, which is always located in the SYSTEM tablespace, is created by the SQL.BSQ script that is

run when you issue the CREATE DATABASE command. It contains information and program units that help to initialize the data dictionary cache and place other information in the SGA when the instance is started.

As a DBA, there is nothing you need to do to manage the bootstrap segment, nor should you. This is an internal resource that manages itself.

Storage Clause Precedence

Oracle allows a great deal of flexibility when creating objects with regard to storage. In an Oracle database you are able to configure how much space each object should have allocated to it when it grows, how large the object can grow, and much more. However, Oracle does not force you to always specify these values and does provide defaults, based upon how the database and tablespaces are configured so far.

When specifying a storage clause, as you have already seen with tablespaces in the previous chapter, you are setting values for the following storage parameters:

- ♦ INITIAL — The size of the initial extent for a segment
- ♦ NEXT — The size of the second extent for a segment
- ♦ PCTINCREASE — The percentage by which all subsequent extents for a segment, after the second extent, should increase rounded up to the next multiple of DB_BLOCK_SIZE
- ♦ MINEXTENTS — The minimum number of extents to allocate to a segment when it is first created
- ♦ MAXEXTENTS — The maximum number of extents that the segment is allowed to occupy

Depending on the segment additional storage clauses may also be specified dealing with space utilization at the block level, and other factors. Figure 9-2 presents a diagram outlining how storage clause precedence works in Oracle8i.

In determining which storage clauses to accept, Oracle generally uses the rule that if a storage clause is specified at the segment level (that is, table, index, and so on), then this storage clause will apply. However, this is not always true. The rules of storage clause precedence are as follows:

- ♦ If a storage clause is specified at the segment level when it is being created, that storage clause has precedence and its settings apply, with the following exception.
- ♦ If the tablespace on which the segment is to be created has a MINIMUM EXTENT value specified, or if the tablespace is locally managed and has a UNIFORM SIZE specified, all extents for the segment to be created will use these values.

- ♦ If the storage clause is not specified at the time the segment is created, and a DEFAULT STORAGE clause was specified at the time the tablespace on which the segment will reside was created, then the tablespace's storage clause will be applied to the segment. In other words, if the table you want to create has no storage clause, the tablespace's DEFAULT STORAGE clause values become the table's storage clause values.
- ♦ If a DEFAULT STORAGE clause was not specified when the tablespace was created, the tablespace will be assigned the Oracle's system default storage clause. Because this storage clause is applied to the tablespace, it will also apply to any segments created on the tablespace that do not have a storage clause, as outlined in the previous bullet point.

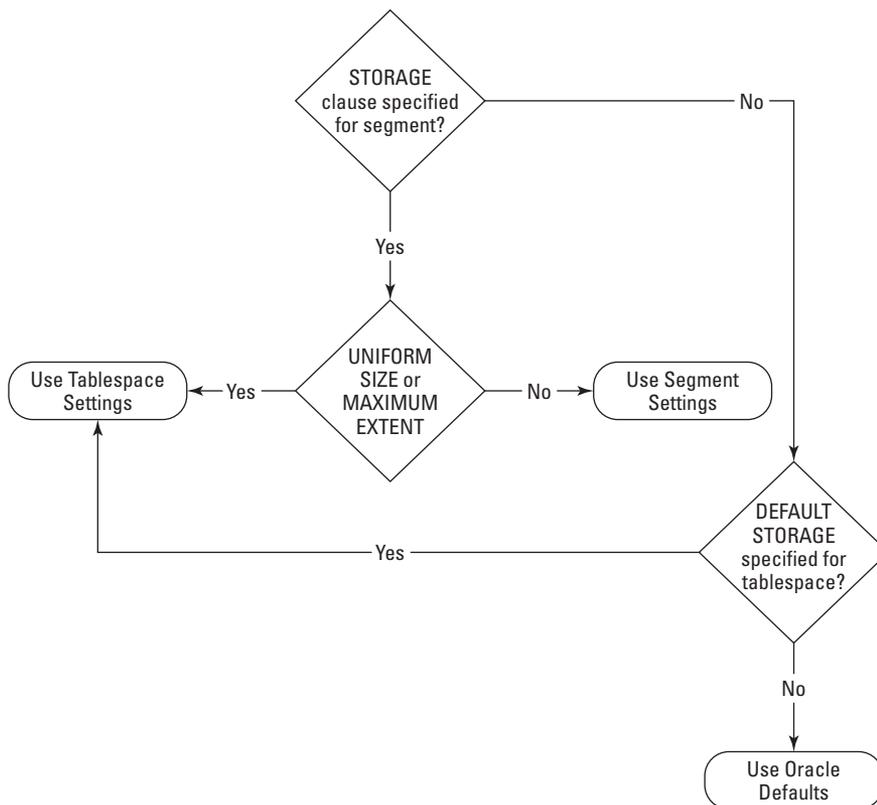


Figure 9-2: Storage clause precedence in Oracle8i

As a general rule of thumb, it is not a good idea to allow Oracle system defaults to be applied to a tablespace. You should always specify a `DEFAULT STORAGE` clause for any tablespace, and, furthermore, specify a storage clause for any segment you create. The reason for this is that the Oracle system defaults for storage, as shown in Table 9-1, are set at an arbitrarily small level for backward compatibility and usually do not allow for good space management in the database. In fact, using Oracle default may eventually cause a segment to report that it cannot allocate another extent (that is, it cannot grow any more), as well as cause excessive disk IO on the database because extent sizes will be too small.

Table 9-1
Oracle System Defaults for the Storage Clause

<i>Parameter</i>	<i>Value</i>
INITIAL	5 database blocks. For a 2KB block size, this means the initial extent will be 10KB; for a 4KB block size, it will be 20KB, and so on.
NEXT	5 database blocks
PCTINCREASE	50. This means that each extent, after the INITIAL and NEXT have been allocated, will be 50 percent larger than the previous extent, rounded up to the next multiple of <code>DB_BLOCK_SIZE</code> .
MINEXTENTS	1 for all segments except rollback segments; 2 for rollback segments.
MAXEXTENTS	Depends upon the value for <code>DB_BLOCK_SIZE</code> , as follows: 2048 = MAXEXTENTS of 121 4096 = MAXEXTENTS of 249 8192 = MAXEXTENTS of 505 16384 = MAXEXTENTS of 1017 32768 = MAXEXTENTS of 2041

Just because you specified a storage clause when you created a segment, or failed to do so, does not mean that you cannot make changes to the way that Oracle allocates space to that segment. Oracle allows you to alter a segment or tablespace and modify most of the values specified for storage attributes (with the exception of `INITIAL` for any segment). Any changes to the storage clause of a segment will apply to all future extents that are created but will not modify the currently allocated extents for the segment. In other words, Oracle does not reorganize your data to satisfy the new storage parameters but rather ensures that they are enforced in the future, while leaving your existing data intact.

As it is possible to specify certain parameters at the tablespace level—such as `MINIMUM EXTENT`—that will override segment level settings, the new values specified apply to all segments on the tablespace and any segments created on the tablespace in the future. You would typically specify tablespace-level settings such as `MINIMUM EXTENT` and `UNIFORM SIZE`, to ensure that segment growth takes place in a uniform fashion allowing more efficient space utilization.



For more information on specifying and altering tablespace storage parameters and their impact, refer to Chapter 8.

Extents

Whenever you create a tablespace, in each of the files that belong to the tablespace Oracle allocates a header block—to store information about the status of the file and the last time it was updated, among other information—and a single free extent for all of the remaining space in the datafile. An *extent* is one or more contiguous database blocks in a data file.

When you create segments, you are allocating space from the free extents of a datafile. Once the space has been allocated to a segment, the extent becomes a used extent and the space within the extent is managed by Oracle as the data for the segment grows. If you DROP or TRUNCATE a segment, the extents that were allocated for it in the datafile are added to the pool of free extents. A single datafile may have many used extents and many free extents at the same time. Frequent allocation and de-allocation of extents can cause fragmentation in the datafile and lead to small free extents that cannot be used by any segments—a kind of Swiss cheese effect.

Automatic and manual allocation of extents

Oracle automatically allocates extents to segments when the data for that segment grows and cannot fit into the existing extents—that is, there is no more room for the data in the existing extent. In order for the new extent to be allocated, Oracle locks the segment while allocating the new extent. While the segment is locked as extent allocation takes place, no users can access its data for read or write operations until the extent allocation completes. This process of allocating extents automatically can have negative performance consequences and should generally be avoided.

As you have read in Chapter 8, Oracle supports two kinds of tablespaces—dictionary-managed or locally managed. Locally managed tablespaces keep track of used blocks within each datafile while dictionary-managed tablespaces need to update data dictionary tables to indicate which blocks are used. If you are using dictionary-managed tablespaces, the time it takes to update the data dictionary will be the duration that the segment is unavailable to users. This interval is generally much greater than the time it takes to update the bitmap of a datafile and only a small subset of the tables in the data dictionary to reflect the additional extent for the segment. For this reason, with the exception of the SYSTEM tablespace, all tablespaces should be created as locally managed. This prevents a potential point of contention from materializing and helps keep your database running well. Furthermore, when you create a locally managed tablespace, you can also specify a uniform size for all extents thereby ensuring that extents that are de-allocated will be reused and fragmentation of the tablespace is not likely to occur.

To further ensure that you have good performance of the database, you can minimize the dynamic allocation of extents by allocating them manually as the free space for a segment's data decreases. The manual allocation of extents requires more work on the part of the DBA since you need to monitor how much free space is left in the segment's used extents and, once it falls below an acceptable threshold, issue a command to alter the segment and allocate an extent. Extents can even be allocated in a specific datafile belonging to the tablespace where the segment resides.

For example, if the Orders table were on the CustomerService tablespace, which was composed of two files (C:\CUSTSRV\DATA01.DBF and C:\CUSTSRV\DATA02.DBF), you would issue the following ALTER TABLE command to allocate an extent on the second datafile for the table:

```
SQL> ALTER TABLE Orders ALLOCATE EXTENT
      2 (SIZE 2M DATAFILE 'C:\CUSTSRV\DATA02.DBF');
```

Extend allocated.

SQL>



How to manually allocate extents for each specific segment is discussed in greater detail in the chapter devoted to the specific segment. Chapters 10, 11, and 12 deal with rollback segments, tables, and indexes, respectively, and include a section on extent allocation and de-allocation, both dynamic and manual.

While manual extent allocation may seem like a lot more work, it enables you to allocate extents of a particular size and on a specific datafile. Furthermore, you are able to allocate extents to segments during off-peak hours when the impact of extent locking during the allocation process can be minimized.

Block Space Utilization

The smallest unit of storage in Oracle is the database block. A database block can be anywhere from 2KB to 32KB (64KB for a very small number of platforms) and is comprised of one or more operating system blocks — usually more. The size of each database block in all tablespaces and datafiles is set at database creation time by the value of the DB_BLOCK_SIZE initialization parameter. The size of the database block cannot be changed after the database is created — if you need to do so, you will need to re-create the database after having exported all of the data.

Database block contents

Each database block in Oracle is made up of three parts, as shown in Figure 9-3. The first part of the database block is the block header, which consists of the database block address, a list of tables stored on the block (data from more than

one table can be stored on the block if the table was created on a cluster), a list of rows stored on the block, and transaction slots that indicate which transactions are active on which rows on the block. The block header is the top part of the block and grows from the top down, if necessary.

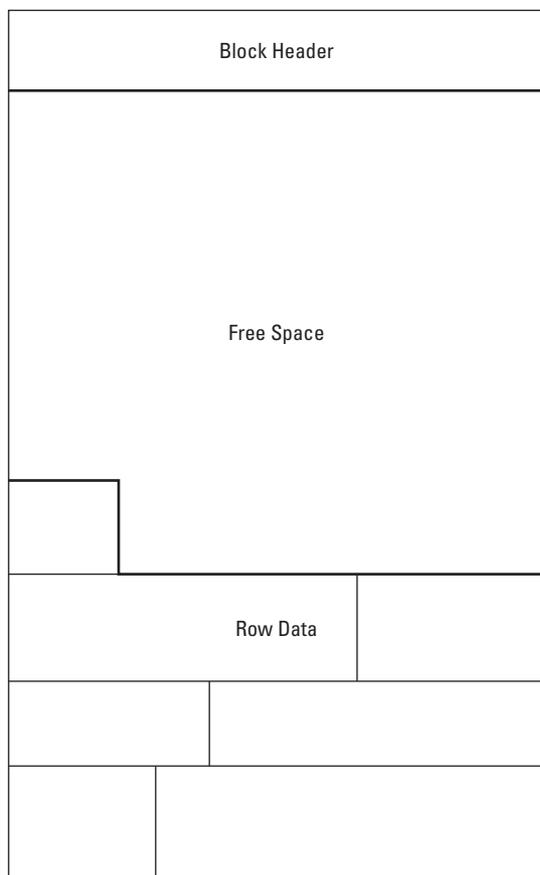


Figure 9-3: Oracle8i database block structure

A database block also contains row data or other entries, depending on the segment, that are stored starting at the bottom of the block and grow from the bottom up. This leaves a whole middle section, which is the free space on the block. The reason that free space is in the middle of the block is that both the block header (if more transaction slots are required) or segment data may need to grow. If the segment data were placed immediately after the block header, it would not be possible to allocate additional transaction slots, which may prevent multiple simultaneous updates to the block—a performance hit.

Because the free space of a block is in the middle of the block, initially it is contiguous. However, if rows are added and then removed from the block, it is possible for the free space to become fragmented. When Oracle needs to place additional data in the block, it will coalesce the free space and reorganize the data on the block so that the free space becomes contiguous. If it is unable to do so, it may move part or all of the data for a row to another block and leave a pointer behind indicating to which other block the data was moved. This is known as *row chaining* and should be minimized because of the negative impact on performance, although it cannot be completely avoided. Proper configuration of database block utilization parameters at the segment level helps to minimize this.



Although it is possible to manually coalesce free space within a tablespace by using the ALTER TABLESPACE command, you cannot force Oracle to coalesce free space inside a block. This process only happens automatically and when needed.

Block space utilization parameters

Up to this point you have learned that you can control and configure the allocation of extents for each segment by specifying storage parameters at the tablespace or segment level. Dynamic allocation of extents can be avoided and this will reduce fragmentation at the tablespace level. Oracle also allows you to minimize row chaining and ensure efficient utilization of database blocks for table and index segments (including partitions) by configuring the INITRANS, MAXTRANS, PCTFREE, and PCTUSED *block space utilization parameters*.

Block space utilization parameters must be specified in the STORAGE clause of a segment and cannot be specified in the DEFAULT STORAGE clause of a tablespace. This is because each table or index has different characteristics and, unless your tablespace contains only one table or index, it does not make sense to have the same settings for all segments on the tablespace. You should determine the optimal setting of these parameters before creating the segment.



More information on how to properly set block space utilization parameters can be found in chapters 11 and 12.

INITRANS

The INITRANS parameter for an index or table segment specifies the initial number of transaction slots to allocate in each block header for the segment. The default value is 1 for a data segment (table, partition, index, and so on) and 2 for a rollback segment. Each transaction that needs to modify data on the block requires a transaction slot, regardless of the number of rows being changed.

The ideal setting for INITRANS is dependent on the average number of rows that are stored on a block and the volatility of the data in the table. If you have a small block size and only three or four rows of data fit on a block, you may be able to live with the default if the data does not change that frequently. On the other hand if you

have a large block size and relatively small rows in comparison to the row size, you may need to specify a higher value for `INITRANS` when you create the segment. The value of `INITRANS` cannot be modified after the segment is created.

MAXTRANS

If more transactions need to perform changes to the block than there are available transaction slots as configured using the `INITRANS` parameter, Oracle allocates additional transaction slots from the free space up to a defined limit set by the value of the `MAXTRANS` block space utilization parameter. The default and maximum value for `MAXTRANS` is 255. In fact, the actual real maximum value for `MAXTRANS` is the physical number of rows that you can fit into a database block since any transaction making changes to data acquires an exclusive lock on the row prohibiting anyone else from making changes to the same block.

In most cases you can leave `MAXTRANS` at the default value because Oracle allocates additional transaction slots from the free space in the block as needed. However, if you wanted to ensure that no more than a specific number of transaction slots were ever allocated on a block, you can specify a value other than the default. For example, setting `MAXTRANS` to 10 would not allow any more than ten transactions to modify data on the block. If more than ten transactions are needed to modify data on the block, the eleventh would need to wait until one of the other transactions completed. Similarly, if there were not sufficient free space in the block to allocate additional transaction slots, any subsequent transaction would also need for a previous one to finish before it could go ahead. For these reasons, as well as to deal with variable character columns, it is important that sufficient free space remains on the block.

PCTFREE

In many cases you may create tables that have columns defined using the `VARCHAR2` or `NVARCHAR2` datatype. These columns, unlike `CHAR` or `NCHAR` that always occupy the amount of space on a block specified when the table was created, only require as much space as is needed to store the data. If you make changes to the data using an `UPDATE` statement, Oracle increases the size of the row and takes some of the space in the free space of the block for the new value. If there is insufficient space left on the block, Oracle chains the row.

The `PCTFREE` block space utilization parameter enables you to define a certain percentage of the space in the block to be left empty so that updates to variable-length columns does not cause a row to be chained. The default value for `PCTFREE` is 10, that is, Oracle will leave ten percent of the block space, minus the header, free for future updates. Essentially, what Oracle does is remove the block from the *freelist* (a list of blocks that can be used for `INSERT` operations).

`PCTFREE` can be specified for either a table segment or an index segment. If specified for a table segment, Oracle monitors the amount of free space on each block belonging to the table and removes the block from the freelist if the amount of free space falls below `PCTFREE`. However, if the `PCTFREE` parameter is specified for an

index, Oracle only uses this value when creating the index to determine how much free space to leave on each index leaf block for new entries. The value for PCTFREE is not monitored or maintained after the index is created. This is one of the reasons that you will need to periodically rebuild indexes in order to ensure good performance and proper utilization of block space.

PCTUSED

Once a block has been taken off the freelist because the amount of free space in the block has fallen below the value configured by the PCTFREE block space utilization parameter, the block will not be placed back on a freelist unless the percentage of used space (that is, space with row data) falls below the value specified by the PCTUSED parameter.

The default value for PCTUSED is 40, which means that if the amount of used space on a block falls below 40 percent of the size of the block, less the header, then Oracle places the block on the freelist and allows INSERT operations to occur once again. Any block placed on the freelist as a result of PCTUSED being crossed goes to the top of the freelist and becomes the first one to which INSERT operations will be performed. In this way, Oracle ensures that all of the currently allocated extents for a segment are full of data before allocating new extents.

Getting Information about Database Storage Structures

The primary way you have in Oracle to determine how much space is available or used in your datafiles is through data dictionary views. Oracle Enterprise Manager allows you to find out which tablespaces and datafiles are in your database and the configuration of storage parameters for an object, but OEM does not provide a way to see which segments have storage allocated on which data files or how much storage is allocated.



The capabilities of Oracle Enterprise Manager and how it can be used to configure storage parameters for a segment is covered in chapters 10, 11, and 12 for rollback segments, tables, and indexes, respectively.

Oracle provides a number of DBA_ views for determining the allocation of storage space within the database. Table 9-2 lists the data dictionary views that can be queried to get storage-related information from the data dictionary.

Freelists

For each table that you create in your database, Oracle allocates one freelist to hold a list of blocks into which INSERT operations can take place. In order for a transaction to INSERT a new row in a block, it places a lock on the freelist, retrieves the location of the first block on the list, performs the INSERT on the block, and then releases the freelist. On a table that could be updated by a great many users simultaneously, a single freelist could become a potential point of contention. For this reason, you should allocate as many freelists when you create a table as the number of simultaneous INSERT operations you expect to take place. In this way, all of the blocks for the table that can accept INSERTs will be spread out among the freelists available. A single block never appears on more than one freelist. However, having more than one freelist for a table enables more than one user to perform an INSERT at the same time, thereby increasing throughput.

For more information on freelists and their performance impact, refer to the *Oracle8i Designing and Tuning for Performance* manual in the Oracle documentation set.

Table 9-2
Storage-Related Data Dictionary Views

<i>View</i>	<i>Description</i>
DBA_TABLESPACES	Contains information about the tablespaces in the database and the values for the DEFAULT STORAGE clause specified at the time the tablespace was created. If no DEFAULT STORAGE clause was specified, the INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, and PCT_INCREASE columns will have Oracle default values.
DBA_DATA_FILES	Contains information about the datafiles in the database and their status. The TABLESPACE_NAME column includes the name of the tablespace to which the file belongs. Other columns provide information on the size of the file in database blocks and bytes, as well as the amount of space in the file that is used by extents.
DBA_SEGMENTS	Has information about each segment created in the database including the name, type, and owner of the segment, as well as the tablespace in which it is stored. You also have information about the storage characteristics of the segment including the values for INITIAL, NEXT, MINEXTENT, MAXEXTENT, and PCTINCREASE. The view also provides information on the location of the header block for the segment (the block that contains information about the segment structure, and so forth) as well as the number of extents, blocks, and bytes currently in use by each segment in the database.

Continued

Table 9-2 (continued)

<i>View</i>	<i>Description</i>
DBA_EXTENTS	The DBA_EXTENTS view provides details about each extent allocated to a segment, including the size of the extent in blocks and bytes, its physical location (including tablespace, datafile number and block number), as well as the extent id for each extent.
DBA_FREE_SPACE	This view allows you to query which tablespaces and datafiles have free space and the size of each free extent in blocks and bytes. This can be useful if you want to create additional segments or allocate an extent manually for a segment and the tablespace has more than one datafile.

If you want to find the tablespaces and segment names for all segments owned by the user Student, you would issue the following command:

```
SQL> col tablespace_name format a30
SQL> col segment_name format a30
SQL> SELECT segment_name, tablespace_name
       2 FROM DBA_SEGMENTS
       3 WHERE owner='STUDENT'
       4 ORDER BY segment_name;
```

SEGMENT_NAME	TABLESPACE_NAME
-----	-----
BATCHJOBS	CERTDB
BATCHJOBS_JOBID_PK	CERTDB
CLASSENROLLMENT	CERTDB
COURSEAUDIT	CERTDB
COURSEAUDIT_PK	CERTDB
COURSES	CERTDB
INSTRUCTORS	CERTDB
LOCATIONS	CERTDB
PK_CLASSID	CERTDB
PK_CLASSID_STUDENTNUMBER	CERTDB
PK_COURSENUMBER	CERTDB
SEGMENT_NAME	TABLESPACE_NAME
-----	-----
PK_INSTRUCTORID	CERTDB
PK_LOCATIONID	CERTDB
PK_STUDENTNUMBER	CERTDB
SCHEDULEDCLASSES	CERTDB
STUDENTS	CERTDB

16 rows selected.

SQL>

To get a list of extents that belong to the ClassEnrollment table, the datafile each is located in, and the size of each extent size, you would query the DBA_EXTENTS view as follows:

```
SQL> col file_name format a28
SQL> SELECT e.extent_id, f.file_name, e.block_id, e.blocks, e.bytes
  2 FROM DBA_EXTENTS e, DBA_DATA_FILES f
  3 WHERE e.file_id = f.file_id
  4 AND e.segment_name='CLASSENROLLMENT';
```

EXTENT_ID	FILE_NAME	BLOCK_ID	BLOCKS	BYTES
0	F:\CERTDB\DISK2\CERTDB01.DBF	52	5	40960

```
SQL>
```

To find out how much free space there is available on the CERTDB tablespace, you can issue the following command:

```
SQL> SELECT tablespace_name, block_id, blocks, bytes
  2 FROM DBA_FREE_SPACE
  3 WHERE tablespace_name = 'CERTDB';
```

TABLESPACE_NAME	BLOCK_ID	BLOCKS	BYTES
CERTDB	82	1199	9822208

```
SQL>
```

Planning the Location of Segments

As a DBA, one of your duties is to ensure that segments are properly placed on tablespaces and that a sufficient number of tablespaces with proper DEFAULT storage clauses exist. The DEFAULT STORAGE clause for each tablespace should be configured to ensure that the segments created on the tablespace inherit appropriate storage parameters.

If you use the Database Configuration Assistant, it will automatically create an Optimal Flexible Architecture–compliant database with six tablespaces. This is a good practice to follow because each tablespace should contain data of a similar type. In other words, it is not a good idea to store table segments and index segments on the same tablespace. When you create a database, you should create one tablespace of each of the following type to hold your data:

- ♦ **SYSTEM**—The SYSTEM tablespace is automatically created with the CREATE DATABASE command. It should not hold any other segments except the data dictionary and other system segments, such as the bootstrap segment and the system rollback segment.

- ♦ **DATA**— You will create one or more tablespaces to hold table data. These same tablespaces may also be used for clusters and partitioned tables. It is likely that you will create more than one data tablespace as you should keep large tables apart from small tables, as well as those heavily accessed apart from other tables that are also heavily accessed. More on this later.
- ♦ **INDEX**— As mentioned earlier, you should keep index segments on separate tablespaces from data segments. In fact, index tablespaces should have datafiles on separate physical disks from the data tablespace datafiles. In this way, you will have more efficient use of disk drives and less chance of the disk becoming a bottleneck.
- ♦ **ROLLBACK**— Rollback segments in an OLTP environment have a lot of activity. For this reason you should have one or more rollback segment tablespaces whose data files are away from the index and data tablespaces' datafiles.
- ♦ **TEMP**— Temporary segments created when a sort spills to disk, or a temporary table is populated with data, should be created on a tablespace that has been designated as temporary (that is, created with the `CREATE TEMPORARY TABLESPACE` command). If temporary segments are created on a tablespace designed to hold permanent segments (tables, indexes, and so on), it can potentially cause a lot of fragmentation because they will be created and dropped fairly regularly. Temporary tablespaces should be stored on disk away from `DATA`, `INDEX`, and `ROLLBACK` segments tablespaces.
- ♦ **TOOLS**— Many applications that run on Oracle need to create segments to support their operation. These segments rarely change after creation and are quite stable. A separate tablespace should be created to hold these segments but, because of the stability of the segments on it, the tablespace's datafiles can be created on the same disk as another tablespace, including `SYSTEM`.

Different tablespaces, due to the nature of the segments created on them, have different fragmentation characteristics, as shown in Table 9-3. Fragmentation occurs when a segment is dropped or truncated and storage is released back to the tablespace. Because this can have a negative effect on performance, it is generally recommended that tablespaces with high fragmentation propensity be stored on separate physical disks from other tablespaces of the same propensity. Also, segments with high fragmentation propensity should be on tablespaces away from those with a low fragmentation propensity.

Table 9-3
Fragmentation Propensity of Tablespace Types

<i>Tablespace Type</i>	<i>Contents</i>	<i>Fragmentation Propensity</i>
SYSTEM	Data Dictionary	Zero, assuming no other segments are created on it
DATA	Data segments	(table, cluster, and so on)
INDEX	Index segments (index, IOT, and so on)	Low in most cases
ROLLBACK	Rollback segments	High
TEMP	Temporary segments	Very High
TOOLS	Application support segments	Very Low

While fragmentation is an issue that needs to be dealt with in almost every database, as stated in Chapter 8, many of these issues can be at least minimized by the use of locally managed tablespaces with a UNIFORM SIZE configured or by using temporary tablespaces for sort segments and GLOBAL TEMPORARY tables. However, other issues require manual intervention by the DBA, as well as good planning to ensure acceptable performance of the database. These include:

Segment Size — In most databases you have tables and indexes that are normally large (detail tables in OLTP systems or fact tables in data warehouses) or small (lookup tables in OLTP systems or dimension tables in data warehouses). It is recommended that large segments be stored in their own tablespaces away from other segments. Small segments should be stored together, provided their access patterns are similar to the other small segments in the same tablespace.

Segment Lifetime — Some segments are continually added to and never shrink, whereas others may be truncated or rebuilt on a regular basis. Store any segments that have a short lifespan away from any segments with a longer lifespan. The frequent truncation of the segment increases fragmentation, which you don't want in a tablespace holding a segment that continuously grows.

Backup and Recovery — Segments that need to be backed up together should be stored on the same tablespace so that you can perform tablespace backups. This practice also allows you to restore them as a group, should the datafile become corrupted or something go wrong.

Disk Configuration — Having multiple physical disks available enables you to better distribute the data across multiple physical devices and thereby reduce the likelihood of a bottleneck. The same benefit is derived from a striped disk array.

Each database and environment may have additional requirements that need to be satisfied. When planning storage of segments in Oracle, ensure that your business needs are satisfied.

Key Point Summary

In preparing for the “Oracle®i DBA: Architecture and Administration” exam, please keep these points regarding storage structure and relationships in mind:

- ♦ The logical structure of the database controls how space is allocated to segments and how segments grow.
- ♦ Oracle supports a great many segment types including tables, indexes, partitions, index-organized tables, clusters, nested tables, LOB segments, LOB indexes, rollback segments, temporary or sort segments, and the bootstrap segment.
- ♦ In most cases, if a storage clause is specified at the segment level, its settings apply to the segment.
- ♦ The only exceptions to segment storage settings taking precedence are if a UNIFORM SIZE was specified for a locally managed tablespace, or a MAXIMUM EXTENT was specified for a dictionary-managed tablespace; then these settings override segment level settings.
- ♦ The tablespace’s DEFAULT STORAGE clause applies to any object for which a storage clause was not specified.
- ♦ Oracle storage defaults apply to all tablespaces for which a DEFAULT STORAGE clause was not specified at creation time.
- ♦ Extents are groups of contiguous database blocks allocated to a segment at creation time, or as it grows.
- ♦ Extents can be allocated dynamically as required by Oracle, or manually by the DBA. Dynamic allocation may cause performance problems if the storage clause was not properly configured for the segment. Frequent deallocation of extents by using the TRUNCATE command can also have negative performance effects.
- ♦ Extents are de-allocated when a segment is dropped or truncated. A DBA can also issue an ALTER command to de-allocate any unused space for a segment.
- ♦ INITIAL, NEXT MINEXTENT, MAXEXTENT storage clause parameters specify the initial and incremental size of extents for a segment, as well as the number to allocate at creation time and the maximum that may be allocated to a segment. If a segment reaches MAXEXTENT, it cannot grow any more until the value is adjusted.

- ♦ Oracle enables you to control space utilization at the database block level through the use of the INITRANS, MAXTRANS, PCTFREE, and PCTUSED parameters for index and table segments. INITRANS and MAXTRANS establish how many transactions are able to operate on a block at the same time, while PCTFREE and PCTUSED determine whether or not a block should appear on the freelist for the segment.
- ♦ The DBA_SEGMENTS, DBA_EXTENTS, DBA_FREE_SPACE, as well as DBA_TABLESPACES and DBA_DATA_FILES data dictionary views enable you to extract information from the data dictionary pertaining to space used by segments, as well as information about available free space.
- ♦ Fragmentation propensity needs to be carefully considered when planning the number and location of tablespaces and datafiles.
- ♦ Each database should have at least one tablespace of DATA, INDEX, TEMP, ROLLBACK, TOOLS, and SYSTEM type to store its appropriate segments, respectively. In many cases, because of fragmentation, segment size, segment lifespan, and other issues, you need to create multiple tablespaces of a particular type—especially DATA and INDEX.



STUDY GUIDE

Now that you have a better idea of how storage is logically organized in an Oracle database, test your knowledge by answering the assessment questions and scenarios. Use the labs to become familiar with performing typical DBA tasks related to storage structures.

Assessment Questions

1. What storage parameter would you specify for a table if you wanted to ensure that at least three concurrent transactions would be allowed on each block? (Choose the best answer.)
 - A. PCTFREE 3
 - B. MAXEXTENTS 3
 - C. MAXTRANS 3
 - D. INITRANS 3
 - E. TRANSCOUNT 3
2. What is the minimum number of extents that must be specified for a rollback segment? (Choose the best answer.)
 - A. 1
 - B. 2
 - C. 5
 - D. 0
 - E. You cannot specify a minimum number of extents for a rollback segment.
3. Which of the following block space utilization parameters is not valid for an index segment? (Choose the best answer.)
 - A. PCTFREE
 - B. PCTUSED
 - C. INITRANS
 - D. MAXTRANS
 - E. MINIMUM EXTENT

4. You create a table called Customers on a tablespace called CustData. You do not specify a STORAGE clause for the table. A DEFAULT STORAGE clause was not specified when the tablespace was created. Your DB_BLOCK_SIZE parameter is set to 4096. What will be the size of the second extent allocated to the Customers table if Oracle allocates it dynamically? (Choose the best answer.)
- A. 10240 bytes
 - B. 4096 bytes
 - C. 20480 bytes
 - D. 30720 bytes
 - E. The size of the second extent cannot be predicted.
5. Which of the following segments must reside on the same tablespace? (Choose two correct answers.)
- A. Table
 - B. LOB segment
 - C. Index
 - D. Cluster
 - E. LOB index
 - F. Nested Table
6. Which type of tablespace should you use to minimize fragmentation of storage space and ensure that extents for segments are of a particular size? (Choose the best answer.)
- A. Permanent
 - B. Dictionary-managed
 - B. Temporary
 - D. Locally managed
 - E. SYSTEM
7. Which data dictionary views would you need to query to determine the name of the datafiles where extents for the Customers table are located? (Choose all correct answers.)
- A. DBA_TABLESPACES
 - B. DBA_SEGMENTS
 - C. DBA_DATA_FILES
 - D. DBA_EXTENTS
 - E. DBA_FREE_SPACE

8. Which information is contained in the header of a database block? (Choose all correct answers.)
- A. Extent Number
 - B. Table directory
 - C. Transaction slots
 - D. Row directory
 - E. Corruption flag
9. What happens to a database block of a table partition when the amount of used space falls below the value specified by PCTUSED? (Choose the best answer.)
- A. The block is placed on the freelist.
 - B. The block is removed from the freelist.
 - C. The data on the block is moved to another block and the block is de-allocated.
 - D. Row migration is eliminated.
 - E. You cannot specify PCTUSED for table partitions.
10. How many extents exist in a tablespace that was just created? (Choose the best answer.)
- A. 0
 - B. 1
 - C. 1 per datafile
 - D. 1 in the first datafile and 0 in all other datafiles
 - E. You cannot determine the number of extents in a newly created tablespace.

Scenarios

1. You are the DBA of a large Internet retailer. You have been advised that a new initiative has been started to take existing data from the Sales database and create a SalesAnalysis database that will be used to perform analytical processing. You have the following information so far:
- A table will be created for each fiscal year's sales data.
 - Dimension tables describing attributes in the fact tables will be the same for all years.
 - It is expected that users will need to create temporary tables for messaging data.

- Indexes will be rebuilt on current year's data on a regular basis. Indexes on previous year's data will be created and then left alone.
- The data will be loaded into the database using scripts that will generate minimal rollback information.

How many tablespaces should you create to satisfy these requirements?

Will storage characteristics be specified at the tablespace or segment level?

How can you ensure good performance of this database?

2. You have been asked to provide consulting assistance to a textile manufacturer in the northeast United States. The company has indicated that they are finding performance to be decreasing on their database. Your initial review finds the following:

- The database has one tablespace — SYSTEM.
- There are four users who own objects in the database.
- The database consists of 300 tables, 650 indexes, and a handful of other segments, not including the data dictionary.
- Of the 300 tables, 50 are large and frequently accessed.
- The segments are not dropped or truncated and remain in the database after creation.
- 500 users connect to the database.
- The server on which the database resides has six additional hard disks that currently have no files on them.

What recommendations would you make to help alleviate some of the performance problems?

Lab Exercises

Lab 9-1 Creating Segments

In this lab you will run the scripts that are provided on the CD-ROM that came with this book. These scripts create a number of segments that will be used in the chapters that follow.

To run the scripts, perform the following tasks:

1. Logon to the computer that you will be running Oracle from as an administrator (preferred).

2. If you are running WindowsNT/2000/9x, create a folder off of the root of your C: drive called **CERTDB**. If you are running Linux or another UNIX variant, create a folder off the root called **CERTDB**. In the Linux/UNIX world, the folder name must be created in all uppercase letters.
3. Insert the CD into your CD-ROM drive and locate the **DBSETUP** folder. Copy the contents of the **DBSETUP** folder to the **CERTDB** folder you created.
4. Review the CREATEUSER.SQL script to ensure that the file that will be created for the CERTDB tablespace is valid for your system. For example, if you did not create the CERTDB folder on the C: drive but the D: drive, modify the filename to reflect this change.
5. Invoke Server Manager line mode from the command line and execute the scripts in the following order:
 - a. CREATEUSER.SQL—To create the user and tablespace.
 - b. CERTDBOBJ.SQL—To create the database objects.
 - c. INSERT_DATA.SQL—To load the sample data.

To execute the scripts, at the command prompt, issue the following commands:

```
C:\CERTDB> SET ORACLE_SID=CERTDB
C:\CERTDB> svrmgr1
```

```
Oracle Server Manager Release 3.1.7.0.0 - Production
```

```
Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.
```

```
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0-Production
```

```
SVRMGRL> @createuser.sql;
...
SVRMGRL> @certdbobj.sql;
...
SVRMGRL> @insert_data.sql;
...
SVRMGRL> quit
Server Manager Complete
```

```
C:\CERTDB>
```

Lab 9-2 Viewing Segment Information

In this lab you will query data dictionary views to determine the space allocation of segments and extents.

1. Query the data dictionary to determine the tablespace, number of extents, and size in bytes of all tables that are owned by the user Student. Repeat the same query for all indexes in the Student schema.
2. Query the data dictionary to determine if any segment in the database contains extents in more than one tablespace.
3. Determine which tablespace and datafile has the most free space.
4. How many free extents are there in the SYSTEM tablespace?

Answers to Chapter Questions

Chapter Pre-Test

1. If you specify a UNIFORM SIZE on a locally managed tablespace, any extent sizes specified in the storage clause of a segment, such as a table or index, is ignored. All extents for all segments created on the tablespace have extents of the size specified by UNIFORM SIZE.
2. To reduce the overhead of dynamically allocating extents, you should use locally managed tablespaces. This is because Oracle does not lock the segment when another extent needs to be allocated, thereby allowing users to continue to add data to the segment. If the tablespace on which the extent is to be allocated is dictionary-managed, Oracle would need to lock the segment until the extent was properly allocated, which would not allow any changes until the operation was completed.
3. The PCTUSED parameter determines how much of the database block can be filled with data before INSERT operations are once again allowed on the block. If the percentage of space used by data rows in the database block falls below PCTUSED, the block is placed on the freelist, thereby allowing INSERTS to once again take place on the block.
4. The PCTFREE parameter, when specified in the storage clause of an index segment, instructs Oracle to leave that percentage of the database block's space free for future index entries. The PCTFREE parameter only applies to the leaf level blocks and does not apply to the root or branch blocks of the index. Setting PCTFREE is useful if data with similar key values will be later added to the database and will minimize index block splits. However, PCTFREE is not maintained after the index is created and is used to initially build the index.
5. Oracle8i supports tables, indexes, table partitions and subpartitions, index partitions and subpartitions, index-organized tables, nested tables, index clusters, hash clusters, LOB segments, LOB indexes, rollback segments, temporary or sort segments, and the bootstrap segment.
6. If a storage clause was specified at the table level, its settings apply unless the table is created on a tablespace with a UNIFORM SIZE or MINIMUM EXTENT specified, in which case those settings override extent settings. If no storage

clause was specified when the table was created, the table inherits the tablespace settings specified in the `DEFAULT STORAGE` clause for the tablespace. If no `DEFAULT STORAGE` clause was specified when the tablespace was created, the tablespace inherits the Oracle default storage settings.

7. You can query the `DBA_SEGMENTS` data dictionary view and get information on the number of extents and total number of blocks allocated to a segment. You can also query the `DBA_EXTENTS` view to find information about the location of each extent of the segment.
8. A transaction slot is an area in a database block header that keeps track of which transaction is modifying which row located on the block. For each row being modified a transaction slot with the row information and the transaction identifier is created in the database block header. The number of transaction slots that each block of a segment will allocate space for is specified by the `INITRANS` parameter of the segment's storage clause. The default value is 1.
9. The `INITRANS` parameter specifies the initial number of transaction slots to allocate on each database block for a segment. If the number of rows being modified on a block exceeds the value of `INITRANS`, Oracle allocates additional transaction slots from the free space in the block up to the maximum allowed and specified by `MAXTRANS`. The default for `MAXTRANS` is 255.
10. When deciding which segments to place on which tablespaces, you should take into account the fragmentation propensity of the segments, the segments' lifespan, the access patterns of the data, the type of object, and other issues specific to the requirements of the organization. The goal is to ensure that those objects that are frequently reorganized — such as indexes — do not coexist with objects that rarely change — such as tables. Furthermore, you do not want to put large objects together on the same tablespace because of the greater likelihood of having full table scans take place, thereby creating a potential bottleneck. Those objects that will persist in the database should not be on the same tablespace with objects that have a short time span. Frequently accessed objects should be placed on separate tablespaces with data files on different disks. No objects, except the data dictionary objects, should be created on the `SYSTEM` tablespace. These are some of the considerations to keep in mind.

Assessment Questions

1. **D.** The `INITRANS` parameter controls the initial number of transaction slots to allocate in the block header of each block in the segment. Setting this value to 3 would ensure that each block would be able to support three concurrent transactions.
2. **B.** The minimum number of extents that must be specified for a rollback segment is 2 and hence `MINEXTENT` for a rollback segment defaults to 2. All other segment types require that only one extent be allocated at creation time.

3. **B.** The PCTUSED block space utilization parameter is not valid for an index segment. This is because you cannot control the allocation of data in an index because all entries must be stored in order. For this reason, Oracle must manage the allocation of space for index segments itself.
4. **C.** If you did not specify a storage clause when creating the table, or a DEFAULT STORAGE clause was not specified for the tablespace, then Oracle storage defaults apply. The Oracle default for the size of INITIAL or NEXT is 5 database blocks. If the value of DB_BLOCK_SIZE is 4096 bytes, the size of the NEXT extent will be 5*4096 or 20480 bytes.
5. **B., E.** The LOB segment and its corresponding LOB index segment cannot be stored on different tablespaces and always reside on the same tablespace. This is because the index and the LOB segment itself are used jointly when querying LOB data, which means that separating them would not allow the LOB data to be retrieved if a tablespace with the LOB index were not online.
6. **D.** You should use a locally managed tablespace to minimize fragmentation of storage space and ensure that extents for segments are of a particular size. Locally managed tablespaces always hold permanent objects and allow you to specify a UNIFORM SIZE for each extent in the tablespace.
7. **C., D.** To determine the name of the datafiles where extents for the Customers table are located, you query the DBA_DATA_FILES and DBA_EXTENTS data dictionary views. They both contain a FILE_ID column, but only DBA_DATA_FILES has the name of the datafile for the specific FILE_ID.
8. **B., C., D.** The header of a database block contains the data block address (its physical location), a list of tables on the block (table directory), a list of rows on the block (row directory), and transaction slots.
9. **A.** When the amount of used space of a database block of a table partition falls below the value specified by PCTUSED, the database block is placed on the freelist to allow further INSERTs into the block. Whether the database block belongs to a table segment that is not partitioned, or to a table partition segment, the rules are still the same — PCTUSED, when crossed, places blocks on the freelist.
10. **C.** In a newly created tablespace, you will find one free extent per datafile. If you only specified one datafile when you created the tablespace, you will have one free extent for the entire tablespace. However, since it is possible to create a tablespace and specify more than one datafile at creation time, it may be possible to have more than one free extent in the tablespace — one per datafile.

Scenarios

1. Because of the nature of the data, you should create one tablespace to hold one year's sales data. You should create another tablespace for all of the dimension tables. You should probably create at least one, and possibly more, temporary tablespaces that will be used for sorts that spill to disk and the

temporary tables used by the analysts. You will also need to create one rollback segment tablespace. You should create a tablespace for previous years' indexes and another tablespace for the index for this year's data. In total, you will need at least seven tablespaces (assuming two year's worth of data is kept in the database), and an additional tablespace for each year's sales data—at a minimum.

When specifying storage characteristics, you should always do so at the segment level. The only real exceptions here are the temporary tablespaces and the tablespaces holding the annual sales figures. Because each year's annual sales figures will be on a separate tablespace, you can specify the storage parameters when the tablespace is created. For temporary tablespaces, you should set them according to the amount of data that will be spilled to disk during sorts.

To ensure that the database performs well, place sales data and index tablespaces on separate hard disks. Place the dimension tables on different disks from the sales tables or indexes. Put rollback segments and temporary tablespaces away from the rest of the tablespaces, and ensure that all of your tablespaces are locally managed.

2. The first recommendation that you should make is that more tablespaces be created. Tables and indexes should be on different tablespaces, preferably on different hard disks. The large tables should be placed on their own tablespaces with tables with similar fragmentation propensities together. You should also strongly recommend that temporary tablespaces be created, as well as several tablespaces to hold rollback segments with datafiles on several hard disks. After the tablespaces are created, objects should be moved to the new tablespaces when the database is experiencing a period of low activity.

Lab Exercises

Lab 9-2

In this lab you will query data dictionary views to determine the space allocation of segments and extents.

1. Query the data dictionary to determine the tablespace, number of extents, and size in bytes of all tables that are owned by the user Student. Repeat the same query for all indexes in the Student schema.

```
SQL> col segment_name format a20
SQL> col tablespace_name format a20
SQL> SELECT segment_name, tablespace_name, bytes
2 FROM DBA_SEGMENTS
3 WHERE owner='STUDENT' AND segment_type = 'TABLE';
```

SEGMENT_NAME	TABLESPACE_NAME	BYTES
COURSES	CERTDB	40960
INSTRUCTORS	CERTDB	40960

```

LOCATIONS          CERTDB          40960
STUDENTS           CERTDB          40960
SCHEDULEDCLASSES  CERTDB          40960
CLASSENROLLMENT   CERTDB          40960
BATCHJOBS          CERTDB          40960
COURSEAUDIT        CERTDB          40960

```

8 rows selected.

```

SQL> SELECT segment_name, tablespace_name, bytes
       2 FROM DBA_SEGMENTS
       3 WHERE owner='STUDENT' AND segment_type = 'INDEX';

```

SEGMENT_NAME	TABLESPACE_NAME	BYTES
PK_COURSENUMBER	CERTDB	40960
PK_INSTRUCTORID	CERTDB	40960
PK_LOCATIONID	CERTDB	40960
PK_STUDENTNUMBER	CERTDB	40960
PK_CLASSID	CERTDB	40960
PK_CLASSID_STUDENTNU MBER	CERTDB	40960
BATCHJOBS_JOBID_PK	CERTDB	40960
COURSEAUDIT_PK	CERTDB	40960

8 rows selected.

SQL>

2. Query the data dictionary to determine if any segment in the database contains extents in more than one tablespace.

```

SQL> SELECT segment_name, owner,
       2 COUNT(DISTINCT tablespace_name)
       3 FROM DBA_EXTENTS
       4 GROUP BY segment_name, owner
       5 HAVING COUNT(DISTINCT tablespace_name) > 1;

```

no rows selected

SQL>

3. Determine which tablespace and datafile has the most free space.

```

SQL> SELECT tablespace_name, SUM(blocks), SUM(bytes)
       2 FROM DBA_FREE_SPACE
       3 GROUP BY tablespace_name
       4 ORDER BY SUM(blocks) DESC, SUM(bytes) DESC;

```

TABLESPACE_NAME	SUM(BLOCKS)	SUM(BYTES)
USERS	6399	52420608

```

TEMP                6287    51503104
INDX                 3199    26206208
TOOLS                3199    26206208
RBS                  2815    23060480
SYSTEM              1232    10092544
CERTDB              1199    9822208

```

7 rows selected.

```

SQL> col file_name format a35
SQL> SELECT file_name, SUM(f.blocks), SUM(f.bytes)
   2 FROM DBA_DATA_FILES d, DBA_FREE_SPACE f
   3 WHERE f.file_id=d.file_id
   4 GROUP BY file_name
   5 ORDER BY SUM(f.blocks) DESC, SUM(f.bytes) DESC;

```

FILE_NAME	SUM(F.BLOCKS)	SUM(F.BYTES)
F:\CERTDB\DISK3\USERS01.DBF	6399	52420608
F:\CERTDB\DISK6\TEMP01.DBF	6287	51503104
F:\CERTDB\DISK2\TOOLS01.DBF	3199	26206208
F:\CERTDB\DISK5\INDX01.DBF	3199	26206208
F:\CERTDB\DISK4\RBS01.DBF	2815	23060480
F:\CERTDB\DISK1\SYSTEM01.DBF	1232	10092544
F:\CERTDB\DISK2\CERTDB01.DBF	1199	9822208

7 rows selected.

SQL>

4. How many free extents are there in the SYSTEM tablespace?

```

SQL> SELECT tablespace_name, COUNT(*)
   2 FROM DBA_FREE_SPACE
   3 WHERE tablespace_name = 'SYSTEM'
   4 GROUP BY tablespace_name;

```

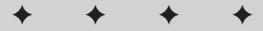
TABLESPACE_NAME	COUNT(*)
SYSTEM	11

SQL>

Managing Rollback Segments

EXAM OBJECTIVES

- ◆ Managing Rollback Segments
 - Create rollback segments using appropriate storage settings
 - Maintain rollback segments
 - Plan the number and size of rollback segments
 - Obtain rollback segment information from the data dictionary
 - Troubleshoot common rollback segment problems



CHAPTER PRE-TEST

1. What are the three main uses of rollback segments?
2. What information is kept in rollback segments?
3. What causes the ORA-01555 SNAPSHOT TOO OLD error?
4. Why do you need multiple rollback segments?
5. What is the difference between an online and offline rollback segment?
6. What is the significance of the OPTIMAL parameter?
7. What data dictionary view contains a row for every rollback segment?
8. What data dictionary view contains a row for every active transaction?
9. What is the command to assign a transaction to a specific rollback segment?

Rollback segments are required for every transaction in the database. When users start transactions, they must be able to acquire a rollback segment; if they don't, they wait. Waits for rollback segments can cause serious performance problems. Waits for rollback segments are caused by incorrectly configured rollback segments or by simply having too few.

This chapter begins with an explanation of how rollback segments are used by transactions. The next part of the chapter explains how and why other components of Oracle use rollback segments. These include transaction rollback, read consistency, and transaction recovery. After explaining the use of rollback segments, the chapter focuses on the different types of rollback segments, as well as their creation. To determine the effectiveness of rollback segment usage and configuration, DBAs must know the data dictionary views that contain rollback segment and transaction information. After the data dictionary section, the chapter investigates the various maintenance operations that can be performed on rollback segments. The last part of the chapter covers troubleshooting rollback segment problems. This includes read consistency errors, insufficient space for transaction errors, problems with blocking sessions, and errors in taking tablespaces offline.

Rollback segments can often be a source of great frustration for DBAs. They can occupy a tremendous amount of space and are usually looking for more. They often generate errors like insufficient space or SNAPSHOT TOO OLD and seem to have a voracious appetite for disk space. But, rollback segments are not fully to blame. They need to be correctly configured and constantly monitored. DBAs also need to keep developers in check, ensuring that large transactions are split up. DBAs must also watch themselves. Most of the tools that can generate a large amount of rollback have features that enable the user to use less. The Import Utility, the SQL Loader Utility, and even the ALTER TABLE command have features for reducing the amount of rollback space required.

Once you have a sound understanding of their use, managing rollback segments is really not that difficult. You must make sure that there are enough of them to prevent contention and make sure they are appropriately sized to avoid dynamic extension or, what is worse, running out of space. If a transaction runs out of rollback space, it rolls back. If it took two hours to run out of space, it will probably take another two hours to rollback. Now four hours have elapsed and the transaction is right back where it started.

The Use of Rollback Segments

Whenever a transaction is started, a rollback segment is acquired. The header of the rollback segment is a transaction table. It records information about all transactions using that particular rollback segment. Each transaction, with the exception of Parallel DML, is assigned to one and only one rollback segment for the duration of the transaction. When a new transaction is started it will again be assigned to a rollback segment and not necessarily the same one as the previous transaction.

A transaction is a logical unit of work that can contain one or many SQL statements. If a user issues an UPDATE statement, that starts a transaction. If the user then issues five more UPDATE statements they are part of the same transaction. The transaction is an atomic unit which means that all statements inside the transaction either all commit or all rollback, depending on what the user does. If the user issues a commit, then all the statements are committed. If a rollback is entered, then all statements inside the transaction are rolled back. The first DML statement issued after a commit or rollback starts a new transaction.

**Caution**

DDL statements are self-contained transactions. Before executing the DDL statement, Oracle first issues a commit. This means that if a user was already inside a transaction, then this work is committed. You should avoid issuing DDL statements with other transactions as this can cause undesired results. For example, if you are in the process of updating, inserting, or deleting rows and subsequently issue a DDL statement (like creating a user or altering a table), all your work is committed.

When the user starts a transaction, it gets assigned to an available rollback segment. Rollback segments that are online are considered available. If there are multiple rollback segments online, Oracle chooses the least busy rollback segment for the transaction. If desired, a user can also specifically request a rollback segment for a given transaction. This may be done to ensure a transaction does not run out of rollback space. Many databases have rollback segments that are larger than others specifically for large transactions. In this case, it is up to the user to assign the transaction to the appropriate rollback segment. However, it is usually best to let Oracle assign transactions to rollback segments. Oracle ensures that transactions are evenly distributed across all available rollback segments ensuring that no one rollback segment is over used.

When the transaction is assigned to the rollback segment, it records transaction information in the header of the rollback segment. This is called the transaction table and each rollback segment has its own transaction table as each rollback segment has its own header. This transaction table is significant by telling all other users when the new data can and cannot be read. For example, if you start a transaction that updates a row in the SCHEDULEDCLASSES table, possibly changing the STATUS for a class that is going to run, you do not want users to see that information until you are satisfied with it. So, when other users query the row you have just updated, they first query the transaction table to see if your transaction has committed yet. If it has, then they can see the updated row; if it has not, they see the row as it existed before you updated it. When you commit your work, part of the commit process is to change the status of your transaction in the transaction table to “committed.”

Along with keeping track of transaction information, the rollback segments also store the before image of rows being changed. When changing the STATUS column of the SCHEDULEDCLASSES table from “Hold” to “Confirmed” the rollback segment your

transaction is assigned to records the data, as it existed prior to the change. In this example, the value of “Hold” is recorded in the rollback segment. When you issue the update command part of the process involves recording the before image of the rows you are changing in the assigned rollback. The information that is kept depends on the action being performed. The before image of an INSERT is just the ROWID while the before image of a DELETE is the entire row. For UPDATES, the rollback segment records the ROWID, the columns being updated and the before image of the columns. So DELETE operations require more rollback space than INSERTS. The amount of space required by an UPDATE statement depends on the columns being updated. Oracle estimates the amount of rollback space your transaction requires and reads the appropriate number of rollback blocks into the database buffer cache from the rollback segment. Those rollback blocks cannot be reused by your transaction or any other transaction until the transaction completes.

Rollback segments are made up of extents and blocks like all other segments. With rollback segments, however, the extents are used in a circular fashion. The rollback segment header keeps track of the current extent and allocates space for each new transaction. As all the blocks of the current extent are used, the next extent in line becomes the current extent until all extents have been used. When this happens, the rollback segment reuses the first extent and continues the same cycle. For example, if a rollback segment were made up of 10 extents, then the blocks on one extent could not be reused until the other 9 extents are used first. This prevents blocks from being overwritten too quickly, when other users may still need them for read consistency.



Read consistency is just one of the uses of rollback segments discussed in the section “Purpose of Rollback Segments.”

The before image of a change is used to rollback a transaction or to provide a read consistent image for another user attempting to read a row being modified by another user. If another user attempts to read the row from the SCHEDULEDCLASSES table that you are modifying, he or she will read a reconstructed image of that row from the rollback segment. Oracle actually reconstructs a new version of the row from the information provided in the rollback segment. This way, users only see committed transactions.

Figure 10-1 illustrates how rollback segments are used by transactions.



Figure 10-1 only illustrates how rollback segments are used by transactions. It is important to note that transaction information is also recorded in the redo log buffer.

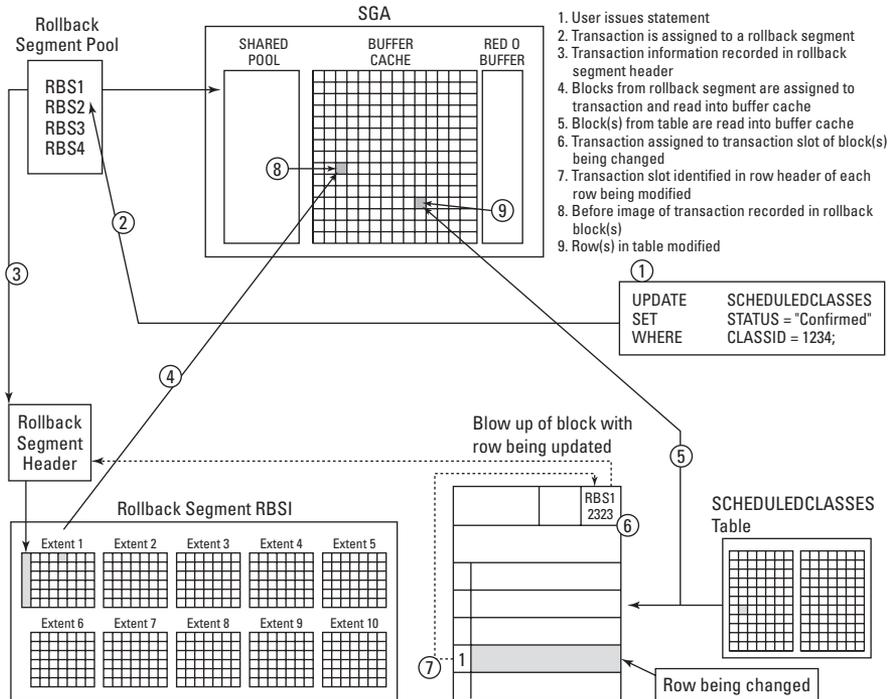


Figure 10-1: Rollback segments and transactions

Purpose of Rollback Segments

The purpose of rollback segments is to support rolling back transactions, recovering transactions, and read consistency.

Transaction rollback

When transactions change rows, the before image of the row is recorded in the rollback segment assigned to that transaction. The new value is recorded in the actual row of the table. If a user issues a rollback statement, Oracle rebuilds the changed rows from the before images located in the rollback segments. The same process is used if it is one row or thousands of rows being rolled back. Since all rows are rolled back, the time that it takes Oracle to perform this action depends on the number of rows involved in the transaction. If many rows are involved, this can take a long time to complete.

Transaction recovery

When there is an instance failure, all uncommitted changes in the buffer cache need to be rolled back during instance recovery. The rollback information, like the table information, is protected in the redo log files so that all rollback blocks are rebuilt and transactions can be rolled back when instance recovery happens.

Read consistency

This is probably the most common use of rollback segments and also the most confusing. When a transaction starts, the only user who can and should see the data being changed is the user issuing the change. Other users cannot see uncommitted transactions being performed by other users. In Figure 10-1, a row in the SCHEDULEDCLASSES table is being modified and no commit has been issued. If another user attempts to read the same row, his or her transaction detects that another user is modifying the row. That user gets the transaction information from the header of the block and goes to the appropriate rollback segment; in this example, it is RBS1. The user goes to the header of rollback segment RBS1 and checks the status of the transaction. In this case it is an uncommitted transaction, therefore, the user cannot read the row, as it exists in the table, and the transaction will reconstruct a new table block using the information from the rollback segment. The query then has an image of the row as it existed before it was modified. If another user attempts to do the same thing, he or she also reads the rollback segment. This process continues until the user who made the change issues a commit. Once the change is committed, users will be able to read the row from the table and not from the rollback segment.

Also, when a user starts a query, the current System Change Number (SCN) is determined. Oracle does not allow for rows to be read that have a higher SCN. Oracle knows about rows that have changed after the start of a query and prevents the rows from being read. Queries cannot read any blocks that have changed after the query started. For example, if a user starts a long query at 9:00 a.m. that takes 20 minutes to complete, the user cannot read any rows that changed since the query started. If another user modifies a row at 9:10 a.m. and commits, the query that started at 9:00 a.m. will detect that the row has changed and read the image of the row from the rollback segment. If the image no longer exists in the rollback segment, the “Snapshot Too Old” error message is returned. If a user starts a transaction at 8:59 and commits at 9:01, this will also be considered an inconsistent image of the row and since the SCN number is assigned at the time of the commit, Oracle knows to read the before image from the rollback segment.

Types of Rollback Segments

There are three different types of rollback segments, SYSTEM, non-SYSTEM, and DEFERRED.

SYSTEM rollback segment

The SYSTEM rollback segment is always created in the SYSTEM tablespace when the database is created. The name of the rollback segment is SYSTEM and is intended to be used exclusively by Oracle when it needs to make changes to the data dictionary. The SYSTEM rollback segment can only be used to make changes to objects in the SYSTEM tablespace. As soon as other tablespaces are created and transactions occur on objects in those tablespaces, Oracle forces the creation of new rollback segments. No special consideration needs to be given to this rollback segment because it is maintained by Oracle.

non-SYSTEM rollback segments

As new tablespaces are created, non-SYSTEM rollback segments must also be created. It is a good idea to create a separate tablespace exclusively for non-SYSTEM rollback segments. These rollback segments must be created after the database has been created by the DBA. There are two types of non-SYSTEM rollback segments, Private and Public.

Private

Private is the rollback segment used in a dedicated-server environment. The Private rollback segment belongs to the instance that brought it online. Rollback segments are either brought online during instance startup if they are listed in the ROLLBACK_SEGMENTS INIT.ORA parameter file or manually using the ALTER ROLLBACK SEGMENT segment_name ONLINE command. Private is the default type of rollback segment when they are created.



Bringing rollback segments online is covered in more detail in the section “Maintaining Rollback Segments.”

Public

Public rollback segments are associated with Oracle Parallel Server. The rollback segments form a pool that can be brought online by any instance of an Oracle Parallel Server. The Public keyword must be specified during rollback segment creation to make it a Public rollback segment.

Oracle Parallel Server

Oracle Parallel Server is a complex Oracle configuration that involves having multiple instances and a single database or one set of data files. The idea behind Parallel Server is to take advantage of having multiple servers servicing user requests, each server with its own instance. All instances share the same data files for reading and writing of table information. This installation is usually reserved for large databases with substantial Oracle DBA expertise.

DEFERRED rollback segments

DEFERRED rollback segments are created and maintained by Oracle automatically in the SYSTEM tablespace (watch free space). No administration is required by the DBA. They may be created when tablespaces are taken offline using the immediate option. The immediate option does not wait for users to commit or rollback uncommitted transactions. When the tablespace is brought online again, the DEFERRED rollback segments are used to rollback the transactions. These rollback segments are dropped when no longer needed.

Rollback Segment Operations

Understanding how transactions use rollback segments is an important element of an Oracle database that DBAs must master. This knowledge is critical for planning the amount of space required by rollback segments, as well as understanding the performance implications that rollback segments have in an Oracle database.

How transactions use rollback segments

When a transaction starts, it is assigned to a rollback segment in one of two ways. Implicitly by Oracle or explicitly, if the user requests a specific rollback segment by issuing the following command:

```
SET TRANSACTION USER ROLLBACK SEGMENT rollback_segment_name
```



Caution

This must be the first command executed in the transaction, and the rollback segment must be online for this command to work.

If this command is not issued, Oracle chooses among the available rollback segments that are online. The first choice is to use the rollback segment with the fewest transactions. If there are ties, it chooses the rollback segment with the smallest transactions, and, if there are still ties, it chooses the rollback segment with the oldest transactions. This ensures that all rollback segments will be evenly used and that no one rollback segment will be inundated with transactions, which could cause contention for the rollback segment.

Once assigned to a rollback segment, Oracle automatically does the following:

1. Record transaction information in the header of the rollback segment.
2. Estimate the amount of rollback space required by the transaction. This determines how many rollback blocks need to be assigned to the transaction.
3. Move the pointer in the rollback segment header to the first block after the blocks allocated for the transaction. The rollback segment header tracks the current extent and the next block to be used. The extents and blocks are always used sequentially, which prevents blocks from being overwritten before all other blocks in the rollback segment have been used.

4. Read the rollback blocks assigned to the transaction into the database buffer cache, if they do not already exist, for storing the before image of the rows being changed.

Multiple transactions often share the same rollback segment. Each transaction using the rollback segment has its transaction information recorded in the header of the rollback segment, which is also called the transaction table. Rollback segments must be made up of at least two extents, and transactions can share the same extent but not the same rollback blocks. The extents are used in a sequential fashion. If a rollback segment is made up of 10 extents, extent 1 is used first, then extent 2, and so on until extent 10 is used. When all the blocks from extent 10 are used, extent 1 will be reused. Figure 10-2 illustrates how the extents of a rollback segment are used.

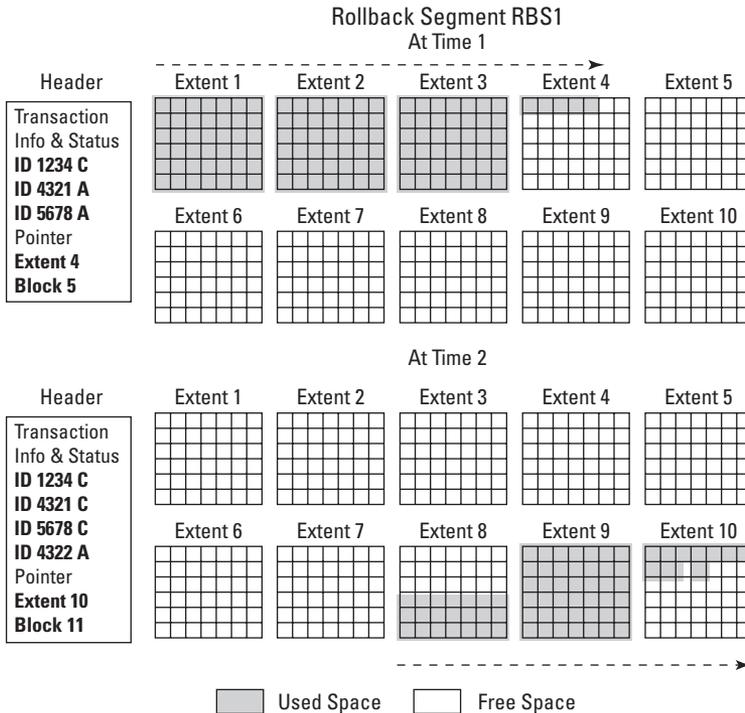


Figure 10-2: How rollback segments use space

In Figure 10-2 you can see how rollback extents and blocks are used by transactions. In Time 1 you can see that extent 4 is the current extent and block 5 is the next block to be used. The extents 1, 2, 3, and 4 have active transactions. In Time 2 you can see that extent 10 is now the current extent and block 11 is the next to be used. Extents 1 through 7 are now free, meaning that the transactions that once used the space have completed. Once all the blocks in extent 10 have been used, the pointer will try to move back to extent 1. It can only move back to extent 1 if there are no

active transactions on extent 1. That is, all transactions on the extent that the pointer is trying to move to have either committed or rolled back. If there are active transactions on the extent that the pointer is trying to move to, Oracle is forced to grow the rollback segment.



The growth of rollback segments is covered in the section called “Growth of rollback segments.”

Growth of rollback segments

The extents of rollback segments are used in a sequential and circular fashion. If there are 10 extents for a rollback segment then they are used in order, starting at 1 and going up to 10. When all the space on extent 10 has been used, it circles back and uses extent 1 again. Refer to Figure 10-3 for an example of how space in rollback segments is used by transactions.

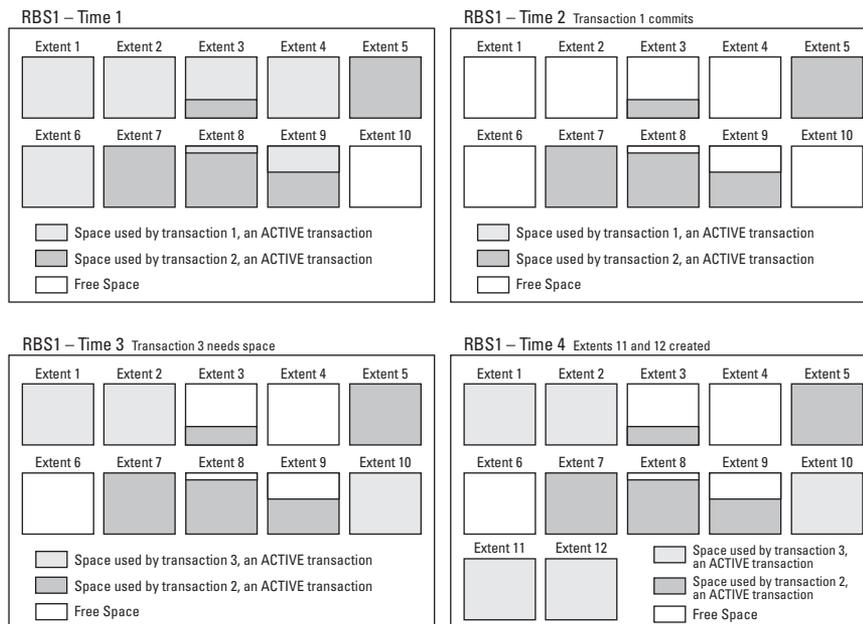


Figure 10-3: Rollback segment space used by transactions

At Time 1, transactions 1 and 2 have used all the free space in the rollback segment except for Extent 10. Transaction 1 started on Extent 1 and transaction 2 started on Extent 3.

At Time 2, transaction 1 commits, which frees up space.

At Time 3, transaction 3 starts on Extent 10. However, the transaction requires more space than is available on Extent 10 so it is forced to wrap. A *wrap* occurs when a transaction cannot be contained on a single extent and wraps to the next. When the wrap occurs, the pointer in the rollback segment header goes to the next extent in the sequence. In this case it is back to Extent 1. Since there are no active transactions, transaction 3 can use Extent 1. Still more space is required so it wraps to Extent 2 and, when it tries to move the pointer to Extent 3, it sees that there is an active transaction on Extent 3—transaction 2—therefore, Extent 3 cannot be used. It can also not be skipped. Oracle does not let the pointer in the rollback segment header skip extents with active transactions. Therefore, Oracle is forced to extend the rollback segment. *Extending* means that additional extents are created.

In Time 4, you can see that Extents 11 and 12 have been created. Oracle will continue extending the rollback segment as required until the MAXEXTENTS parameter for the rollback segment has been reached, there is no space left in the tablespace, or until the blocking transaction—in this case, transaction 2—completes. If transaction 2 completes, Oracle will start using Extent 3, 4, and so on. The newly created extents remain in use until the rollback segment shrinks.



You can find more on the MAXEXTENTS parameter later in this chapter in the section “Setting MAXEXTENTS.” Shrinks are covered in the section called “Shrinkage of rollback segments.”

Shrinkage of rollback segments

If a rollback segment extends, it is likely that the space required that caused the extend is a temporary measure. There may have been an abnormally large transaction or an abnormally large number of concurrent transactions that caused the rollback segment to extend. If this is not a permanent behavior for the database, then the space used by the extra extents is likely just wasted space and should be reclaimed. If rollback segments are consistently extending, it is usually an indication of a configuration problem.



Rollback segment configuration is covered in the “Creating Rollback Segments” section.

Space can be reclaimed one of two ways—either manually by the DBA through an ALTER ROLLBACK SEGMENT SHRINK command or automatically by Oracle if the OPTIMAL parameter is set for the rollback segment.

The OPTIMAL parameter can only be specified for rollback segments and controls the size that a rollback segment should try to shrink to if extends have occurred. The minimum size for OPTIMAL must be equal to the value of INITIAL * MINEXTENTS, which is the initial size of the rollback segment when it is created. Therefore, if a rollback segment has extended, by default it shrinks back to the value of OPTIMAL.

By specifying a value for `OPTIMAL`, Oracle can release space required by the large transactions that forced the growth. The deallocation of space is not done immediately after the transaction that forced the extend completes. Rather, Oracle waits until the pointer in the rollback segment header moves to the next extent. At that time, space is deallocated from the rollback segment if the size of the rollback segment exceeds the value of `OPTIMAL` and there are contiguous inactive extents. If both conditions are met, Oracle tries to deallocate extra space down to the value of `OPTIMAL`. It only succeeds if the extents being deallocated are not in use. If the value of `OPTIMAL` is 10MB and the total size of the rollback is 14MB, made up of 14 one-megabyte extents, then Oracle attempts to deallocate four extents to get the rollback segment back to 10MB, the value of `OPTIMAL`. If the next three extents that the pointer in the rollback segment header is moving to are inactive but the one after that is not, then Oracle can only deallocate two extents. It will not deallocate three since the fourth one is still active. If it deallocated the third and attempted to move the pointer into the fourth, it would just need to allocate space again. Oracle performs the same tests the next time the pointer is moved until the size of the rollback segment is equal to the value of `OPTIMAL`.

Figure 10-4 illustrates the preceding example. A large transaction forced the rollback segment to grow or extend and the space is eventually deallocated when the correct circumstances are encountered. When deallocating space, Oracle always deallocates the oldest extents because oldest extents have the greatest probability of not being needed by any other users. This reduces the likelihood of a “Snapshot Too Old” error message occurring because space was deallocated.

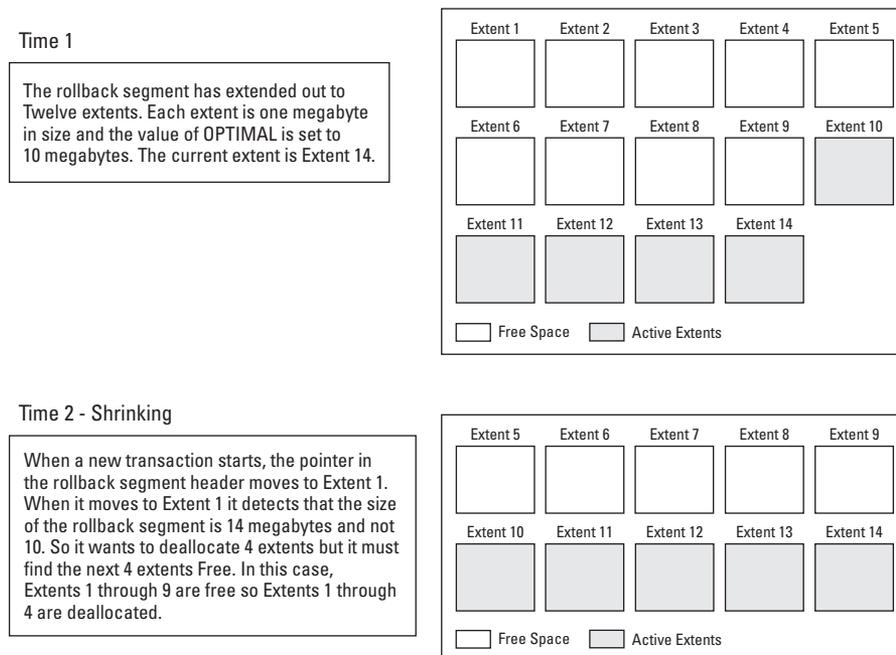


Figure 10-4: Shrinking rollback segments

The value of OPTIMAL should be increased if shrinks are happening in the database. It is usually an indication of an inappropriate setting. Rollback segments are extending from use or large transactions and space is subsequently being deallocated or shrunk back after the transaction completes. If many “shrinks” are happening, space is being allocated, then deallocated, allocated, then deallocated, and so on, which is inefficient. If transactions are forcing the rollback segment to extend, increase the value of OPTIMAL so that the deallocation or shrinking can be avoided. By not shrinking, the future extends will also be reduced. The rollback segment should be at a better size if the shrinks stop after a change to OPTIMAL.



Modifying rollback segments is covered in the “Maintaining Rollback Segments” section.

Creating Rollback Segments



Create rollback segments using appropriate storage settings

Creating rollback segments is something that should be done in all databases. If non-SYSTEM tablespaces are created, then non-SYSTEM rollback segments must also be created. The CREATE ROLLBACK SEGMENT system privilege and a quota in the tablespace or UNLIMITED TABLESPACE system privilege is required to create a rollback segment. DBAs have the required privileges to create rollback segments and it is part of their job to create them. The following is the syntax for creating rollback segments:

```
CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment_name
  [TABLESPACE tablespace_name]
  [STORAGE ([INITIAL integer[K|M]]
            [NEXT integer[K|M]]
            [MINEXTENTS integer]
            [MAXEXTENTS {integer|UNLIMITED}]
            [OPTIMAL {integer[K|M]|NULL}]
            )
  ]
```

The following is a list of restrictions to creating rollback segments:

- ♦ User must have CREATE ROLLBACK SEGMENT system privilege.
- ♦ User must have quota in tablespace where the rollback segment is being created or have the UNLIMITED TABLESPACE system privilege.
- ♦ Private rollback segments are the default unless the keyword Public is specified. The type cannot be altered from Private to Public or vice versa.
- ♦ MINEXTENTS must be at least two and defaults to this value if MINEXTENTS is not set.

- ♦ PCTINCREASE is always 0 and cannot be changed. If specified, Oracle returns an error.
- ♦ OPTIMAL must be at least INITIAL * MINEXTENTS, the initial size of the rollback segment when created.

Guidelines for creating rollback segments

The following is a list of guidelines for creating rollback segments.

Size

Rollback segments present a particularly interesting dilemma for DBAs. There are conflicting recommendations that exist regarding their size that can be confusing. If large rollback segments are created, the likelihood that the rollback segments will extend is reduced but so is the likelihood of a cache hit when blocks are needed. If created small, the chance of a cache hit is increased but so are the odds that they are going to extend, so what do you do? It depends on the nature of the database. If the database were more geared towards small, quick transactions, then the database would benefit from smaller rollback segments. This increases the likelihood that rollback blocks already exist in the database buffer cache for when they are needed. The chance of the rollback segments extending is minimal. If the database has large transactions, then large rollback segments are more beneficial. The larger the rollback segment, the less likely is dynamic extension. If the rollback segment runs out of space, it will dynamically extend until it reaches the MAXEXTENTS parameter or until there is no free space left in the tablespace. This dynamic extension is expensive and should be avoided.

While these particular situations are straightforward and easy to resolve, what happens when you have a combination of large and small transactions? The problem is not that easily solved. Generally, the rule of thumb is: Always to avoid dynamic extension. Dynamic extension is much more expensive than is the IO that results from rollback segments not existing in the cache because they are too big. However, care must be given not to make them too big which leads to wasted space. Look for a happy medium. In these scenarios, it is best to create rollback segments large enough to avoid dynamic extension, but monitor them after creation to ensure that they are not extending. Be sure to set INITIAL=NEXT to ensure that all extents of the rollback segment are the same size.

Another fix to this problem is creating two different sizes of rollback segments. Create many small rollback segments for the small transactions and a couple of large ones for the large transactions. This implementation requires more work by DBAs and developers but is usually the best solution. To ensure that the large transactions are assigned to the large rollback segments, users must specifically request the rollback segment by using the SET TRANSACTION USE ROLLBACK SEGMENT segment_name command.



Monitoring rollback segments is covered in the section titled "Getting Information about Rollback Segments."

When determining the rollback segment size, you must consider the problems that rollback segments face and the problems you want to avoid as a DBA. The two main problems are rollback segments running out of room and rollback segments extending. When this happens, the transaction fails and starts rolling back. If this process took two hours to reach the critical point of failure, then it is likely going to take at least another two hours for the transaction to rollback. If this happens, four hours have gone by and nothing has happened with the transaction.

The second problem is that of rollback segment extending, which you want to avoid. The most popular solution for this is to simply ensure that many extents are created. Set `MINEXTENTS` to a large value. This will reduce the chances of the rollback segment extending. But is this true? This is certainly the most common solution, but also the most misunderstood. The biggest factor influencing the likelihood that a rollback segment is going to extend is not the number of extents but rather the total size of the rollback segment. Rollback segments extend when the pointer in the rollback segment header attempts to advance to the next extent segment but that extent has an active transaction. Oracle is forced to extend. If there are two rollback segments, `RBS1` and `RBS2` each with 20 extents, however, `RBS1`'s extents are 1MB in size while `RBS2`'s extents are 10KB in size, which one is more likely to extend? `RBS1` is a total of 20MB in size while `RBS2` is a total of 200KB in size. `RBS2` is more likely to extend. Only 200KB of space is used before Oracle attempts to reuse the first extent or wraps completely around all extents. With `RBS1`, 20MB must be used before Oracle attempts to reuse space. Along the same lines, which is more likely to extend: a rollback segment with 200 extents all 10KB in size or a rollback segment with 4 extents all 5MB in size? The answer is the first one, even though it has 200 extents. The total size of that rollback segment is only 2000KB or 2MB, while the other is 20MB. The first one only needs to use 2MB of space before attempting to write to the first extent again while the second will not attempt to reuse the first extent until 20MB of space has been used.

The point is this: Be careful not to fall into the trap of simply creating many extents and thinking that you have solved all problems of rollback segments extending. The total size of the rollback segment is the more important factor.

Location

Create at least two tablespaces for storing rollback segments and try to ensure that the rollback segments are located on separate disks. This helps reduce disk contention between rollback segments and other segments, as well as helps prevent fragmentation. Rollback segments have a tendency to grow and shrink, a perfect climate for fragmentation. If they are on their own tablespaces, the chance of fragmentation is zero.

You should also consider two other important issues. First, use locally managed tablespaces for the rollback segments. This improves the performance of the rollback segments that dynamically extend. Second, if you don't use locally managed tablespaces, ensure that default storage parameters are set for the tablespace. Identify the size and number of extents desired for each rollback segment and set

the appropriate default values for INITIAL, NEXT, MINEXTENTS, and MAXEXTENTS. OPTIMAL cannot be specified as a tablespace default; it can only be specified during rollback segment creation or when altering rollback segments.

Sizing INITIAL and NEXT

Before deciding upon a size for INITIAL and NEXT, you should have already determined what the total size for the rollback segment is going to be. If this is done, then determining values for INITIAL and NEXT is pretty simple. If you want the total size for each rollback segment to be 20MB made up of 40 extents, the values of INITIAL and NEXT should be 500KB.

Multiple transactions can write to the same extent of a rollback segment. What you are trying to avoid are wraps. If many wraps are encountered, consider increasing the size of INITIAL and NEXT.



Changing the values of INITIAL and NEXT can only be done by dropping and recreating the rollback segment.

Setting OPTIMAL

Optimal should always be set equal to the value of INITIAL * MINEXTENTS, the initial size of the rollback segment when it is created. If set to a higher value, the rollback segment is allowed to grow and will not be shrunk until the rollback segment is bigger than OPTIMAL. It does not, however, make much sense to do this. Allowing the rollback segment to grow — dynamically extend — is expensive. If you have determined that the optimal size for rollback segments is 6MB, then create them that big. Do not create them 4MB in size and set OPTIMAL to 6MB. This only causes the rollback segment to dynamically extend. If 6MB is the number you have determined, then create them that big initially, avoiding the dynamic extension.

The only benefit that OPTIMAL has is if the rollback segment was inappropriately sized when created. This gives DBAs a way to increase the size of the rollback segment without having to drop and re-create it. Since INITIAL and MINEXTENTS cannot be changed once the rollback segment is created, the only way to make it bigger is to increase OPTIMAL. This helps prevent the rollback segment from extending and shrinking.

Setting MAXEXTENTS

You should always set MAXEXTENTS to a value, usually a large value, but nonetheless a value. If you set it to UNLIMITED, then the only thing that is going to stop a runaway transaction — that is, an accidentally large transaction run by a user — is the size of the tablespace. If the tablespace is 500MB in size, the rollback segment could possibly continue growing unnecessarily for hundreds of megabytes. Setting the value helps reduce the time and resources used by abnormally large user transactions or potential application errors.

Creating rollback segments using OEM Storage Manager

Outlined below are the steps for creating rollback segments using OEM.

STEP BY STEP: Creating rollback segments with DBA Studio

1. Launch DBA Studio and select the Launch DBA Studio in Stand Alone option. Start ⇨ Programs ⇨ Oracle OEM (This name may be different on your installation.) ⇨ Database Administration ⇨ DBA Studio.
2. Select the CERTDB database and enter the login information if required.
3. Select Storage.
4. Select Rollback Segments.
5. Right-click the Rollback Segments folder and click Create from the menu.
6. On the General page, enter the name for the rollback segment, select the tablespace, and indicate if it is a Public or Private rollback segment. You can also choose to bring the rollback segment ONLINE after creation. Figure 10-5 shows an image of the Create Rollback Segment screen from DBA Studio.

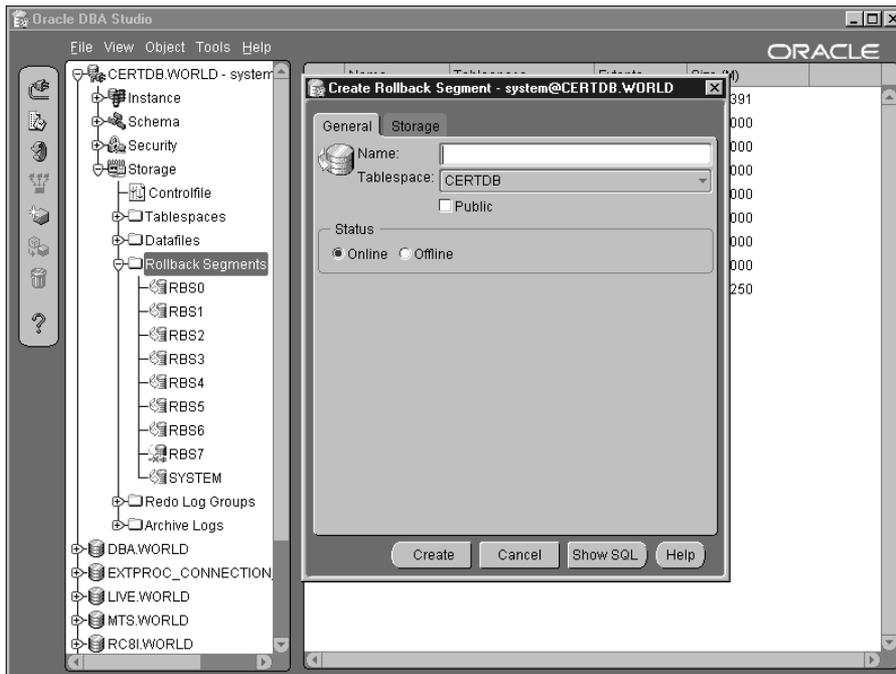


Figure 10-5: Entering rollback information

- Click the Storage button and specify the storage options for the rollback segment. When done, click the Create button.

Getting Information about Rollback Segments

Objective

Obtain rollback segment information from the data dictionary

Several data dictionary views contain information about rollback segments. The two main views are `DBA_ROLLBACK_SEGS`, which lists information about all rollback segments, and `V$ROLLSTAT`, which provides statistics for rollback segments. These statistics include the number of shrinks, wraps, and extends per rollback segment — valuable tuning information. The `V$ROLLSTAT` view only has statistics for online rollback segments and, like all other `V$` views, it is a *virtual view*. That means the statistics reported are based upon available memory information. For rollback segments, the statistics are kept since the last time the rollback segment was brought online, which is usually when the instance was started since rollback segments are usually brought online when the database is started. If a rollback segment was brought online after the instance, then the information recorded in `V$ROLLSTAT` will be from that time forward. Remember that when the instance shuts down, the `V$` views are cleared out.

There are a couple of other views that contain rollback segment information. The `V$TRANSACTION` view lists all active transactions. This includes the rollback segment the transaction is assigned to. The `V$SESSION` view is linked to the `V$TRANSACTION` view to find out which users have active transactions.

DBA_ROLLBACK_SEGS

The `DBA_ROLLBACK_SEGS` data dictionary view has one row for every rollback segment in the database, even rollback segments that are offline.

COLUMN NAME	DESCRIPTION
<code>SEGMENT_NAME</code>	Name of the rollback segment.
<code>OWNER</code>	Owner of the rollback segment. This column indicates the type of rollback segment. If <code>Public</code> is the <code>OWNER</code> then it is a <code>Public</code> rollback segment. If anything other than <code>Public</code> , it is a <code>Private</code> rollback segment.
<code>TABLESPACE_NAME</code>	Name of the tablespace that contains the rollback segment.

Continued

<i>COLUMN NAME</i>	<i>DESCRIPTION</i>
SEGMENT_ID	Segment ID number for the rollback segment.
FILE_ID	File ID of the file that contains the rollback segment header.
BLOCK_ID	Block ID for the block that contains the rollback segment header.
INITIAL_EXTENT	Size of the INITIAL parameter. This is the size of the first extent created for the rollback segment specified in bytes.
NEXT_EXTENT	Size of the NEXT parameter. This is the size of all extents, except the first, created for the rollback segment specified in bytes.
MIN_EXTENTS	The number of extents initially created for the rollback segment.
MAX_EXTENTS	Maximum number of extents created for the rollback segment. If UNLIMITED is specified, the value is still represented as an integer. The maximum number of extents for a database with an 8K blocks size is 32,765. Avoid setting MAX_EXTENTS to UNLIMITED.
PCT_INCREASE	Always 0 for rollback segments
STATUS	Status of the rollback segment. Some of the possible values are: ONLINE OFFLINE PARTLY AVAILABLE OFFLINE PENDING
INSTANCE_NUM	Parallel server number instance that owns the rollback segment.
RELATIVE_FNO	Relative file number containing the header of the rollback segment.

Here are some useful queries that can be run against the DBA_ROLLBACK_SEGS view. The first lists all rollback segments, the tablespace the rollback segment is contained on, and the status. A status of ONLINE indicates it is available for use.

```
SQL> COL "ROLLBACK NAME" for A15
SQL> SELECT SEGMENT_NAME "ROLLBACK NAME",
2         TABLESPACE_NAME,
3         OWNER,
4         STATUS
5 FROM   DBA_ROLLBACK_SEGS
6 /
```

ROLLBACK NAME	TABLESPACE_NAME	OWNER	STATUS
SYSTEM	SYSTEM	SYS	ONLINE
RBS0	RBS	PUBLIC	ONLINE
RBS1	RBS	PUBLIC	ONLINE
RBS2	RBS	PUBLIC	ONLINE
RBS3	RBS	PUBLIC	ONLINE

```

RBS4          RBS          PUBLIC ONLINE
RBS5          RBS          PUBLIC ONLINE
RBS6          RBS          PUBLIC ONLINE

```

8 rows selected.

The following query can be run to extract storage information. The size of the extents, the number of extents created initially, and the maximum number of extents allowed. If `MAX_EXTENTS` is 32,765, then you know the value has been set to `UNLIMITED` and should be investigated.

```

SQL> SELECT SEGMENT_NAME "ROLLBACK NAME"
2      ,      INITIAL_EXTENT "INITIAL"
3      ,      NEXT_EXTENT "NEXT"
4      ,      MIN_EXTENTS "# CREATED"
5      ,      MAX_EXTENTS "MAX EXT"
6 FROM      DBA_ROLLBACK_SEGS
7 /

```

ROLLBACK NAME	INITIAL	NEXT	# CREATED	MAX EXT
SYSTEM	57344	57344	2	505
RBS0	524288	524288	8	4096
RBS1	524288	524288	8	4096
RBS2	524288	524288	8	4096
RBS3	524288	524288	8	4096
RBS4	524288	524288	8	4096
RBS5	524288	524288	8	4096
RBS6	524288	524288	8	4096

8 rows selected.

From the results of the query, you can see that all the non-SYSTEM rollback segments are the same size. If your database has a large or a couple of large rollback segments for large transactions, you should be able to identify them from this query.

V\$ROLLNAME

The `V$ROLLNAME` view contains the name of the rollback segment (`NAME`), as well as the rollback segment number, called the Undo Segment Number (`USN`). The `USN` is the column that is stored in the statistic tables for rollback segments. It contains one row for every rollback segment that does not have a status of `OFFLINE`.

V\$ROLLSTAT

The `V$ROLLSTAT` view contains one row of statistics for all rollback segments that do not have a status of `OFFLINE`. The statistics in this view offer the best tuning information for rollback segments.

COLUMN NAME	DESCRIPTION
USN	The Undo Segment Number. You join V\$ROLLSTAT with V\$ROLLNAME by this column to get the name of the rollback segment.
EXTENTS	The number of extents created for the rollback segment.
RSSIZE	Size in bytes of the rollback segment.
WRITES	The number of bytes written to the rollback segment.
XACTS	The number of active transactions using the rollback segment. The transaction information is stored in the V\$TRANSACTION view.
GETS	The number of times transactions have requested the header of the rollback segment.
WAITS	The number of times transactions have requested the header of the rollback segment and have not gotten it. This is an important column for tuning rollback segments. If this number is high relative to the number of gets, or 1 percent or more of GETS, then more rollback segments should be created. Waits for the rollback segment header are expensive and can be avoided with the addition of more rollback segments.
OPTSIZE	The value of OPTIMAL for the rollback segment. The V\$ROLLSTAT view is the only view that contains this value. It should be equal to the value of INITIAL*MINEXTENTS for each rollback segment unless the rollback segments were created with the incorrect size. Join the V\$ROLLSTAT view with DBA_ROLLBACK_SEGS to determine if OPTSIZE is correctly set.
HWMSIZE	The high-water mark size is the maximum size that the rollback segment has grown to in bytes since the instance started or the rollback segment was brought online. This column can be used to determine the optimal size for rollback segments to avoid rollback segments extending.
SHRINKS	The number of times the rollback segment has grown past the value of OPTSIZE and then been shrunk to deallocate space. In most databases, if the value of SHRINKS is high or is increasing, it indicates that the value of OPTIMAL needs to be increased. The value of OPTIMAL should be high enough to prevent rollback segments from shrinking. The exception to this is the rare large transactions that require more rollback space than normal transactions. For these transactions, it should be expected that shrinks will occur.

COLUMN NAME	DESCRIPTION
WRAPS	The number of times a transaction has started on one extent of a rollback segment and been forced to wrap onto the next extent because not enough space was available in the first extent it grabbed. Wraps can be an indication that the extent sizes are too small. If the number of wraps is increasing, investigate the size of your transactions compared to the size of the rollback segment extents to see if a better size can be set.
EXTENDS	The number of times the rollback segment has extended or created new extents. Rollback segments extend when the header of the rollback segment attempts to move into an extent with an active transaction. Since it cannot move the pointer into these extents it is forced to extend or create more extents. This is called dynamic extension and should be avoided. This can have a dramatic negative impact on performance. If the number of extends is high or is increasing, it indicates that there are either not enough rollback segments or that they are too small. Try adding more rollback segments first and if the number of EXTENDS does continue to increase, consider resizing the rollback segments. Some EXTENDS are acceptable for the large transactions. You may consider creating a large rollback segment specifically for the large transactions to avoid the regular rollback segments extending.
AVESHINK	Total size of freed extents divided by number of shrinks.
AVEACTIVE	Current average size of active extents, where "active" extents have uncommitted transaction data.
STATUS	ONLINE if the segment is online, or PENDING OFFLINE if the segment is going offline but some active transactions are using the rollback segment. When the transactions complete, the segment is supposed to go offline. However, you will need to reissue the OFFLINE command. This is a bug. You will never see a status of OFFLINE as the V\$ROLLSTAT view only contains rollback segments that are ONLINE or PENDING OFFLINE.
CUREXT	Current position of the rollback segment pointer. This is the current extent that is being used.
CURBLK	Current position of the rollback segment pointer. This is the current block that is being used.

Here are some useful queries that use the V\$ROLLSTAT and V\$ROLLNAME views.

```
SQL> SELECT RN.NAME "Name"
2 ,      RS.EXTENTS
3 ,      RS.RSSIZE "SIZE"
4 ,      RS.OPTSIZE "OPTIMAL"
5 ,      RS.HWMSIZE
6 ,      RS.XACTS "TRANS"
7 ,      STATUS
8 FROM   V$ROLLNAME RN,
9        V$ROLLSTAT RS
10 WHERE  RS.USN = RN.USN
11 /
```

Name	EXTENTS	SIZE	OPTIMAL	HWMSIZE	TRANS	STATUS
SYSTEM	5	401408		401408	0	ONLINE
RBS0	8	4186112	4194304	4186112	0	ONLINE
RBS1	8	4186112	4194304	4186112	0	ONLINE
RBS2	8	4186112	4194304	4186112	0	ONLINE
RBS3	8	4186112	4194304	4186112	0	ONLINE
RBS4	8	4186112	4194304	4186112	0	ONLINE
RBS5	8	4186112	4194304	4186112	0	ONLINE
RBS6	8	4186112	4194304	4186112	0	ONLINE

This query gets the number of extents, the total size of the rollback segment **SIZE**, the value of **OPTIMAL**, the high-water mark, the number of active transactions using rollback segments and the status.

To compare the value of **OPTIMAL** against the values of **INITIAL*MINEXTENTS** for each rollback segment, run the following query:

```
SQL> SELECT RN.NAME "Name"
2 ,      RS.OPTSIZE "OPTIMAL"
3 ,      DBA_RS.INITIAL_EXTENT * DBA_RS.MIN_EXTENTS "CREATED
SIZE"
4 FROM   V$ROLLNAME RN,
5        V$ROLLSTAT RS,
6        DBA_ROLLBACK_SEGS DBA_RS
7 WHERE  RS.USN = RN.USN
8 AND    RN.NAME = DBA_RS.SEGMENT_NAME
9 /
```

Name	OPTIMAL	CREATED	SIZE
SYSTEM		114688	
RBS0	4194304	4194304	
RBS1	4194304	4194304	
RBS2	4194304	4194304	
RBS3	4194304	4194304	
RBS4	4194304	4194304	

```
RBS5      4194304      4194304
RBS6      4194304      4194304
```

8 rows selected.

The results from this query indicate the value of `OPTIMAL` is set to the value of `INITIAL * MINEXTENTS` for the rollback segments. This is the ideal setting and should only differ if the rollback segments were not created using the correct storage parameters.

The last query is one that is used for tuning rollback segments. This query selects the number of gets, waits, shrinks, wraps, and extends for each rollback segment.

```
SQL> SELECT RN.NAME "Name"
       2 ,      RS.GETS
       3 ,      RS.WAITS
       4 ,      RS.SHRINKS
       5 ,      RS.WRAPS
       6 ,      RS.EXTENDS
       7 FROM    V$ROLLNAME RN,
       8         V$ROLLSTAT RS
       9 WHERE   RS.USN = RN.USN
      10 /
```

Name	GETS	WAITS	SHRINKS	WRAPS	EXTENDS
SYSTEM	294	0	0	0	0
RBS0	255	0	0	0	0
RBS1	272	0	0	0	0
RBS2	261	0	0	0	0
RBS3	459	0	0	0	0
RBS4	251	0	0	0	0
RBS5	603	0	0	0	0
RBS6	1518	0	0	0	0

The query is analyzed based on the following criteria:

- ♦ If the value of `WAITS` is one percent of `GETS` or the number of `WAITS` is increasing, consider adding more rollback segments.
- ♦ If there are `SHRINKS` for rollback segments it usually indicates that the rollback segments are too small. Consider increasing the value of `OPTIMAL` to reduce the number of shrinks. If the number of `SHRINKS` and/or `EXTENDS` is high for only one rollback segment, it usually indicates that there was one large transaction responsible for the growth. Consider creating a large rollback segment and assigning the large transactions to it.
- ♦ `WRAPS` indicates the number of times transactions have started on one extent and wrapped to another. A value for `WRAPS` may indicate that the extent sizes are not large enough. Consider recreating the rollback segments with larger extent sizes to reduce the number of wraps.

- ♦ **EXTENDS** is the number of times the rollback segment grew. If this number is high for all rollback segments and is increasing, it indicates that the rollback segments are either too small or there are not enough of them. Add rollback segments and if the number is still increasing consider increasing the size. If there are extends for only some of the rollback segments, it is usually caused by one or two large transactions. Consider creating a large rollback segment for those transactions and explicitly assign those transactions to the large rollback segment.

You can determine the size of the large transactions one of two ways. Either by querying the **V\$TRANSACTION** view before the transaction completes or by querying the **V\$ROLLSTAT** view before and after the transaction. If you cannot stop a transaction before it completes, you must use the **V\$ROLLSTAT** method.

```
SQL> SELECT RN.NAME "Name"
2      ,      RS.WRITES "SIZE IN BYTES"
3 FROM      V$ROLLNAME RN,
4           V$ROLLSTAT RS
5 WHERE     RS.USN = RN.USN
6 /
```

Name	SIZE IN BYTES
SYSTEM	9606
RBS0	6100
RBS1	52282
RBS2	5258
RBS3	2198
RBS4	5682
RBS5	3434
RBS6	65496

THEN RUN THE TRANSACTION.

```
UPDATE SCHEDULEDCLASSES
SET    STATUS = STATUS;
```

```
SQL> UPDATE STUDENT.SCHEDULEDCLASSES
2 SET    STATUS=STATUS
3 ;
```

3 rows updated.

```
SQL> SELECT RN.NAME "Name"
2      ,      RS.WRITES "SIZE IN BYTES"
3 FROM      V$ROLLNAME RN,
4           V$ROLLSTAT RS
5 WHERE     RS.USN = RN.USN
6 /
```

Name	SIZE IN BYTES
------	---------------

SYSTEM	9606
RBS0	6100
RBS1	52282
RBS2	5258
RBS3	2606
RBS4	5682
RBS5	3434
RBS6	65496

8 rows selected.

From this, you can see that the transaction has been assigned to rollback segment RBS3 and that the rollback space required for the transaction is $2606 - 2198 = 408$ bytes. If there are many active transactions, it may be more difficult to identify which rollback segment the transaction used so you may need to run these tests in a test environment.

V\$TRANSACTION

The V\$TRANSACTION view has one row for every active transaction. When a transaction starts a row is added to the view. When the transaction completes, it is removed from the view. The view can be used to find out the size of the transactions as well as the rollback segments with active transactions. This will be required if rollback segments need to be taken offline.

COLUMN NAME	DESCRIPTION
ADDR	Address of transaction state object
XIDUSN	Undo segment number. Join the V\$TRANSACTION view to the V\$ROLLSTAT or V\$ROLLNAME view using this column
XIDSLOT	Rollback segment header slot number
XIDSQN	Rollback segment header sequence number
UBAFIL	Undo block address (UBA) filenum where the transaction is currently writing to
UBABLK	UBA block number where the transaction is currently writing to
UBASQN	UBA sequence number where the transaction is currently writing to
UBAREC	UBA record number where the transaction is currently writing to
STATUS	Status of the transaction
START_TIME	Start time (wall clock)
START_SCNB	Start system change number (SCN) base
START_SCNW	Start SCN wrap

Continued

COLUMN NAME	DESCRIPTION
START_UEXT	Start extent number
START_UBAFIL	Start UBA file number
START_UBABLK	Start UBA block number
START_UBASQN	Start UBA sequence number
START_UBAREC	Start UBA record number
SES_ADDR	User session object address. Join this column of V\$TRANSACTION to the SADDR column of V\$SESSION to identify the user performing the transaction.
FLAG	Flag
SPACE	Yes, if a space transaction
RECURSIVE	Yes, if a recursive transaction
NOUNDO	Yes, if a no undo transaction
PTX	Yes, if parallel transaction, otherwise set to NO
PRV_XIDUSN	Previous transaction undo segment number
PRV_XIDSLT	Previous transaction slot number
PRV_XIDSQN	Previous transaction sequence number
PTX_XIDUSN	Rollback segment number of the parent XID
PTX_XIDSLT	Slot number of the parent XID
PTX_XIDSQN	Sequence number of the parent XID
DSCN_B	Dependent SCN base
DSCN_W	Dependent SCN wrap
USED_UBLK	Number of undo blocks used. This is the number of rollback segment blocks used by the transaction. This number is in blocks. The size of the database block is determined by the size of the INIT.ORA parameter DB_BLOCK_SIZE.
USED_UREC	Number of undo records used
LOG_IO	Logical IO performed by transaction
PHY_IO	Physical IO performed by transaction
CR_GET	Consistent gets performed by transaction
CR_CHANGE	Consistent changes performed by transaction

The following query can be run to list the users who currently have active transactions, as well as the size of the transaction.

```

SQL> SELECT SUBSTR(S.USERNAME,1,10) WHO
2 ,      SUBSTR(S.OSUSER,1,20) OS_WHO
3 ,      SUBSTR(R.NAME,1,8) "ROLLNAME"
4 ,      S.COMMAND
5 ,      S.SID
6 ,      S.SERIAL#
7 ,      T.USED_UBLK "ROLLBACK"
8 FROM    V$SESSION S, V$TRANSACTION T, V$ROLLNAME R
9 WHERE   T.SES_ADDR = S.SADDR
10 AND    R.USN = T.XIDUSN
11 /

```

WHO	OS_WHO	ROLLNAME	COMMAND	SID	SERIAL#	ROLLBACK
STUDENT	TODD\Administrator	RBS4		0	8	25

From this, you can see that the user **STUDENT** is currently the only user with an active transaction. The current size of this transaction is 1 rollback block and it is using rollback segment **RBS4**.

The following query identifies the rollback segment and the location in the rollback segment where the transaction is currently writing.

```

SQL> SELECT SUBSTR(S.USERNAME,1,10) WHO
2 ,      SUBSTR(R.NAME,1,8) "ROLLNAME"
3 ,      T.UBAFIL
4 ,      T.UBABLK
5 ,      T.USED_UBLK "ROLLBACK BLOCKS"
6 FROM    V$SESSION S, V$TRANSACTION T, V$ROLLNAME R
7 WHERE   T.SES_ADDR = S.SADDR
8 AND    R.USN = T.XIDUSN
9 /

```

WHO	ROLLNAME	UBAFIL	UBABLK	ROLLBACK BLOCKS
STUDENT	RBS4	2	2455	1

Maintaining Rollback Segments

Objective

Maintain rollback segments

Bringing rollback segments online

There are three ways that rollback segments can be brought online. They can be brought online when the instance is started by being referenced in the **ROLLBACK_SEGMENTS** parameter of the **INIT.ORA** parameter file, manually by the DBA through the **ALTER ROLLBACK SEGMENT** command or automatically by Oracle.

INIT.ORA method of bringing rollback segments online

The INIT.ORA parameter called `ROLLBACK_SEGMENTS` enables the DBA to identify all the rollback segments to be brought online when the instance is started. The rollback segments must exist and the instance must be restarted if the parameter is changed for the parameter to take effect. This method is certainly the most common one. It ensures that the correct rollback segments are brought online when the instance is started. By default, when rollback segments are created, they are offline. To ensure they are always online when the instance is started, make sure that new rollback segments are added to the list in the INIT.ORA PARAMETER file. Here is an example of the `ROLLBACK_SEGMENTS` parameter in the parameter file.

```
rollback_segments = ( RBS0, RBS1, RBS2, RBS3, RBS4, RBS5, RBS6 )
```

If new rollback segments are added, simply add them to this list, using commas to separate the names.

The maximum number of rollback segments that can be brought online is controlled by the `MAX_ROLLBACK_SEGMENTS` parameter. The default value is the greater of 30 or $(\text{TRANSACTIONS} / \text{TRANSACTIONS_PER_ROLLBACK_SEGMENT})$.

Since the DBA is responsible for adding rollback segments to this list, he or she can control the rollback segments that can be brought online. The DBA may want to create a large rollback segment for processing large batch jobs only. To ensure this, the DBA leaves the large rollback segment offline until needed. This can only be guaranteed by using this method.

ALTER ROLLBACK SEGMENT method of bringing rollback segments online

If a rollback segment is offline and the DBA does not want to shut down the database to bring it online, the DBA can issue the `ALTER ROLLBACK SEGMENT segment_name ONLINE` command to bring individual rollback segments online. Rollback segments are always created `OFFLINE` so this command is required to put them `ONLINE`. Many DBAs use this command to quickly put rollback segments online instead of shutting down the database, but remember to add this rollback segment to the INIT.ORA file so that the next time the instance is restarted, the newly created rollback segments will be online. If you manually `ONLINE` a rollback segment that is not in the INIT.ORA PARAMETER file and then restart the instance, you need to reissue the `ONLINE` command.

Having Oracle automatically bring rollback segments online

Oracle can also be used to bring rollback segments online automatically when the instance is started. There are two INIT.ORA PARAMETERS that control the number of rollback segments that should be brought online by Oracle. The calculation that

Oracle uses $(\text{TRANSACTIONS} / \text{TRANSACTIONS_PER_ROLLBACK_SEGMENT})$ where `TRANSACTIONS` is the number of concurrent transactions in the database and `TRANSACTIONS_PER_ROLLBACK_SEGMENT` is the desired number of concurrent transactions that each rollback segment should manage. If `TRANSACTIONS` is 100 and `TRANSACTIONS_PER_ROLLBACK_SEGMENT` is 5, then Oracle attempts to bring 20 non-SYSTEM Public rollback segments online. If there are 25 possible to choose from, it stops at 20. If there are only 15, it will bring all 15 online.

The only problem with this method is that the DBA does not control the rollback segments that are brought online. If the DBA has created a couple of large rollback segments for the large transactions, these may be brought online by Oracle. If you use this method, ensure that the large rollback segments are Private. Private rollback segments cannot be brought online by Oracle.

Using OEM to bring rollback segments online

The following steps outline the tasks involved in bringing rollback segments online with OEM.

STEP BY STEP: Bringing Rollback Segments Online or Offline using DBA Studio

1. Launch DBA Studio and select the Launch DBA Studio in Stand Alone option. Start ⇨ Programs ⇨ Oracle OEM (This name may be different on your installation.) ⇨ Database Administration ⇨ DBA Studio
2. Select the CERTDB database and enter the login information if required.
3. Select Storage.
4. Select Rollback Segments.
5. Double-click the rollback segment you want to modify.
6. On the General page, you can put rollback segments ONLINE or OFFLINE.
7. Click the Apply button when done.

Taking rollback segments offline

Rollback segments need to be taken offline to be dropped or to prevent transactions from using them. In order to take a rollback segment offline, no active transactions can be using the rollback segment. Run the following query to determine if there are any transactions currently using the rollback segment RBS1:

```
SQL> SELECT SUBSTR(S.USERNAME,1,10) WHO
2 , SUBSTR(S.OSUSER,1,20) OS_WHO
```

```

3 ,      SUBSTR(R.NAME,1,8) "ROLLNAME"
4 ,      S.COMMAND
5 ,      S.SID
6 ,      S.SERIAL#
7 ,      T.USED_UBLK "ROLLBACK"
8 FROM  V$SESSION S, V$TRANSACTION T, V$ROLLNAME R
9 WHERE T.SES_ADDR = S.SADDR
10 AND  R.USN = T.XIDUSN
11 AND  R.NAME = 'RBS1'
12 /

```

WHO	OS_WHO	ROLLNAME	COMMAND	SID	SERIAL#	ROLLBACK
STUDENT	TODD\Administrator	RBS1		0	8	17

From this query, you can see that there is one user, STUDENT, who has an active transaction on RBS1. You can wait for the user to complete the transaction or you may need to kill the user session if there are no other means of ending the transaction and the user is not available.

When the query is run again, you can see that there are no transactions using this rollback segment. You can now take the rollback segment offline.

```

SQL> SELECT SUBSTR(S.USERNAME,1,10) WHO
2 ,      SUBSTR(S.OSUSER,1,20) OS_WHO
3 ,      SUBSTR(R.NAME,1,8) "ROLLNAME"
4 ,      S.COMMAND
5 ,      S.SID
6 ,      S.SERIAL#
7 ,      T.USED_UBLK "ROLLBACK"
8 FROM  V$SESSION S, V$TRANSACTION T, V$ROLLNAME R
9 WHERE T.SES_ADDR = S.SADDR
10 AND  R.USN = T.XIDUSN
11 AND  R.NAME = 'RBS1'
12 /

```

no rows selected

The syntax for taking rollback segments offline is ALTER ROLLBACK SEGMENT *segment_name* OFFLINE

Before running the query, the status in DBA_ROLLBACK_SEGS for rollback segment RBS1 is ONLINE. After the rollback segment is taken offline, the status changes to OFFLINE.

```

SQL> SELECT SEGMENT_NAME,STATUS
2 FROM  DBA_ROLLBACK_SEGS

```

```

3 WHERE SEGMENT_NAME = 'RBS1';

SEGMENT_NAME          STATUS
-----
RBS1                   ONLINE

SQL> ALTER ROLLBACK SEGMENT RBS1 OFFLINE;

Rollback segment altered.

SQL> SELECT SEGMENT_NAME,STATUS
2 FROM   DBA_ROLLBACK_SEGS
3 WHERE  SEGMENT_NAME = 'RBS1';

SEGMENT_NAME          STATUS
-----
RBS1                   OFFLINE

```

If there are active transactions against the rollback segment when it is taken offline, the status becomes **PENDING OFFLINE** — similar to being **OFFLINE** in that no transactions can use the rollback segment but the rollback segment cannot be dropped until the transactions on using the rollback segment complete. The status in `DBA_ROLLBACK_SEGS` will still be **ONLINE**, however, the rollback segment in `V$ROLLSTAT` will have a status of **PENDING OFFLINE**.

```

SQL> SELECT SUBSTR(S.USERNAME,1,10) WHO
2 ,        SUBSTR(S.OSUSER,1,20) OS_WHO
3 ,        SUBSTR(R.NAME,1,8) "ROLLNAME"
4 ,        S.COMMAND
5 ,        S.SID
6 ,        S.SERIAL#
7 ,        T.USED_UBLK "ROLLBACK"
8 FROM     V$SESSION S, V$TRANSACTION T, V$ROLLNAME R
9 WHERE    T.SES_ADDR = S.SADDR
10 AND     R.USN = T.XIDUSN
11 /

WHO      OS_WHO          ROLLNAME  COMMAND  SID  SERIAL#  ROLLBACK
-----
STUDENT TODD\Administrator RBS0          0      8      17      1

```

This indicates that rollback segment **RBS0** has an active transaction.

```

SQL> SELECT RN.NAME "Name"
2 ,        RS.XACTS "TRANS"
3 ,        STATUS
4 FROM     V$ROLLNAME RN,
5          V$ROLLSTAT RS
6 WHERE    RS.USN = RN.USN
7 /

```

Name	TRANS	STATUS
SYSTEM	0	ONLINE
RBS0	1	ONLINE
RBS2	0	ONLINE
RBS3	0	ONLINE
RBS4	0	ONLINE
RBS5	0	ONLINE
RBS6	0	ONLINE

7 rows selected.

All the rollback segments are ONLINE. Now issue the OFFLINE command.

```
SQL> ALTER ROLLBACK SEGMENT RBS0 OFFLINE;
```

Rollback segment altered.

```
SQL> SELECT RN.NAME "Name"
2 ,      RS.XACTS "TRANS"
3 ,      STATUS
4 FROM   V$ROLLNAME RN,
5        V$ROLLSTAT RS
6 WHERE  RS.USN = RN.USN
7 /
```

Name	TRANS	STATUS
SYSTEM	0	ONLINE
RBS0	1	PENDING OFFLINE
RBS2	0	ONLINE
RBS3	0	ONLINE
RBS4	0	ONLINE
RBS5	0	ONLINE
RBS6	0	ONLINE

7 rows selected.

Once the active transaction using rollback segment RBS0 completes, the status should be changed to OFFLINE.

```
SQL> SELECT SUBSTR(S.USERNAME,1,10) WHO
2 ,      SUBSTR(S.OSUSER,1,20) OS_WHO
3 ,      SUBSTR(R.NAME,1,8) "ROLLNAME"
4 ,      S.COMMAND
5 ,      S.SID
6 ,      S.SERIAL#
7 ,      T.USED_UBLK "ROLLBACK"
8 FROM   V$SESSION S, V$TRANSACTION T, V$ROLLNAME R
9 WHERE  T.SES_ADDR = S.SADDR
```

```

10 AND      R.USN = T.XIDUSN
11 /

```

no rows selected

```

SQL> SELECT RN.NAME "Name"
       ,      RS.XACTS "TRANS"
       ,      STATUS
       FROM    V$ROLLNAME RN,
       ,      V$ROLLSTAT RS
       WHERE   RS.USN = RN.USN
       /

```

Name	TRANS	STATUS
SYSTEM	0	ONLINE
RBS0	0	PENDING OFFLINE
RBS2	0	ONLINE
RBS3	0	ONLINE
RBS4	0	ONLINE
RBS5	0	ONLINE
RBS6	0	ONLINE

7 rows selected.

You can see that even though there are no active transactions, as indicated by the TRANS column, the status remains PENDING OFFLINE. However, if you query DBA_ROLLBACK_SEGS, you see that the status is now OFFLINE.

Changing rollback segment storage parameters

You can change storage parameter for rollback segments by using the ALTER ROLLBACK SEGMENT command. It is generally used to change the value of MAXEXTENTS or OPTIMAL when transaction volume and size have changed since the rollback segment was created. You can also change the value of NEXT and MINEXTENTS. Here is the syntax:

```

ALTER ROLLBACKK SEGMENT segment_name
  [STORAGE ( [NEXT integer [K|M]]
             [MINEXTENTS integer]
             [MAXEXTENTS {integer|UNLIMITED}]
             [OPTIMAL {integer[K|M]|NULL}]
           )
]

```

Changing rollback segment storage parameters using OEM

The following outlines the steps involved in changing storage parameters for rollback segments using OEM.

STEP BY STEP: Modifying Rollback Segments with DBA Studio

1. Launch DBA Studio and select the Launch DBA Studio in Stand Alone option. Start ⇨ Programs ⇨ Oracle OEM (This name may be different on your installation.) ⇨ Database Administration ⇨ DBA Studio.
2. Select the CERTDB database and enter the login information if required.
3. Select Storage.
4. Select Rollback Segments.
5. Double-click on the rollback segment you want to modify.
6. On the General page, you can put rollback segments ONLINE or OFFLINE. On the Storage page, you can change the values of Next Extent Size, Optimal, and Maximum Extent.
7. Click the Apply button when done.

Shrinking rollback segments

When rollback segments extend beyond their created size, one of two things will happen. First, if OPTIMAL is set for the rollback segment, Oracle attempts to deallocate space back to OPTIMAL. If a rollback segment has grown to 15MB and OPTIMAL is set to 10MB, Oracle tries to shrink or deallocate space, returning the rollback segment to its optimal size of 10MB. The extents being deallocated must not have active transactions. If they do, Oracle will not deallocate them. There must also be at least two contiguous free extents for Oracle to deallocate one of them. The space is not deallocated immediately after the transaction that caused it to extend completes. Rather, the next time the pointer in the rollback segment header moves to the next extent, it will check to see if space can be deallocated. If it cannot deallocate completely to OPTIMAL, it will do as much as it can. The extents that are deallocated are the oldest extents in the rollback segment. The oldest extents are the least likely to be required for read consistency.

If OPTIMAL is not set, Oracle does not shrink rollback segments automatically. The DBA must do this manually. The ALTER ROLLBACK SEGMENT command has a special feature for performing this task. Here is the syntax:

```
ALTER ROLLBACK SEGMENT rollback_segment_name
  SHRINK [ TO integer [ K|M]]
```

If only the key word SHRINK is specified, Oracle attempts to shrink the rollback segment back to OPTIMAL. If a size is specified, then Oracle attempts to shrink the rollback segment back to that size. The same rules apply to this method as they do for the automatic one. Extents with active transactions cannot be deallocated and the space being deallocated must have a minimum of two free extents. The oldest extents are always deallocated first.

The V\$ROLLSTAT view can be queried to determine the size of the rollback segment. After the ALTER ROLLBACK SEGMENT command completes, verify the results by querying V\$ROLLSTAT. If Oracle cannot deallocate any space, it does not return an error message so you must verify yourself.

In the following example, a large transaction was run against rollback segment RBS1. You can see from the following query that RBS1 has extended and the current size is greater than the value of OPTIMAL.

```
SQL> SELECT RN.NAME "Name"
 2    ,      RS.EXTENTS
 3    ,      RS.RSSIZE "SIZE"
 4    ,      RS.OPTSIZE "OPTIMAL"
 5    ,      RS.HWMSIZE
 6    ,      RS.XACTS "TRANS"
 7    ,      STATUS
 8 FROM    V$ROLLNAME RN,
 9         V$ROLLSTAT RS
10 WHERE   RS.USN = RN.USN
11 /
```

Name	EXTENTS	SIZE	OPTIMAL	HWMSIZE	TRANS	STATUS
SYSTEM	5	401408		401408	0	ONLINE
RBS1	17	8904704	4194304	8904704	1	ONLINE
RBS2	8	4186112	4194304	4186112	0	ONLINE
RBS3	8	4186112	4194304	4186112	0	ONLINE
RBS4	8	4186112	4194304	4186112	0	ONLINE
RBS5	8	4186112	4194304	4186112	0	ONLINE
RBS6	8	4186112	4194304	4186112	0	ONLINE

7 rows selected.

Now, because the rollback segment size is greater than optimal, you can expect the rollback segment to shrink back to optimal when the transaction using rollback segment RBS1 completes and pointer in the rollback segment header is advanced to the next extent.

```
SQL> SELECT RN.NAME "Name"
 2    ,      RS.EXTENTS
 3    ,      RS.RSSIZE "SIZE"
 4    ,      RS.OPTSIZE "OPTIMAL"
 5    ,      RS.HWMSIZE
 6    ,      RS.XACTS "TRANS"
 7    ,      STATUS
 8 FROM    V$ROLLNAME RN,
 9         V$ROLLSTAT RS
10 WHERE   RS.USN = RN.USN
11 /
```

Name	EXTENTS	SIZE	OPTIMAL	HWMSIZE	TRANS	STATUS
------	---------	------	---------	---------	-------	--------

```

SYSTEM      5      401408      401408      0 ONLINE
RBS1        8      4186112    4194304    8904704    0 ONLINE
RBS2        8      4186112    4194304    4186112    0 ONLINE
RBS3        8      4186112    4194304    4186112    0 ONLINE
RBS4        8      4186112    4194304    4186112    0 ONLINE
RBS5        8      4186112    4194304    4186112    0 ONLINE
RBS6        8      4186112    4194304    4186112    0 ONLINE

```

7 rows selected.

You can see that the `HWMSIZE` for `RBS1` is 8,904,704 bytes but the size has returned to 4,186,112 bytes. Oracle automatically shrunk rollback segment `RBS1`. You may also want to look at the number of `EXTENDS` and `SHRINKS` for the rollback segments—two entries that may indicate a need for increasing the value of `OPTIMAL`.

```

SQL> SELECT RN.NAME "Name"
       2 ,      RS.GETS
       3 ,      RS.WAITS
       4 ,      RS.SHRINKS
       5 ,      RS.WRAPAS
       6 ,      RS.EXTENDS
       7 FROM   V$ROLLNAME RN,
       8        V$ROLLSTAT RS
       9 WHERE  RS.USN = RN.USN
      10 /

```

Name	GETS	WAITS	SHRINKS	WRAPS	EXTENDS
SYSTEM	258	0	0	0	0
RBS1	103847	1	1	18	9
RBS2	232	0	0	0	0
RBS3	360	0	0	2	0
RBS4	225	0	0	0	0
RBS5	348	0	0	0	0
RBS6	245	0	0	0	0

From this query, you can see that `RBS1` had a total of 9 extends and 1 shrink, caused by the long transactions. If there are multiple `SHRINKS`, you may consider increasing the value of `OPTIMAL`.

Dropping rollback segments

If you want to change the storage settings for a rollback segment because you are not happy with the number of `WRAPS`, `SHRINKS`, or `EXTENDS`, you may find it necessary to drop and re-create the rollback segment. This is the only way to change the size of the `INITIAL` and `MINEXTENTS` parameters. You may also just not need a rollback segment anymore. They do, after all, take up space.

In order to drop a rollback segment, the rollback segment must be offline. To take a rollback segment offline there must be no active transactions on it. Once offline, issue the following command:

```
DROP ROLLBACK SEGMENT rollback_segment_name;
```

If the rollback segment is not offline, an error message will be returned.



When dropping a rollback segment, make sure you remove it from the INIT.ORA PARAMETER file. If you do not, an error will be raised the next time you attempt to start it.

Troubleshooting Rollback Segments

Objective

Troubleshoot common rollback segment problems

Troubleshooting rollback segment problems is an unfortunate reality for most DBAs. Thankfully, Oracle provides excellent information in the data dictionary views to resolve most of the problems. This section looks at some of the more common problems.

Insufficient space for transactions

When a transaction starts it must complete on the same rollback segment. Transactions cannot use multiple rollback segments if the rollback segment runs out of space. Therefore, it is critical that the rollback segments have sufficient space to store all the transaction information or else the transaction will fail. When the transaction fails, it rolls back. The issue at hand is that large transactions may take several hours to run out of rollback space. Once out of space, the rollback may take several hours as well. If this happens, several hours can elapse with nothing being accomplished, a situation that tends to displease users.

Rollback segments run out of room when one of two things happens: Either the maximum number of extents for the rollback segment has been reached, determined by the parameter MAXEXTENTS (ORA-01628) or there is no more space left in the tablespace holding the rollback segment (ORA-01650).

Solution

If the problem occurred because of insufficient space in the tablespace, consider making the tablespace bigger. You can add a new datafile or increase the size of the existing one. You also want to ensure that OPTIMAL is set for all rollback segments. If OPTIMAL is not set, extra space allocated to rollback segments will not be deallocated. If the transaction failed because the MAXEXTENTS parameter was reached for the rollback segment, you can increase the value of MAXEXTENTS. If MAXEXTENTS is at its limit, you need to drop and re-create the rollback segment with larger extent sizes. Larger extent sizes require fewer extents.

If the problem still persists, consider rewriting the large transaction to many smaller ones or create a large rollback segment, large enough to hold all the transaction information, and explicitly assign the transaction to this rollback segment using the SET TRANSACTION command.

Read consistency error

When a user starts a query or transaction, Oracle guarantees that any changes made by other users that are not committed when the statement begins or changes made after the statement begins and committed are not seen by the query. Queries cannot read rows that have changed since the query started. When a user starts a query, Oracle determines the current System Change Number (SCN) and ensures that the query will not see rows that have changed after this SCN number. Since the SCN number is assigned when transactions are committed, this model works for both transactions that start before the query and have since committed and new transactions that started after the query started.

Since long running queries can take several minutes to complete, it is likely that some of the rows are going to change before the query finishes. If the query encounters a row that has an SCN number greater than the SCN number that the query got when it started, it knows that that row has changed and cannot be read. For these rows, Oracle goes to the rollback segment that holds the change information and reconstructs a read consistent image of the row, even if the transaction has committed. If the transaction information exists in the rollback segment, it is used. The problem happens when the transaction information is not there. Rollback segments work in a circular fashion overwriting the oldest committed data when space is needed. This means that the information needed to reconstruct the read consistent image is gone. When this occurs, Oracle returns an ORA-1555 SNAPSHOT TOO OLD error. This only happens when

- ♦ The transaction slot in the rollback segment header has been reused.
- ♦ The before-image in the rollback segment has been overwritten by another transaction.

Solution

This error happens less frequently on large rollback segments. So increasing the size of the rollback segments helps reduce the frequency of the error. Adding more rollback segments reduces the error occurring because of transaction slots in the rollback segment headers being overwritten. With more rollback segments, there are more transaction slots.

Blocking session

When a user starts a transaction, that transaction is assigned to a rollback segment. The space used cannot be overwritten until the transaction completes. This is not a problem until Oracle attempts to position the pointer for the rollback segment into the extent that this transaction occupies. The pointer cannot be positioned into an

extent that has an active transaction. When it finds that the next extent has an active transaction, it is forced to create a new extent. Oracle cannot skip over an extent with an active transaction even if there is free space in other extents. This is considered a blocking session.

This usually happens when transactions have been idle. If a user starts a transaction and goes to lunch or is busy doing something else, that transaction stays active. There is no such thing as a transaction timeout in Oracle. So rollback segments are forced to grow even though space is available. This causes space to be wasted and users to suffer through their transactions performing the dynamic extension to create the new extents. This has a negative effect on performance.

Solution

Educating users is the best way to reduce the chances of this happening. Users should always complete transactions once they have started. This error is also less likely to happen when rollback segments are larger. If a rollback segment is 20MB in size and the blocking session has a transaction using one block on extent 1, then 19.9MB+ of rollback space can be used before the session is a blocking session. If the rollback segments are 1MB in size and the user has the same transaction, then only 992KB of space needs to be used in the rollback segment before the session is a blocking session.

If there are blocking sessions, you can run the following query to determine which session is the blocking one:

```
SQL> SELECT      S.SID
 2 ,            S.SERIAL#
 3 ,            S.USERNAME
 4 ,            S.OSUSER
 5 ,            T.START_TIME
 6 ,            T.XIDUSN
 7 FROM          V$SESSION S, V$TRANSACTION T, V$ROLLSTAT R
 8 WHERE         S.SADDR = T.SES_ADDR
 9 AND          T.XIDUSN = R.USN
10 AND          ((R.CUREXT = T.START_UEXT-1) OR
11              ((R.CUREXT = R.EXTENTS-1) AND T.START_UEXT=0));
```

SID	SERIAL#	USERNAME	OSUSER	START_TIME	XIDUSN
9	68	STUDENT	TODD\Administrator	04/18/01 14:02:24	4

From this query, you can see that the session with SID=9 is a blocking session. To fix this, you can either ask the user to complete the transaction or kill the user session. The following command terminates the user session:

```
ALTER SYSTEM KILL SESSION '9,68';
```

Rerun the query and the blocking session should be gone.

```
SQL> SELECT      S.SID
 2  ,            S.SERIAL#
 3  ,            S.USERNAME
 4  ,            S.OSUSER
 5  ,            T.START_TIME
 6  ,            T.XIDUSN
 7  FROM        V$SESSION S, V$TRANSACTION T, V$ROLLSTAT R
 8  WHERE       S.SADDR = T.SES_ADDR
 9  AND         T.XIDUSN = R.USN
10  AND         ((R.CUREXT = T.START_UEXT-1) OR
11             ((R.CUREXT = R.EXTENTS-1) AND T.START_UEXT=0));
```

no rows selected

Error in taking a tablespace offline

If a tablespace has any rollback segment created on it, those rollback segments must be taken offline in order to take the tablespace offline. Rollback segments can only be taken offline if there are no active transactions using the rollback segment. The error returns if you try to take a tablespace offline with online rollback segments is ORA-01546.

Solution

Rollback segments should always be on their own tablespace. This prevents this problem. If, however, you run into the problem, follow these steps:

1. Query `DBA_ROLLBACK_SEGS` to get a list of all the rollback segments located on the tablespace you are trying to take offline.
2. Take all the rollback segments on that tablespace offline.
3. Check the status of the rollback segments in `V$ROLLSTAT` to ensure that they are all `OFFLINE`.
4. If any of the rollback segments have a status of `PENDING OFFLINE`, then you need to query the `V$TRANSACTION` and `V$SESSION` views to determine which user has an active transaction on the rollback segment.
5. Have the user complete the transaction or kill the session.
6. Take the tablespace offline.



Rollback segments should always be stored in tablespace exclusively used for rollback segments. Doing so eliminates this error from ever happening unless it is the rollback segment tablespace you want to take offline.

Planning Rollback Segments

Objective

Planning the number and size of rollback segments

Online transaction processing environments

In an Online Transaction Processing Environment (OLTP), throughput is everything. Transactions must be processed as quickly as possible. The header of a rollback segment contains the transaction information for all transactions using the rollback segment. This header needs to be updated every time a transaction starts and completes. Therefore, if many transactions are trying to simultaneously use the rollback segment header, there may be contention. Such contention can have a negative impact on performance.

In an OLTP environment, you need to create many rollback segments. The ideal number depends on many factors, such as the number of concurrent transactions and the number of users. Ideally, you want to have enough rollback segments so that the column `WAITS` in `V$ROLLSTAT` is static. That is, no users are experiencing waits for the rollback segment header. Start with a ratio of four transactions per rollback segment. If you estimate 100 concurrent transactions, then you would want to have 25 rollback segments. Once the rollback segments have been created and the database has been running for some time, query `V$ROLLSTAT` and verify that the `WAITS` column is not increasing.

The size of the rollback segments is also important. Rollback segments should be made up of a minimum of 20 extents. This is a recommendation from Oracle based upon some extensive testing. The size of the extents should be equal to the size of the average transactions. The average transaction size can be obtained by querying the `V$TRANSACTION` view after running your average transaction. A `DELETE` is much more expensive than an `INSERT`. The undo of a `DELETE` is the entire row because that is the before image. The undo of an `INSERT` is just a `ROWID`. If the average transaction size is 16KB, the rollback segment should be made up of 20 16KB extents. You can add more extents by increasing the value of `OPTIMAL` if you notice that the column `EXTENDS` in `V$ROLLSTAT` is increasing.

You should try to avoid creating small rollback segments. The smaller the rollback segment the more likely it is to dynamically extend. So why not just create large rollback segments to prevent it from ever needing to extend? There are two considerations here: First is that rollback segments use space, which is wasted with large rollback segments. Second, if the rollback segments are large, the likelihood of getting cache hits on rollback segment blocks is small. The rollback segment blocks need to be read into the database buffer cache when they are used. The more there are, the less likely you are to find them in the cache when needed.



Most DBAs use many large rollback segments primarily to prevent dynamic extension. While the larger rollback segments result in better cache hits, the cost of dynamic extensions is greater than the IO costs of reading in rollback blocks.

Data warehousing environments

Data warehousing environments are characterized as being primarily query-only. If the database is only running queries, there is no need for rollback segments. However, most data warehouses are not that strictly defined. Most have some batch processes that load data in the database. These batch processes are usually large transactions. The rollback segments required in this environment is different from that of an OLTP. Here, the concern is not of rollback segment header contention but simply space. You do not want these large transactions to run out of rollback space so you ensure that they are large — large enough to hold all the transaction information. If the rollback segment runs out of space, the transaction rolls back, so create a few very large rollback segments. Make sure that MAXEXTENTS is high and that the extent sizes are large.

Key Point Summary

In preparing for the Oracle8i DBA: Architecture and Administration exam, please keep these points in mind regarding Rollback Segments:

- ♦ Rollback segments are used by Oracle for Read Consistency by storing before images of rows that have changed.
- ♦ Rollback segments are used by Oracle to rollback transactions or undo transactions requested by the user or automatically in the event of a statement failure.
- ♦ Rollback segments are used by Oracle for transaction recovery in the event of an instance failure.
- ♦ Rollback segments are vital for transactions so it is important to ensure that enough rollback segments have been created and that they are created the appropriate size. The data dictionary views V\$ROLLNAME, V\$ROLLSTAT, V\$TRANSACTION, V\$SESSION, and DBA_ROLLBACK_SEGS contain the necessary information to make those decisions.
- ♦ Rollback segments should be created in tablespaces that only contain rollback segments. It makes tablespaces easier to manage when they do not contain rollback segments. A tablespace with an active rollback segment cannot be taken offline.
- ♦ A rollback segment should always have the same size extents.
- ♦ PCTINCREASE has no meaning for rollback segments.
- ♦ OPTIMAL is a storage parameter used to control the “optimal size” for a rollback segment. If a rollback segment grows beyond the value of OPTIMAL, then Oracle will automatically deallocate space from the rollback segment back to the value of OPTIMAL. If OPTIMAL is not set, then the DBA must manually deallocate the space.

- ♦ Specify the rollback segments you want online for your database in the `ROLLBACK_SEGMENTS` parameter of the `INIT.ORA` file. Otherwise, they need to be brought online manually with an `ALTER ROLLBACK SEGMENT segment_name ONLINE` command.
- ♦ If large transactions are being run ensure that the rollback segments are large enough to hold the entire transaction. If a transaction runs out of space in a rollback segment, the transaction fails and rolls back. To prevent this, it may be necessary to create one or two large rollback segments and assign specific transactions to them using the `SET TRANSACTION USE ROLLBACK SEGMENT rollback segment name`. If transactions are not assigned explicitly by the user to a rollback segment then Oracle assigns the transaction to a rollback segment. This may cause the large transaction to be assigned to one of the standard rollback segments and not one of the large rollback segments.
- ♦ The more concurrent transactions in the database the more rollback segments that should exist. Oracle's recommendation for small transactions is 1 rollback segment for every 4 concurrent transactions. Again, you should check the data dictionary views to ensure that there is no contention for the rollback segments.
- ♦ Users should code short transactions and complete them as quickly as possible by either committing or rolling back their work. This prevents transactions from blocking the use of free space in a rollback segment. If an active transaction exists then Oracle cannot skip the extent with the active transaction and this can lead to dynamic extension of rollback segments.
- ♦ If rollback segments are too small or there are too few of them, this can lead to Snapshot Too Old errors. If this happens, increase the size or the existing rollback segments or add more rollback segments.



STUDY GUIDE

Now that you have learned about rollback segments, you should test your understanding by reviewing the assessment questions and performing the exercises below.

Assessment Questions

1. What are the three main uses of rollback segments? Select three answers.
 - A. read consistency
 - B. transaction rollback
 - C. transaction recovery
 - D. database recovery
 - E. transaction logging
2. What is the minimum number of extents that every rollback segment must have?
 - A. 0
 - B. 1
 - C. 2
 - D. 10
 - E. 20
3. What extent storage parameter cannot be specified for rollback segments?
 - A. INITIAL
 - B. NEXT
 - C. OPTIMAL
 - D. MINEXTENTS
 - E. PCTINCREASE

4. Which rollback segment parameter should be set to ensure that when extra space is required that it is automatically deallocated whenever possible.
 - A. MINEXTENTS
 - B. MAXEXTENTS
 - C. PCTFREE
 - D. OPTIMAL
 - E. PCTUSED

5. When will rollback segments shrink?
 - A. When not being used
 - B. When the database is restarted
 - C. When the rollback segment grows in size beyond the value of MAXEXTENTS and OPTIMAL is set
 - D. When the rollback segment grows in size beyond the value of OPTIMAL and OPTIMAL is set
 - E. Rollback segments never shrink

6. Which command is used to take rollback segment RBS1 OFFLINE?
 - A. ALTER DATABASE ROLLBACK SEGMENT RBS1 OFFLINE
 - B. ALTER TABLESPACE RBS1 OFFLINE
 - C. ALTER ROLLBACK SEGMENT RBS1 OFFLINE IMMEDIATE
 - D. ALTER ROLLBACK SEGMENT OFFLINE
 - E. ALTER ROLLBACK SEGMENT RBS1 OFFLINE

7. If, after taking a rollback segment offline, you query the V\$ROLLSTAT view and see that the status of the rollback segment is PENDING OFFLINE, what do you think is the most likely cause of this status?
 - A. Oracle has not had a chance to take it offline and is waiting.
 - B. There are active transactions using the rollback segment.
 - C. You do not have permission to take the rollback segment offline.
 - D. You are not the owner of the rollback segment.
 - E. That is the status of an OFFLINE rollback segment.

8. How can you select the rollback segments that you want brought ONLINE every time the instance starts?
- A. Place the names of the rollback segments in the ORASTART.ORA file.
 - B. Place the names of the rollback segments in the ROLLBACK_SEGMENTS parameter of INIT.ORA PARAMETER FILE.
 - C. ONLINE rollback segments will always be ONLINE when the instance is started.
 - D. Place the names of the rollback segments in the ONLINE_ROLLBACK_SEGMENTS parameter of INIT.ORA PARAMETER FILE.
 - E. Place the names of the rollback segments in the ONLINE_ROLLBACKS parameter of INIT.ORA PARAMETER FILE.
9. What data dictionary view contains a row for every active transaction in the database?
- A. V\$TRANSACTION
 - B. V\$ACTIVE_TRANSACTION
 - C. DBA_TRANSACTIONS
 - D. DBA_TRANS_INFO
 - E. DBA_ACTIVE_TRANSACTIONS
10. What data dictionary view contains a row for every rollback segment in the database?
- A. V\$ROLLNAME
 - B. V\$ROLLSTAT
 - C. V\$TRANSACTION
 - D. DBA_ROLLBACK_SEGMENTS
 - E. DBA_ROLLBACK_SEGS

Scenarios

1. You are concerned about the size of your rollback segment tablespace. It has grown by 500MB. After further investigation of the V\$ROLLSTAT view, you find that all the rollback segments have EXTENDED and none of them have any SHRINKS. There are no other problems with the rollback segments and no transactions have run out of room or failed because of the rollback segments.
- A. What should be done to ensure that the space used by the rollback segments is deallocated after transactions complete?
 - B. What considerations should be given about the fact that all the rollback segments have extended?

2. You have been hired as a consultant by HANNAH and JAKE enterprises to investigate some transaction failures due to insufficient rollback space. The transactions that are failing are large batch processes that UPDATE and DELETE thousands of rows in the database.
 - A. What recommendations would you make to resolve the transaction failures?

Lab Exercises

Lab 10-1 Creating Rollback Segments

1. Using SQLPLUS line mode (sqlplus), connect to your instance as a user with SYSDBA privileges.
2. Using SQLPLUS line mode (sqlplus), connect another, second, session to your instance as user STUDENT/ORACLE.
3. Create a tablespace called RBS_TEST without specifying tablespace storage defaults. The size of the tablespace should be two megabytes and the datafile should be called RBS_TEST.DBF and located in the CERTDB\DISK5 directory.
4. Create a new rollback segment called RBS7 in the tablespace RBS_TEST. Set the extent size to 16K for all extents make sure that only two extents are initially created. Also, set the maximum number of extents to 10.
5. Query the data dictionary to verify that the rollback segment has been created, the status and that the storage parameters are correct.
6. Query the data dictionary to verify that two extents have been created for rollback segment RBS7.
7. Bring the rollback segment ONLINE manually by using the ALTER ROLLBACK command. You do not need to shut down the database to do this.
8. Verify that it is now ONLINE by querying the V\$ROLLSTAT view.

Lab 10-2 Managing Rollback Segments

1. Using SQLPLUS line mode (sqlplus), connect to your instance as a user with SYSDBA privileges.
2. Using SQLPLUS line mode (sqlplus), connect another, second, session to your instance as user STUDENT/ORACLE.
3. In the SQLPLUS session started in exercise 2, verify you are connected as the user STUDENT and create a table in the USER_DATA tablespace using the following command:

```
CREATE TABLE DB_TABLES
TABLESPACE USERS
AS
SELECT *
FROM DICT;
```

4. Use the SET TRANSACTION command to start a transaction using rollback segment RBS7.
5. Issue the following command to delete 200 rows from the DB_TABLES table:

```
DELETE FROM DB_TABLES
WHERE ROWNUM < 201;
```

6. Using the SQLPLUS session created in exercise 1, verify that a transaction was started for the user STUDENT and that it is using rollback segment RBS1. Verify you are in the correct session first by typing in the SHOW USER command in SQLPLUS.
7. Using the SQLPLUS session created in exercise 1, take the rollback segment RBS7 OFFLINE.
8. Did the command succeed?
9. Query the V\$ROLLSTAT view to get the status of the rollback segment RBS7.
10. Why is it PENDING OFFLINE?
11. In the SQLPLUS session started in exercise 2, complete the transaction by issuing a commit.
12. Using the SQLPLUS session created in exercise 1 query the V\$ROLLSTAT view to get the status of the rollback segment.
13. Why is the status still PENDING OFFLINE?
14. Query the DBA_ROLLBACK_SEGS view and check the status of the rollback segment RBS7.

Lab 10-3 Managing Rollback Segments

1. Using SQLPLUS line mode (sqlplus), connect to your instance as a user with SYSDBA privileges.
2. Using SQLPLUS line mode (sqlplus), connect another, second, session to your instance as user STUDENT/ORACLE.
3. Using the SQLPLUS session created in exercise 1, bring rollback segment RBS7 online. If the ONLINE command fails, then issue the ALTER ROLLBACK SEGMENT RBS7 OFFLINE command followed by the ALTER ROLLBACK SEGMENT ONLINE command. (The rollback segment was created in Lab 10-1).
4. Using the SQLPLUS session created in exercise 2, drop the table DB_TABLES. This table was created in the previous exercise.

5. In the SQLPLUS session started in exercise 2, verify you are connected as the user STUDENT and create a table in the USER_DATA tablespace using the following command.

```
CREATE TABLE DB_TABLES
TABLESPACE USERS
AS
  SELECT *
  FROM   DICT;
```

6. Issue the following command three times to insert more data into the DB_TABLES table. Make sure it is run three times. When done, there should be 2,424 rows in the DB_TABLES when you are done.

```
INSERT INTO DB_TABLES
SELECT * FROM DB_TABLES;
```

7. Complete the transaction by issuing a COMMIT.
8. Use the SET TRANSACTION command to start a transaction using rollback segment RBS7.
9. Delete all the rows in the DB_TABLES table.
10. What happened and why?
11. Using the SQLPLUS session created in exercise 1, modify rollback segment RBS7 so that it is allowed to grow beyond 10 extents. Let it grow to 100 extents by setting the MAXEXTENTS parameter to 100.
12. Using the SQLPLUS session created in exercise 1, reissue the delete command to delete all the rows in the DB_TABLES table?
13. What happened now and why?
14. Using the SQLPLUS session created in exercise 1, make the tablespace RBS_TEST bigger by resizing the datafile to 5MB.
15. Using the SQLPLUS session created in exercise 1, alter rollback segment RBS7 so that MAXEXTENTS is set to 256.
16. Using the SQLPLUS session created in exercise 2, try to delete all the rows from the DB_TABLES table.
17. Did the command complete successfully?
18. Commit the transaction.
19. Using the SQLPLUS session created in exercise 1, query the V\$ROLLSTAT view and find the size, the number shrinks, and the number of extends for rollback segment RBS7.
20. Deallocate space in the rollback segment RBS7 keeping 256KB of space.
21. Using the SQLPLUS session created in exercise 1, query the V\$ROLLSTAT view and find the size of rollback segment RBS7 after deallocating or shrinking the space from in it.

Answers to Chapter Questions

Chapter Pre-Test

1. The three main uses of rollback segments are transaction rollback, read consistency, and transaction recovery.
2. Rollback segments contain the before image of a row that is changing. DML statements change rows. The before image of a DELETE is the entire row; the before image of an INSERT is the ROWID; and the before image of an UPDATE is the before values of the column or columns that are changing. The rollback segment headers contain transaction information for all transactions currently assigned to it. This transaction information includes the status of the transaction — that is, whether it has been committed or not.
3. The most common cause of the SNAPSHOT TOO OLD error is insufficient rollback space. This error happens when long queries try to acquire a read consistent image of a block that changed since the query started. If the information is unavailable because the rollback space has been overwritten, Oracle is forced to return the error because it cannot guarantee a read consistent image for the query. Adding more rollback segments or increasing the size of the rollback segments should help minimize the frequency of the SNAPSHOT TOO OLD error.
4. Multiple rollback segments help reduce contention for rollback segments, especially the rollback segment headers. The headers are the transaction tables and are always busy. By adding more rollback segments you help reduce multiple processes trying to read and write to the header. Multiple rollback segments can also help reduce the frequency of the SNAPSHOT TOO OLD error.
5. ONLINE rollback segments are ones that are available for use. This is the normal status for rollback segments. Rollback segments that are OFFLINE cannot be used by transactions. This is usually the status of a rollback segment under maintenance or about to be dropped.
6. Since rollback segments have a tendency to grow, Oracle has a special parameter for rollback segments called OPTIMAL. The value of OPTIMAL is used when Oracle is deallocating space used by transactions that forced the rollback segment to grow.
7. The DBA_ROLLBACK_SEGS data dictionary view contains a row for every rollback segment in the database. The V\$ROLLSTAT has a row for every ONLINE or OFFLINE PENDING rollback segment.
8. The V\$TRANSACTION view contains a row for every active transaction. There is a column called XIDUSN that links a transaction to a rollback segment.
9. The SET TRANSACTION USE ROLLBACK SEGMENT LARGE_RBS; command assigns the transaction to rollback segment LARGE_RBS.

Assessment Questions

1. **A,B,C.** The three functions of rollback segments are read consistency, transaction recovery, and transaction rollback. For information see the “Purpose of Rollback Segments” section.
2. **C.** The minimum number of extents that a rollback segment must have is 2. Refer to the “Creating Rollback Segments” section for more information.
3. **E.** PCTINCREASE cannot be specified for rollback segments. All extents of a rollback segment must be the same size. For more information, refer to the section on “Creating Rollback Segments” section.
4. **D.** By setting OPTIMAL, Oracle automatically deallocates space that is no longer needed from the rollback segment. This prevents the rollback segments from growing and wasting space. If OPTIMAL is not set then the space must be deallocated manually. Refer to section “Shrinking Rollback Segments” for more information.
5. **D.** Rollback segments will shrink automatically when the size of the rollback segment exceeds the value of OPTIMAL. OPTIMAL must be set for this to occur. Refer to section “Shrinking Rollback Segments” for more information.
6. **E.** For more information, refer to section “Taking Rollback Segments Offline.”
7. **B.** If there are active transactions using the rollback segment when the OFFLINE command is issued the status goes from ONLINE to PENDING OFFLINE. When the active transactions complete the status changes to OFFLINE. For more information, refer to section “Taking Rollback Segments Offline.”
8. **B.** By placing the names of the rollback segments that you want brought online in the ROLLBACK_SEGMENTS parameter of the INIT.ORA PARAMETER file you guarantee that those rollback segments will be brought online whenever the instance is started. Just because a rollback segment is ONLINE now does not mean that it will still be ONLINE when the instance is restarted. For more information, refer to section “Bringing Rollback Segments Online.”
9. **A.** Refer to the section on “Getting Information About Rollback Segments” for more information.
10. **E.** The DBA_ROLLBACK_SEGS has a row for every rollback segment in the database. The V\$ROLLNAME and V\$ROLLSTAT views only have rows for ONLINE or PENDING OFFLINE rollback segments. For more information, refer to section “Getting Information about Rollback Segments.”

Scenarios

1. **A.** The fact that space is not being deallocated is most likely due to the fact that OPTIMAL has not been set for the rollback segments. If OPTIMAL is not set, DBAs must manually deallocate space from rollback segments. You should set the value of OPTIMAL for all the rollback segments.

1. **B.** The fact that all rollback segments have EXTENDED and none have SHRUNK is due to the fact that OPTIMAL is not set. However, setting OPTIMAL only causes space to be deallocated; it does not solve the problem of the EXTENDS. In fact, it causes more EXTENDS because, as transactions require more space, they are forced to extend. You should set the value of OPTIMAL equal to the amount of space required by each rollback segment. Or, consider adding more rollback segments. Both options should help reduce the number of times the rollback segment is forced to extend.
2. **A.** If there are transactions in the database that are failing due to lack of space in the rollback segment, the rollback segments need to be larger. This is only possible if there is enough space in the tablespace that is storing the rollback segments. First, ensure the space is available so that this does not cause the transactions to fail, and, second, increase the size of the rollback segments. This can be done one of two ways: Either increase the value of MAXEXTENTS or drop and re-create the rollback segments with larger extent sizes. Another option would be to create a large rollback segment specifically for the large transactions and ensure that the large transactions get assigned to that rollback segment by using the SET TRANSACTION USE ROLLBACK SEGMENT *rollback_segment_name* command.

Lab Solutions

Lab 10-1

3.

```
CREATE TABLESPACE RBS_TEST
  DATAFILE 'C:\CERTDB\DISK5\RBS_TEST.DBF' SIZE 2M
```

4.

```
CREATE ROLLBACK SEGMENT RBS7
  TABLESPACE RBS_TEST
  STORAGE (INITIAL 16k
           NEXT 16k
           MINEXTENTS 2
           MAXEXTENTS 10);
```

5.

```
SQL> SELECT SEGMENT_NAME
 2 ,        STATUS
 3 ,        INITIAL_EXTENT
 4 ,        NEXT_EXTENT
 5 ,        MIN_EXTENTS
 6 ,        MAX_EXTENTS
 7 FROM    DBA_ROLLBACK_SEGS
 8 WHERE   SEGMENT_NAME = 'RBS7';
```

SEGMENT_NAME	STATUS	INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENTS	MAX_EXTENTS
RBS7	OFFLINE	16384	16384	2	10

6.

```
SQL> SELECT COUNT(*)
2 FROM DBA_EXTENTS
3 WHERE SEGMENT_NAME = 'RBS7';
```

```

COUNT(*)
-----
                2
```

7.

```
ALTER ROLLBACK SEGMENT RBS7 ONLINE;
```

8.

```
SQL> SELECT RN.NAME
2 , RS.STATUS
3 FROM V$ROLLNAME RN
4 , V$ROLLSTAT RS
5 WHERE RN.USN = RS.USN
6 AND RN.NAME = 'RBS7'
7 /
```

NAME	STATUS
RBS7	ONLINE

Lab 10-2

3.

```
SQL> SHOW USER
USER is "STUDENT"
```

```
SQL> CREATE TABLE DB_TABLES
2 TABLESPACE USERS
3 AS
4 SELECT *
5 FROM DICT
6 /
```

Table created.

4.

```
SET TRANSACTION USE ROLLBACK SEGMENT RBS7
```

5.

```
DELETE FROM DB_TABLES WHERE ROWNUM < 201;
```

6.

```
SQL> show user
USER is "SYS"
```

```
SQL> SELECT SUBSTR(S.USERNAME,1,10) WHO
2 ,      SUBSTR(S.OSUSER,1,20) OS_WHO
3 ,      SUBSTR(R.NAME,1,8) "ROLLNAME"
4 ,      S.COMMAND
5 ,      S.SID
6 ,      S.SERIAL#
7 ,      T.USED_UBLK "ROLLBACK"
8 FROM    V$SESSION S, V$TRANSACTION T, V$ROLLNAME R
9 WHERE   T.SES_ADDR = S.SADDR
10 AND    R.USN = T.XIDUSN
11 /
```

WHO	OS_WHO	ROLLNAME	COMMAND	SID	SERIAL#	ROLLBACK
STUDENT	TODD\Administrator	RBS7		0	7	33

7.

```
SQL> ALTER ROLLBACK SEGMENT RBS7 OFFLINE;
```

Rollback segment altered.

8. The command did complete successfully.

9.

```
SQL> SELECT RN.NAME "Name"
2 ,      STATUS
3 FROM    V$ROLLNAME RN,
4         V$ROLLSTAT RS
5 WHERE   RS.USN = RN.USN
6 AND    RN.NAME = 'RBS7'
7 /
```

Name	STATUS
RBS7	PENDING OFFLINE

10. The rollback segment is PENDING OFFLINE because there is an active transaction on rollback segment RBS7. Taking it offline prevents other users from using the rollback segment, but it will not come offline until the active transaction completes.

11.

```
SQL> COMMIT;
```

Commit complete.

12.

```
SQL> SELECT RN.NAME "Name"
       2 ,
       3 STATUS
       4 FROM V$ROLLNAME RN,
       5 V$ROLLSTAT RS
       6 WHERE RS.USN = RN.USN
       7 AND RN.NAME = 'RBS7'
       8 /
```

Name	STATUS
RBS7	PENDING OFFLINE

13. The status is still pending offline. There should be no rows in V\$ROLLSTAT for rollback segment RBS7 as it is now OFFLINE.

14.

```
SQL> SELECT SEGMENT_NAME
       2 ,
       3 STATUS
       4 FROM DBA_ROLLBACK_SEGS
       5 WHERE SEGMENT_NAME = 'RBS7';
```

SEGMENT_NAME	STATUS
RBS7	OFFLINE

Lab 10-3

3.

```
SQL> show user
USER is "SYS"
```

```
SQL> ALTER ROLLBACK SEGMENT RBS7 ONLINE;
```

Rollback segment altered.

4.

```
SQL> SHOW USER
USER is "STUDENT"
```

```
SQL> DROP TABLE DB_TABLES;
```

Table dropped.

5.

```
SQL> CREATE TABLE DB_TABLES
2     TABLESPACE USERS
3     AS
4     SELECT *
5     FROM     DICT
6     /
```

Table created.

6.

```
INSERT INTO DB_TABLES
SELECT * FROM DB_TABLES;
```

```
INSERT INTO DB_TABLES
SELECT * FROM DB_TABLES;
```

```
INSERT INTO DB_TABLES
SELECT * FROM DB_TABLES;
```

7.

```
SQL> COMMIT;
```

Commit complete.

8.

```
SET TRANSACTION USE ROLLBACK SEGMENT RBS7;
```

9.

```
SQL> DELETE FROM DB_TABLES;
DELETE FROM DB_TABLES
*
ERROR at line 1:
ORA-01562: failed to extend rollback segment number 8
ORA-01628: max # extents (10) reached for rollback segment
RBS7
```

10. The transaction failed because it ran out of room in rollback segment RBS7. It reached the maximum number of allowable extents of 10.

11.

```
SQL> ALTER ROLLBACK SEGMENT RBS7 STORAGE (MAXEXTENTS 100);
```

Rollback segment altered.

12.

```
DELETE FROM DB_TABLES;
DELETE FROM DB_TABLES
*
ERROR at line 1:
ORA-01562: failed to extend rollback segment number 8
ORA-01650: unable to extend rollback segment RBS7 by 4 in
tablespace RBS_TEST
```

13. The transaction failed again, but this time it was because it ran out of room in tablespace RBS_TEST. The rollback segment could not create another extent.

14.

```
SQL> ALTER DATABASE DATAFILE 'C:\CERTDB\DISK5\RBS_TEST.DBF'
RESIZE 5M;
```

Database altered.

15.

```
SQL> ALTER ROLLBACK SEGMENT RBS7 STORAGE (MAXEXTENTS 256);
```

Rollback segment altered.

16.

```
SQL> DELETE FROM DB_TABLES
```

2424 rows deleted.

17. The command completes successfully this time.

18.

```
SQL> commit;
```

Commit complete.

19.

```
SQL> SELECT RN.NAME "Name"
2      ,      RS.EXTENTS
3      ,      RS.RSSIZE "SIZE"
4      ,      RS.OPTSIZE "OPTIMAL"
5      ,      RS.XACTS "TRANS"
6      ,      RS.EXTENDS
7      ,      RS.SHRINKS
8 FROM      V$ROLLNAME RN,
9           V$ROLLSTAT RS
10 WHERE    RS.USN = RN.USN
11 AND      RN.NAME = 'RBS7'
```

```
12 /
```

Name	EXTENTS	SIZE	OPTIMAL	TRANS	EXTENDS	SHRINKS
RBS7	77	2539520		0	74	0

20.

```
SQL> ALTER ROLLBACK SEGMENT RBS7 SHRINK TO 256K;
```

```
Rollback segment altered.
```

21.

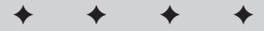
```
SQL> SELECT RN.NAME "Name"
2 , RS.EXTENTS
3 , RS.RSSIZE "SIZE"
4 , RS.OPTSIZE "OPTIMAL"
5 , RS.XACTS "TRANS"
6 , RS.EXTENDS
7 , RS.SHRINKS
8 FROM V$ROLLNAME RN,
9 V$ROLLSTAT RS
10 WHERE RS.USN = RN.USN
11 AND RN.NAME = 'RBS7'
12 /
```

Name	EXTENTS	SIZE	OPTIMAL	TRANS	EXTENDS	SHRINKS
RBS7	8	253952		0	74	7

Managing Tables

EXAM OBJECTIVES

- ◆ Managing Tables
 - Create tables using appropriate storage settings
 - Control the space used by tables
 - Analyze tables to check integrity and migration
 - Retrieve information about the tables from the data dictionary
 - Convert between different formats of ROWID



CHAPTER PRE-TEST

1. Who can create tables?
2. What is an Oracle Cluster?
3. What is the difference between the char and varchar2 datatypes?
4. How big can LOBs be?
5. What is a nested table?
6. What does PCTFREE parameter mean?
7. What is a free list?
8. How can the NOLOGGING parameter improve table-creation performance?
9. Where is the data stored for temporary tables?
10. Who can see the data inside a temporary table?
11. What does the MOVE option of the ALTER TABLE command do?
12. What is the high-water mark?
13. What data dictionary view reports the number of chained rows per table, and what must be done to determine this value?

The main focus of this chapter is on creating tables. When tables are created, DBAs must be careful to choose the correct storage parameters; otherwise, performance can suffer. The first part of the chapter covers the basics of data storage in an Oracle database. Part of this section discusses the different datatypes for columns in tables and compares the datatypes of LOB and LONG.

The object relational databases model is something that Oracle has pioneered. In Oracle, you can define both tables and columns as objects. An object can be made up of scalar datatypes or collections. Oracle has two types of collections: nested tables and VARRAYs. The last part of this section covers collections.

This chapter starts by covering the various aspects of creating tables. When tables are created, the DBA can either specify storage options for extent sizes and block storage or rely on defaults. First you will find out about storage parameters and the other options available for creating tables. Next, you will learn about something new in Oracle8.1, temporary tables. Temporary tables offer a fantastic way to store data temporarily that is used in a report or when performing maintenance operations. Data is stored in temporary segments; no rollback information is required and no redo information is generated, the absence of which can result in faster processing. Clusters offer DBAs added flexibility in terms of storage options, which can potentially lead to better query performance. Clusters do, however, have some limitations, which this section covers. Finally, some guidelines for reducing chaining and migration by appropriately setting the values for PCTFREE and PCTUSED, which are both block storage parameters will be outlined.

Because most of the DBA's work consists of administering existing databases, with tables that were already created, the next section, "Modifying Tables," is important. The DBA can change most of the original storage parameters that were specified when the tables were created. When changing these parameters, it is important to understand the steps involved in fixing it. A big part of this section interprets data dictionary views that contain information on tables. This is generally where a DBA determines that a problem may exist.

Last, this chapter covers the basics of ROWIDs. A ROWID is Oracle's unique identifier for every row in a table. A ROWID, which is not the same as a PRIMARY KEY, consists of four components that you can use to determine the location of a row. These components are the following:

- ♦ The file that a row belongs to
- ♦ The table that the row belongs to
- ♦ The block that the row belongs to
- ♦ The row number on the block

The ROWID changed in Oracle8, and Oracle contains a package called the DBMS_ROWID that you can use to extract the ROWID information and to convert between Oracle7 and Oracle8 ROWID formats.

Creating the structure of a table is generally the job of a designer. After that, all the responsibility for a table usually falls on the shoulders of a DBA. Tables are the main element of every database. Virtually every aspect of database activity centers on tables. DBAs have several options for creating tables as well as several different types of tables to use. Choosing among the options is what DBAs must be able to do and do well. The Oracle community has perpetuated so many different thoughts and myths concerning the relationship between table storage and performance that the variety can become overwhelming at times. Keeping things consistent in a database with regard to storage is always a good rule to follow. But the main requirement for effective management of tables in an Oracle database is sticking to the KYD principle. KYD stands for Know Your Data. DBAs must have a good understanding of not only the physical makeup of a table but also the types of activities performed against tables. A table that has data only inserted into (but never deleted from or updated) has much different storage requirements than a volatile one (a volatile one has many updates and deletes). For this reason, DBAs must always abide by the KYD principle.

Data Storage Basics

This section provides an overview of data storage in Oracle. Most people understand what a table is — maybe not in database terms, but the concept of rows and columns is not limited to databases. Tables are a very common way to organize data. Rows are composed of columns. The columns can consist of elements such as dates, names, numbers, comments, quantities, and even pictures or movies. The problem that DBAs face is that these different column types have different impacts on the distribution of data in both the table and row. So, let's look first at the basic storage areas for tables in Oracle.

Basic data storage structures

Oracle has several different options for storing data, some more complex than others. Outlined below are the basic data storage structures available in Oracle.

Regular table

People commonly refer to database tables as spreadsheets made up of a bunch of columns and rows. This basic type of table in Oracle is referred to as a “regular table.” A regular table consists of a collection of rows and columns that store information that you can retrieve, modify, and delete. No guidelines exist for the distribution of data. When an insert is processed, the SERVER process making the request looks for an available block on the free list for the table and inserts the row on that block. You have no guarantee that the next insert into the same table is going to go onto the same block. The data has no order to it and the DBA has little control over this situation.



You'll find information about free lists later in this chapter in the sections "The CREATE TABLE Syntax" and "Guidelines for creating tables."

Partitioned tables

Partitioned tables are tables that are split into many smaller tables based upon column values. These smaller tables, which all contain the same columns, store specific rows that share the same partition key value. For example, assume that the CLASSENROLLMENT table is partitioned by the ENROLLMENTDATE column and four partitions are created, one for each quarter. All rows in which the ENROLLMENTDATE column value is less than April 1 go on partition 1, less than July 1 but greater than April 1, on partition 2, less than October 1 but greater than July 1, on partition 3, and all values greater than October 1 on partition 4. When a row is being inserted, the SERVER process performing the insert determines which partition, 1 through 4, this row belongs to and then goes to the free list for that partition and gets a block to store the row. If the ENROLLMENTDATE value were October 11, then the row would be stored on partition 4. Figure 11-1 illustrates the difference between a regular table and a partitioned table.

Employee (Regular Table)				Employee (Partitioned Table by Hiredate)			
Id	Name	Hiredate	Salary	Id	Name	Hiredate	Salary
1	Bob	11-Jan-00	100	1	Bob	11-Jan-00	100
2	Bill	15-Oct-00	500	66	Lisa	20-Jan-00	440
75	Todd	02-Jul-00	300	...			
56	John	22-May-00	200	56	John	22-May-00	200
57	Mary	13-Oct-00	1000	200	Marc	22-Apr-00	400
66	Lisa	20-Jan-00	440	...			
21	Ian	17-Sep-00	200	75	Todd	02-Jul-00	300
133	Tim	12-Aug-00	500	133	Tim	12-Aug-00	500
122	Hees	21-Dec-00	800	21	Ian	17-Sep-00	200
...				...			
200	Marc	22-Apr-00	400	2	Bill	15-Oct-00	500
				57	Mary	13-Oct-00	1000
				122	Hees	21-Dec-00	800
				...			

Figure 11-1: Partitioned tables versus regular tables

You can partition tables by Range or Hash, or you can make them a composite of Range and Hash. Each partition is effectively a separate table, which means that partitions can belong to separate tablespaces and have their own storage settings.

Partitions are most useful for large tables. Large tables benefit from being distributed over multiple disks; this distribution improves IO and read performance for full table scans. Also, Parallel DML, which is multiple processes processing a DML statement is supported only on partitioned tables. This can dramatically improve the performance of DML statements on large tables. Oracle has also added features to the optimizer to limit queries using the partition key to individual partitions. For example, in Figure 11-1, if you ran a query looking for all employees hired in the month of May, the Oracle server would know to query only Partition 2 and ignore Partitions 1, 3, and 4. It does this even without the benefit of an index on the HIREDATE column.

Index-organized tables

Index-organized tables (IOT) store data in the index as opposed to the table. The data is stored in PRIMARY KEY order. Oracle does not maintain two separate segments, one for the table and another for the index. Because data in a Regular Table is not stored in any particular order, queries must scan every row to satisfy the WHERE clause unless an index is on the column(s) in the WHERE clause. For example, without any indexes on the CLASSENROLLMENT table, if a user issued the following query:

```
SELECT * FROM CLASSENROLLMENT
WHERE STUDENTNUMBER = 1001
```

Oracle would need to scan every row in the table looking for a STUDENTNUMBER of 1001. If an index existed on the STUDENTNUMBER column, Oracle could simply scan the index. Indexes are stored in a sorted order, which means that not every row would need to be scanned. Each index value, in this case STUDENTNUMBER 1001, has the STUDENTNUMBER and a pointer, called a ROWID, to the exact position in the table where the row for STUDENTNUMBER 1001 is located. By using the index, Oracle scans the index to retrieve the matching values and then goes to the table to retrieve the row using the ROWID from the index. Doing so eliminates the need to perform full table scans, which are very expensive when the table contains many rows. So, when you perform a query using an index, Oracle scans the index first and then extracts the data from the table using the ROWID.

This type of table is very efficient for what is called key-based access. Because IOTs store the table data in the order of the PRIMARY KEY, queries using the PRIMARY KEY return results quicker than from a regular table with a PRIMARY KEY because less IO is required. A single index lookup in this type of table requires a minimum of two units of IO, one for the index and another for table. IOTs can perform the same query in one unit of IO.

Structure of a row

Rows are always stored in Oracle blocks. The size of the blocks is determined by the database initialization parameter `DB_BLOCK_SIZE`. The size of the rows is a variable length, determined by the columns used to define the rows and the actual data being stored. Columns are stored in the order by which they are defined in the table, and any trailing NULL columns are not stored until they are assigned values. Every row has a Row Header, which is used to store the number of columns in the row, chaining information, and the lock status. The Row Data portion has two components for each column; the first is the column length and the second is the column value. One byte is stored if the length cannot exceed 250 bytes for the column values, and anything longer than 250 bytes requires three bytes to store the length. The column data or value is stored adjacent to the length. Because the table can have variable-length columns, Oracle needs to know how many bytes of data belong to each column. No space is required between rows. The starting position for each row is stored in the Row Directory. The Row Directory is part of the block header. Figure 11-3 illustrates the structure of a block as well as the structure of a row.

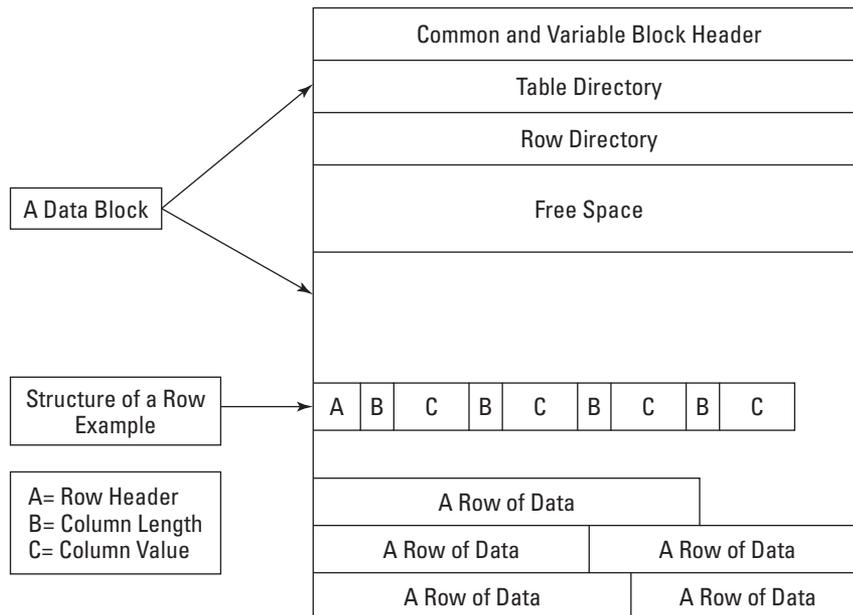


Figure 11-3: Block and row structure

Each block and row in Oracle has a defined structure shown in Figure 11-3, with the following elements making up these structures:

- ◆ The Common and Variable Block Header contains general information about the block, such as the block address, transaction information, and the type of segment this block belongs to (for example, table, index, rollback).
- ◆ Table Directory stores information about the table(s) that has rows on this block and is used mainly for clustered tables in which multiple tables can share the same block.
- ◆ Row Directory contains information about the actual rows, including the row addresses for each piece of a row in the row data area.
- ◆ Free space is space reserved in the block for updates of the rows on the block. Because Oracle has variable-length columns, room needs to be reserved on the block for future updates. Updates could change the amount of storage required by the row. For example, if an employee's job title changes from CLERK to SALES MANAGER, the amount of space required to store this information has changed from 5 bytes to 13 bytes. If space is not reserved for updates, changes can lead to row chaining and migration.



See the "Setting PCTFREE and PCTUSED" section, later in this chapter, for more details about free space and row migration and chaining.

- ◆ Row Data stores the actual row values for the table.

Built-in Oracle datatypes

Oracle offers several different built-in datatypes for storing scalar data, collections, and relationships. Table 11-1 lists and describes the scalar datatypes.

Table 11-1
Scalar Datatypes

<i>Built-In Datatypes</i>	<i>Description</i>
VARCHAR2(size)	Variable-length character string having maximum-length-size bytes. Maximum size is 4000 and minimum is 1. You must specify size for VARCHAR2.
NVARCHAR2(size)	Variable-length character string having maximum length size characters or bytes, depending on the choice of national character set. Maximum size is determined by the number of bytes required to store each character, with an upper limit of 4000 bytes. You must specify size for NVARCHAR2.
NUMBER(p,s)	Number having precision p and scale s. The precision p can range from 1 to 38. The scale s can range from -84 to 127.
LONG	Character data of variable length up to 2GB, or 2 ³¹ -1 bytes.
DATE	Valid date range from January 1, 4712 BC to December 31, 9999 AD.

Continued

Table 11-1 (continued)

Built-In Datatypes	Description
RAW(size)	Raw binary data of length-size bytes. Maximum size is 2000 bytes. You must specify size for a RAW value.
LONG RAW	Raw binary data of variable length up to 2GB.
ROWID	Hexadecimal string representing the unique address of a row in its table. This datatype is primarily for values returned by the ROWID pseudocolumn.
UROWID [(size)]	Hexadecimal string representing the logical address of a row of an index-organized table. The optional size is the size of a column of type UROWID. The maximum size and default is 4000 bytes.
CHAR(size)	Fixed-length character data of length-size bytes. Maximum size is 2000 bytes. Default and minimum size is 1 byte.
NCHAR(size)	Fixed-length character data of length size characters or bytes, depending on the choice of national character set. Maximum size is determined by the number of bytes required to store each character, with an upper limit of 2000 bytes. Default and minimum size is 1 character or 1 byte, depending on the character set.
CLOB	A character large object containing single-byte characters. Both fixed-width and variable-width character sets are supported, both using the CHAR database character set. Maximum size is 4GB.
NCLOB	A character large object containing multibyte characters. Both fixed-width and variable-width character sets are supported, both using the NCHAR database character set. Maximum size is 4GB. Stores national character set data.
BLOB	A binary large object. Maximum size is 4GB.
BFILE	Contains a locator to a large binary file stored outside the database. Enables byte stream IO access to external LOBs residing on the database server. Maximum size is 4GB.

Character data

Character data is stored in either fixed- or variable-length strings in the block. Character data is always stored in the character set of the database and will be converted if necessary. Fixed-length datatypes are CHAR and NCHAR; variable-length are VARCHAR2 and NVARCHAR2. Fixed-length character data-types store column values with padded blanks. The maximum number of bytes for these columns is 2000. Variable-character-length columns store only the column data without the padded blanks. The maximum number of bytes for these columns is 4000. Variable-length character columns generally require less storage space than the same data stored in fixed-length character columns. Following are examples of two tables,

each with one column. TAB1 has one column, COL1, that is a char(30). TAB2 also has one column called COL1 but it is a varchar2(30). The following string 'ABCDEFGHIJ', 10 characters, will be stored in the column COL1 of both tables, and the length of the string stored in the database will be retrieved from the DBA_TABLES data dictionary view.

```
SQL> DESC TAB1
Name                                     Null?    Type
-----
COL1                                              CHAR(30)

SQL> DESC TAB2
Name                                     Null?    Type
-----
COL1                                              VARCHAR2(30)
```

Just to prove that the same data exists in the two tables:

```
SQL> SELECT * FROM TAB1;

COL1
-----
ABCDEFGHIJ

SQL> SELECT * FROM TAB2;

COL1
-----
ABCDEFGHIJ
```

The average row length is taken from DBA_TABLES:

```
SQL> SELECT TABLE_NAME,AVG_ROW_LEN "AVG ROW LENGTH IN BYTES"
2 FROM DBA_TABLES
3* WHERE TABLE_NAME IN ('TAB1','TAB2')
```

TABLE_NAME	AVG ROW LENGTH IN BYTES
TAB1	34
TAB2	14

From this output, you can see that TAB1 with the CHAR(30) column is using 34 bytes. Three bytes are for the Row Header, 1 byte is for the Column Length, and 30 bytes are for the Column Value. You can see that the data is padded to the full 30 characters. TAB2 with the VARCHAR2(30) column stores only 14 bytes: again, 3 for the Row Header, 1 for the Column Length and only 10 bytes for the Column Value. The data is not padded.



The AVG_ROW_LEN column is populated only after the table is ANALYZED.

Numeric data

Numbers in Oracle are stored as variable-length data up to a maximum of 38 significant digits. Numeric datatypes have the following storage requirements:

- ♦ One byte for the exponent
- ♦ One byte for every two significant digits of the mantissa or decimal component of a number
- ♦ One byte for negative numbers when the number of significant digits is less than the maximum of 38

You can specify the scale and precision of a fixed-point number column for extra integrity checking on input. An example is something like

```
COL2 NUMBER(8,4)
```

where 8 is the precision, meaning that the numeric has eight significant digits, and 4 is the scale, meaning that four digits will be to the right of the decimal. The decimal does not count as a digit in the precision. Specifying scale and precision does not force all values to a fixed length. If a value exceeds the precision, Oracle returns an error. In the preceding example for COL2, if the following value were inserted (12345.1), Oracle would return an error because the precision is exceeded. Only four digits can go to the left of the decimal point. However, the following would not return an error: (1234.1234567890). Rather, the scale would be rounded, and 1234.1234567890 would become 1234.1235, with the 4 being rounded up to a 5. Oracle will round values that exceed the scale up.

Table 11-2 lists some examples of numbers with and without precision and scale.

Table 11-2 Numeric Data		
<i>Datatype</i>	<i>Insert Value</i>	<i>Value Stored in Table</i>
NUMBER	123456.987	123456.987
NUMBER	1234567890	1234567890
NUMBER	1234567890.987654321	1234567890.987654321
NUMBER(6,2)	1234.99	1234.99
NUMBER(6,2)	12345.99	Error, exceed Precision
NUMBER(6,2)	1234.4567890	1234.46
NUMBER(5)	123	123
NUMBER(5)	12345.678	12346

The DATE datatype

Dates are stored as fixed-length character strings of seven bytes. The dates are stored as an internal numeric value regardless of the format used to insert the date value into the column. If a date is inserted into a column using the format January twenty-second, 2001, it will still occupy only seven bytes. Oracle converts this format to the internal numeric format and then reconverts it, when retrieved, to the format specified by the query. Oracle will always store the time with dates. There is no separate date/time datatype. You use the TO_CHAR function to extract the time from a date column.

```
SQL> DESC TAB1
Name                                     Null?      Type
-----
COL1                                     NUMBER
COL2                                     DATE

SQL> INSERT INTO TAB1
2  VALUES (1,SYSDATE);

1 row created.
```

Note that COL2 is a DATE datatype and SYSDATE was inserted. Oracle always inserts the current time as well as the current date when SYSDATE is specified.

```
SQL> SELECT      COL1
2  , TO_CHAR(COL2,'DD-MM-YY:HH:MI:SS') "Date and Time"
3* FROM TAB1

          COL1 Date and Time
-----
1  21-03-01:03:01:43
```

If a date is inserted without a time, Oracle defaults the time to 12:00:00 AM.

```
SQL> INSERT INTO TAB1 VALUES (2,'01-JAN-01');

SQL> SELECT      COL1
2  , TO_CHAR(COL2,'DD-MM-YY:HH:MI:SS PM') "Date and Time"
3* FROM TAB1

          COL1 Date and Time
-----
2  01-01-01:12:00:00 AM
1  21-03-01:03:01:43 PM
```

The RAW datatype

The RAW datatype is used to store binary data less than 2000 bytes in length. Oracle does not convert the RAW data the same way that character data is converted into the database character set.

Large object datatypes

Prior to Oracle8, two datatypes existed for storing large objects: LONG and LONG RAW. These datatypes typically stored unstructured data such as chapters of books, recipes, comments, documents, and geographical information. LONG datatypes are variable-length character strings that can store up to two gigabytes of data. LONG RAW permits the storage of up to two gigabytes of binary data. Oracle8 introduced four new datatypes for storing large objects that are replacing LONG and LONG RAW: CLOB (character large object), BLOB (binary large object), BFILE, and NCLOB. These will be referred to as LOB datatypes for this book. LOBs can be character or binary and can store up to 4GB of data. Oracle recommends that LONG datatypes be migrated or converted to LOB because LONGs are being de-supported in a future release of Oracle.

When deciding to make use of the LONG datatype, be aware that it is only provided for historical reasons. The LONG and LONG RAW datatypes have a number of inherent restrictions including:

- ♦ A table cannot contain more than one LONG column.
- ♦ You cannot create an object type with a LONG attribute.
- ♦ LONG columns cannot appear in integrity constraints (except for NULL and NOT NULL constraints).
- ♦ LONG columns cannot be indexed.
- ♦ A stored function cannot return a LONG value.
- ♦ Within a single SQL statement, all LONG columns, updated tables, and locked tables must be located on the same database.

LONG columns cannot appear in certain parts of SQL statements:

- ♦ WHERE clauses, GROUP BY clauses, ORDER BY clauses, or CONNECT BY clauses or with the DISTINCT operator in SELECT statements
- ♦ The UNIQUE operator of a SELECT statement
- ♦ The column list of a CREATE CLUSTER statement
- ♦ The CLUSTER clause of a CREATE MATERIALIZED VIEW statement
- ♦ SQL functions (such as SUBSTR or INSTR)
- ♦ Expressions or conditions
- ♦ SELECT lists of queries containing GROUP BY clauses
- ♦ SELECT lists of subqueries or queries combined by set operators
- ♦ SELECT lists of CREATE TABLE ... AS SELECT statements
- ♦ SELECT lists in subqueries in INSERT statements

The best way to appreciate LOBs is to do a quick comparison of the differences between the LOBs and LONGs. Table 11-3 outlines the key differences between the two.

Table 11-3
Comparing LOBs to LONGs

<i>LONG, LONG RAW</i>	<i>LOB</i>
Single column per table and should always be the last column in a table.	No restrictions on the number per table.
Maximum size 2GB.	Maximum size 4GB.
Select returns the data.	Select returns the LOB locator.
Data is stored with the table. This means that every time something is selected from the table, the LONG column is processed as well. If the column has 50 megabytes, then 50 megabytes are processed.	Data is stored out-of-line in a LOB segment, which is like another table. The table stores only a LOB locator, which is a pointer, similar to a ROWID that is used to retrieve the data. The LOB information is processed only if the LOB is requested. LOBs are stored inline with the table if they are less than 4KB in size and only if the LOB "enable storage in row" was specified at creation.
No object type support.	Supports object types such as user-defined objects.
Sequential access to chunks	Random access to chunks, which means that parts of the LOB can be updated as opposed to the entire LOB as it is with LONGs.

The ROWID and UROWID datatypes

A ROWID is how Oracle uniquely identifies every row in the database. The ROWID is not stored as a column of a table; it can, however, be queried but not modified. The ROWID is the absolute fastest way—even faster than an index—to retrieve a row in a query or an update or delete statement. Indexes store the value being indexed and the ROWID of the associated row. Take, for example, an index on the LOCATIONNAME column of the LOCATIONS table. It stores the LOCATIONNAMES (Downtown Toronto, New York Park Avenue, San Francisco Downtown) and the corresponding ROWID to the LOCATIONS table for each row. When retrieving a row from the LOCATIONS table using the index on LOCATIONNAME, you first need to locate the correct LOCATIONNAME and get the associated ROWID. The ROWID is then used to get the row from the LOCATIONS table. This process would be quicker if the ROWID could be used directly; however, this is not usually possible. Oracle maintains the ROWIDs, and a user would have no way of deciphering their values without some help. If you use a ROWID datatype, you generally do so to point to

rows in other tables. You may use this datatype to speed up join performance of some queries, instead of using indexes, or you may use it to keep track of change information. The downside to using the ROWID datatype for maintaining relationships between tables is that Oracle does not update the ROWID column when the ROWID it is pointing to changes. ROWIDs change when rows are deleted. If the DBA rebuilds a table, this involves deleting rows and then reinserting them. When this happens, new ROWIDs are assigned even though exactly the same rows are being deleted and then reinserted. You should not use ROWIDs to maintain a relationship between two tables if the ROWIDs are changing. If the relationship is maintained through constraints, Oracle will prevent relationships from becoming invalid.



Constraints are covered in Chapter 13.

The UROWID datatype was introduced in Oracle8.1. It stands for Universal ROWID and supports ROWIDs or pointers to tables other than ORACLE tables. Oracle uses the UROWID for Index-Organized tables that require secondary indexes. Index-Organized tables do not have ROWIDs; therefore, you must use the UROWID. The data is stored as a hexadecimal string.

Structure of the ROWID

The ROWID in Oracle8 and above requires 10 bytes of storage per row and is displayed using 18 characters. The new ROWID is broken into four parts: Data Object Number, Relative File Number, Block Number, and Row Number. The following is an example of a query extracting the ROWID from the LOCATIONS table:

```
SQL> col "Data Object Number" head "Data|Object|Number"
      2 col "Relative File Number" head "Relative|File|Number" for A10
      3 col "Block Number" head "Block|Number"
      4 col "Row Number" head "Row|Number" for A10
      5 SELECT ROWID, SUBSTR(ROWID,1,6) "Data Object Number",
      6             SUBSTR(ROWID,7,3) "Relative File Number",
      7             SUBSTR(ROWID,10,6) "Block Number",
      8             SUBSTR(ROWID,16,3) "Row Number"
      9 FROM   LOCATIONS
     10 /
     11 col "Data Object Number" clear
     12 col "Relative File Number" clear
     13 col "Block Number" clear
     14* col "Row Number" clear
```

ROWID	Data Object Number	Relative File Number	Block Number	Row Number
AAAFB2AAHAAAAAXAAA	AAAFB2	AAH	AAAAAX	AAA
AAAFB2AAHAAAAAXAAB	AAAFB2	AAH	AAAAAX	AAB
AAAFB2AAHAAAAAXAAC	AAAFB2	AAH	AAAAAX	AAC

Written this way, it is easy to see the four distinct elements of the ROWID, which are as follows:

- ♦ The Data Object Number identifies the object or segment. In this case, the object is the LOCATIONS table that this ROWID belongs to. It is the first six characters of the ROWID. The Data Object Number is AAafb2. Internally on the block, the number needs 32 bits. The Data Object Number will be the same for all ROWIDs in the same table.
- ♦ The Relative File Number is the file number that this ROWID belongs to. It is three characters long and occupies position 7–9 of the ROWID. Internally, the number requires 10 bits for storage. The file number is relative to the tablespace. Each Relative File Number is unique within a tablespace, and because objects can belong to only one tablespace, this feature guarantees that the number will be unique. The Relative File Number can be different for ROWIDs in the same table only if the tablespace the object was created on is composed of multiple datafiles. In the preceding example, all the rows are on Relative File AAH.
- ♦ The Block Number is the block number of the block in the datafile. The number is located in positions 10–15 of the ROWID. In this case, all rows are located on BLOCK AAAAAX. The Block Number is unique to the combination of Relative File Number and Data Object Number. Internally, it requires 22 bits for storage.
- ♦ The Row Number identifies the position of the row in the directory slot of the block header. Each Row Number will be unique within the Block Number, Relative File Number, and Data Object Number combination. Internally, it requires 16 bits for storage on the block.

The storage requirements for a ROWID is 32 bits (Data Object Number) plus 10 bits (Relative File Number) plus 22 bits (Block Number) plus 16 bits (Row Number) for a total of 80 bits or 10 bytes.

Oracle displays the ROWID using a base-64 encoding scheme that uses the characters “A-Z”(26) “a-z”(26) “0-9”(10) “+” (1) “/”(1) for a total of 64 characters. The values themselves do not tie directly to values in the data dictionary. The DBMS_ROWID package is required to convert the base-64 values to numbers that can be used to compare to the data dictionary.



The DBMS_ROWID Package is covered later in this chapter in the section “DBMS_ROWID Package.”

Restricted ROWID

The format of the ROWID changed starting in Oracle8. Prior to Oracle8, the restricted ROWID format was used. The restricted ROWID does not have the Object Number component of the new ROWID. The restricted ROWID is made up of the Block Number, Row Number, and File Number. The File Number was unique within

the database but was limited to 1022 numbers. This meant that the maximum number of datafiles that a database could have was 1022. The restricted ROWID required only 6 bytes of storage on the block. The new ROWID format contains a Relative File Number. It is relative to the tablespace. This has removed the 1022 datafile limit in the database, which was a major limitation for very large databases. The maximum number of datafiles is now 65533.

In Oracle8 and later, the restricted ROWID is still used by Oracle in nonpartitioned indexes in which all index entries refer to rows within the same segment. This is possible because when Oracle is using an index, it already knows the Data Object Number, so it does not need to be stored at the index level. This capability saves four bytes per index entry, which is a significant amount of space on large tables. By using the restricted ROWID in indexes for a table with one million rows, Oracle is saving four million bytes, or roughly 4MB.

Collection types

The introduction of objects to the Oracle database allowed developers to take full advantage of Object Oriented programming languages such as Java. Prior to Oracle8, when object-oriented programming languages were interacting with the database, objects had to be decomposed down to a relational level and then recomposed when information was passed back. This “impedance mismatch” caused unnecessary difficulty for developers and resulted in increased development times. Database objects eliminate the “impedance mismatch” between programming languages and databases. Oracle calls this capability the “Object Relational Model.”

In Oracle8 and later, you can define tables as objects or define objects as elements of tables with other scalar datatypes. Collections are a group of elements of the same type. A VARRAY, short for variable array, is similar to arrays in third-generation languages. Nested tables let you define tables as columns of a table.

VARRAYs

A VARRAY is an ordered set of data elements that are all of the same datatype. They have a count for the number of elements in the array as well as an upper limit on the number of elements. As with other arrays, an index is used to identify each element. Following is an example of using a VARRAY to resolve a classic header/detail or parent/child relationship, the ORDERS and ITEMS relationship.

First, you create an OBJECT called `line_item_type`. You use this OBJECT rather than create a table called ITEMS.

```
SQL> CREATE OR REPLACE TYPE LINE_ITEM_TYPE AS OBJECT
2   ( PROD_ID          NUMBER(6),
3     PRICE            NUMBER(9,2),
4     QUANTITY         NUMBER(5) );
5 /
```

Type created.

Next, you create a TYPE as a VARRAY. It references the OBJECT created in the previous step. Therefore, instead of being an ITEMS table, it will be a VARRAY. This TYPE is reusable, as is the OBJECT.

```
SQL> CREATE OR REPLACE TYPE LINE_ITEM_ARRAY_TYPE
  2   AS VARRAY(100) OF LINE_ITEM_TYPE;
  3   /
```

Type created.

Now, create the ORDERS table and notice how the LINE_ITEMS column is defined. It is defined as LINE_ITEM_ARRAY_TYPE.

```
SQL> CREATE TABLE ORDERS
  2   ( ORD_ID NUMBER(9),
  3     CUST_ID NUMBER(6),
  4     LINE_ITEMS LINE_ITEM_ARRAY_TYPE )
VARRAY LINE_ITEMS STORE AS LOB ORD_LINE_ITEMS;
```

Table created.

```
SQL> desc ORDERS
Name                                                    Null?   Type
-----
ORD_ID                                                    NUMBER(9)
CUST_ID                                                    NUMBER(6)
LINE_ITEMS
LINE_ITEM_ARRAY_TYPE
```

Now you insert data into the ORDERS table. Notice how the ITEMS for the ORDER are inserted. This is very different from conventional header/detail entries in which the data would be inserted into two separate tables. You have no FOREIGN KEY constraints!

```
SQL> INSERT INTO ORDERS
  2   VALUES (100, 200,
  3     LINE_ITEM_ARRAY_TYPE (
  4     LINE_ITEM_TYPE (300, 4.5, 600),
  5     LINE_ITEM_TYPE (400, 5.75, 200) ));
```

1 row created.

```
SQL>
SQL>
SQL> INSERT INTO ORDERS
  2   VALUES (500, 600,
  3     LINE_ITEM_ARRAY_TYPE (
  4     LINE_ITEM_TYPE (700, 6.5, 60),
  5     LINE_ITEM_TYPE (800, 7.75, 100) ));
```

1 row created.

When you query the ORDERS table, the output will look very different than most people are used to seeing:

```
SQL> COL LINE_ITEMS      FORMAT A40
SQL>
SQL> SELECT      *
      2 FROM      ORDERS;

      ORD_ID      CUST_ID LINE_ITEMS(PROD_ID, PRICE, QUANTITY)
-----
      100         200 LINE_ITEM_ARRAY_TYPE(LINE_ITEM_TYPE(300,
                        4.5, 600), LINE_ITEM_TYPE(400, 5.75, 20
                        0))

      500         600 LINE_ITEM_ARRAY_TYPE(LINE_ITEM_TYPE(700,
                        6.5, 60), LINE_ITEM_TYPE(800, 7.75, 100
                        ))
```

You can use a special TABLE function to make the output look much neater:

```
SQL> COL ORD_ID FORMAT 99999
SQL> COL CUST_ID      FORMAT 999999
SQL> COL PROD_ID      FORMAT 999999
SQL> COL PRICE        FORMAT 99,999.99
SQL> COL QUANTITY     FORMAT 999,999
SQL>
SQL> SELECT      0.ORD_ID,
      2          0.CUST_ID,
      3          I.PROD_ID,
      4          I.PRICE,
      5          I.QUANTITY
      6 FROM      ORDERS                                0,
      7          TABLE(0.LINE_ITEMS)                  I
      8 WHERE 0.ORD_ID = 100;

ORD_ID CUST_ID PROD_ID      PRICE QUANTITY
-----
      100      200      300         4.50      600
      100      200      400         5.75      200
SQL>
SQL> COL LINE_ITEMS      CLEAR
SQL> COL ORD_ID CLEAR
SQL> COL CUST_ID      CLEAR
SQL> COL PROD_ID      CLEAR
SQL> COL PRICE        CLEAR
SQL> COL QUANTITY     CLEAR
SQL>
```

Nested tables

A nested table is just that: a table within a table. It provides a natural mapping for one-to-many or Header/Detail relationships such as the ORDERS and ITEMS example. The nested table is defined separately from the main table and can have its own storage parameters and even be stored in a separate tablespace. Unlike with a VARRAY, no limit exists for the number of elements and piece-wise updates of rows supported in the nested table. Individual rows in a VARRAY cannot be modified. Here is the same ORDERS and ITEMS example but this one uses a nested table rather than a VARRAY:

```
SQL> DROP TABLE ORDERS;
```

Create the `LINE_ITEM_TABLE_TYPE` TYPE. It is created as a TABLE instead of a VARRAY.

```
SQL> CREATE OR REPLACE TYPE LINE_ITEM_TABLE_TYPE
2   AS TABLE OF LINE_ITEM_TYPE;
3   /
```

Type created.

You create the table `ORDERS` specifying the column `LINE_ITEMS` as the `OBJECT TYPE` created in the previous step:

```
SQL>
SQL> CREATE TABLE ORDERS
2   ( ORD_ID NUMBER(9),
3     CUST_ID   NUMBER(6),
4     LINE_ITEMS LINE_ITEM_TABLE_TYPE )
5   NESTED TABLE LINE_ITEMS STORE AS LOB ORD_LINE_ITEMS;
```

Table created.

```
SQL>
SQL> INSERT INTO ORDERS
2   VALUES (100, 200,
3     LINE_ITEM_TABLE_TYPE (
4     LINE_ITEM_TYPE (300, 4.5, 600),
5     LINE_ITEM_TYPE (400, 5.75, 200) ));
```

1 row created

```
SQL>
SQL> INSERT INTO ORDERS
2   VALUES (500, 600,
3     LINE_ITEM_TABLE_TYPE (
4     LINE_ITEM_TYPE (700, 6.5, 60),
5     LINE_ITEM_TYPE (800, 7.75, 100) ));
```

1 row created.

Now select the rows just inserted. Like the VARRAY example, the output looks a little funny.

```
SQL>
SQL> COL LINE_ITEMS          FORMAT A40
SQL>
SQL> SELECT      *
      2 FROM      ORDERS;

ORD_ID  CUST_ID  LINE_ITEMS(PROD_ID, PRICE, QUANTITY)
-----
      100      200  LINE_ITEM_TABLE_TYPE(LINE_ITEM_TYPE(300,
                        4.5, 600), LINE_ITEM_TYPE(400, 5.75, 20
                        0))
      500      600  LINE_ITEM_TABLE_TYPE(LINE_ITEM_TYPE(700,
                        6.5, 60), LINE_ITEM_TYPE(800, 7.75, 100
                        ))
```

Rewrite the query using the TABLE function to reformat the output:

```
SQL> SELECT      O.ORD_ID,
      2          O.CUST_ID,
      3          I.PROD_ID,
      4          I.PRICE,
      5          I.QUANTITY
      6 FROM      ORDERS          O,
      7          TABLE(O.LINE_ITEMS) I
      8 WHERE     I.PROD_ID IN (300, 700);

ORD_ID  CUST_ID  PROD_ID  PRICE  QUANTITY
-----
      100      200      300      4.5      600
      500      600      700      6.5      60
```

Creating Tables

Objective

Create tables using appropriate storage settings

Creating tables is one of the most important jobs of a DBA. The name of the table, the column names and datatypes, and the constraint information are usually provided to the DBA by a developer or designer. Many times, the third-party application products have scripts that are used to create the database objects. DBAs must decide upon proper extent storage and block utilization parameters as well as the tablespace where the table will be created. The DBA should make these decisions even when provided with a script. Managing tables is the responsibility of the DBA.

To create a table, the user must have the CREATE TABLE or CREATE ANY TABLE (CREATE ANY TABLE can be used to create tables in other users' schemas) system privilege as well as a quota on the tablespace where the table is created, or have the UNLIMITED TABLESPACE system privilege.

The CREATE TABLE syntax

The following is the simplified syntax of the CREATE TABLE statement:

```
CREATE TABLE [schema.] table_name
(column_name data type [in-line constraint_clause]
 [ ,column_name data type] ... out_of_line constraint_clause)
[TABLESPACE tablespace_name ]
  [ PCTFREE integer ]
  [ PCTUSED integer ]
  [ INITRANS integer ]
  [ MAXTRANS integer ]
  [ LOGGING | NOLOGGING ]
  [ CACHE | NOCACHE ]
[STORAGE ( [ INITIAL integer [K|M] ]
  [ NEXT integer [K|M] ]
  [ MINEXTENTS integer ]
  [ MAXEXTENTS integer | UNLIMITED ]
  [ PCTINCREASE integer ]
  [ FREE LISTS integer ]
  [ BUFFER POOL KEEP | RECYLCE | DEFAULT ]
  )]
```



Chapter 13 covers the inline and out-of-line constraint clauses.

Table 11-4 defines the meaning of each of the key words.

Table 11-4 Options of the CREATE TABLE Command	
<i>Option</i>	<i>Description</i>
Schema	Owner of the table.
table_name	Name of the table.
column_name	Name of the column.
Datatype	Datatype of the column.
TABLESPACE	The name of the tablespace where this table is to be located. The tablespace cannot be a temporary tablespace.
	Default value: the user default tablespace

Continued

Table 11-4 (continued)

<i>Option</i>	<i>Description</i>
PCTFREE	<p>The amount of space, specified as a percentage, reserved on a block for rows to be updated. If PCTFREE is set to 20, then 20 percent of the block will be reserved for updates. The percentage does not include the block header. If the database block size is 2048 bytes, then a value of 20 percent for PCTFREE will reserve 20 percent of (db_block_size – block_header). In this example, it is roughly 370 bytes because the block header occupies roughly 200 bytes.</p> <p>Default value: 10 percent</p>
PCTUSED	<p>A lower limit for space used on a block after it has exceeded the value of PCTFREE. Updates and inserts use space on blocks; updates and deletes free space on blocks. When the amount of free space on a block goes below the value of PCTUSED, then the block is made available for inserts again.</p> <p>Default value: 40 percent</p>
INITRANS	<p>Specifies the number of permanent transaction slots configured in the header portion of the block. The transaction slots are used to hold the information about rows being modified on the block.</p> <p>Default value: 1</p>
MAXTRANS	<p>Specifies the maximum number of transactions permitted per block. If INITRANS is set to 1 and a user is currently updating a row on a block, then that user will get the permanent transaction slot. If another user comes along and starts a transaction on another row on the same block, a temporary transaction slot must be created because the 1 permanent one is being used. The MAXTRANS value sets the maximum number of temporary transaction slots that can be created on a block.</p> <p>Default value: 255</p>
LOGGING	<p>Indicates whether the table creation is to be logged in the redo log files. Turning off logging can improve the table creation performance for tables created using the CREATE TABLE AS SELECT (CTAS) statement. Only the table creation is not logged; future DML statements are logged in the redo log files regardless of the value of LOGGING on the table.</p> <p>Default value: LOGGING</p>
NOLOGGING	<p>Indicates that table creation is not to be logged in the redo log files. This can dramatically improve the performance of the CTAS command</p> <p>Default value: LOGGING</p>
CACHE	<p>Place blocks read from this table to the Most Recently Used end of the LRU list when the table is read during Full Table Scan.</p> <p>Default value: NOCACHE</p>
NOCACHE	<p>Place blocks read from this table to the Least Recently Used end of the LRU list when the table is read during a Full Table Scan.</p>

<i>Option</i>	<i>Description</i>
STORAGE CLAUSE	
INITIAL	Size of the first extent created. Value can be specified in bytes, kilobytes, megabytes, or gigabytes. Should be a multiple of DB_BLOCK_SIZE * DB_FILE_MULTIBLOCK_READ_COUNT
NEXT	The size of the second extent created. Value can be specified in bytes, kilobytes, megabytes, or gigabytes. Should be a multiple of DB_BLOCK_SIZE * DB_FILE_MULTIBLOCK_READ_COUNT
MINEXTENTS	Number of extents initially created.
MAXEXTENTS	Maximum number of extents allowed for this table.
PCTINCREASE	<p>You use this parameter to determine the size of every extent created after the second. The value is a percentage. When the third extent is created, the size of the extent is determined by multiplying the value of the previous extent, which in this case is the value of NEXT times PCTINCREASE plus the value of NEXT. (SIZE OF LAST EXTENT × PCTINCREASE + SIZE OF LAST EXTENT). The size of the fourth extent will be the size of the third extent times PCTINCREASE of the size of the third extent, and so on. The calculation will always round to the nearest multiple of DB_BLOCK_SIZE.</p> <p>For example, if the value of NEXT is 1MB and the value of PCTINCREASE is 50 percent, the third extent will be 1MB + (1MB × 50 percent). This equals 1.5MB. The fourth extent will be 1.5MB + (1.5MB * 50 percent) which equals 2.25MB, and so on. The extent size will be rounded to the nearest multiple of DB_BLOCK_SIZE.</p> <p>A value of zero for PCTINCREASE means that all extents from the third and on will be equal to the value of NEXT. This is the recommended value for most tables.</p> <p>Default value: 50 percent</p>
FREE LISTS	<p>This parameter specifies the number of free lists for the table. Free lists contain a list of blocks that are available for insert. When a block has less free space than that specified by PCTFREE, the block is removed from the free list. When rows are deleted from a block and the amount of used space on the block shrinks below the value of PCTUSED, the block is placed on the free list again, making it able to accept inserts again.</p> <p>Default Value : 1</p>
BUFFER POOL	<p>The BUFFER POOL parameter specifies the database buffer cache where blocks for this table will reside when read into memory. The possible values are KEEP, RECYLCE, and DEFAULT.</p> <p>Default Value: DEFAULT</p>

Storage parameter exceptions

When you create tables, you will always use the extent and block storage parameters specified, with the following two exceptions. If you set the `MINIMUM EXTENT` parameter in the tablespace where the table is being created, then all extents must be a multiple of this value, regardless of the values specified for `INITIAL` and `NEXT`. To demonstrate this requirement, Oracle creates a tablespace specifying a default storage parameter for `MINIMUM EXTENT` to 100KB. It then creates a table on this tablespace specifying a value for `INITIAL` of 10KB. When the data dictionary view `DBA_EXTENTS` is queried, you will see that the extent size for this table is 100KB.

```
CREATE TABLESPACE TEST_STORAGE
DATAFILE 'c:\certdb\disk6\test_storage.dbf' size 10M
MINIMUM EXTENT 100k

CREATE TABLE TEST_EXT (X NUMBER(1)) TABLESPACE TEST_STORAGE
STORAGE (INITIAL 10k);

SELECT substr(SEGMENT_NAME,1,15) TABLE_NAME,
       TABLESPACE_NAME, BYTES/1024 "EXT SIZE in K"
FROM   DBA_EXTENTS
WHERE  SEGMENT_NAME='TEST_EXT'
```

TABLE_NAME	TABLESPACE_NAME	EXT SIZE in K
TEST_EXT	TEST_STORAGE	104

You can see that even though the value of `INITIAL` was set to 10KB when the table was created, the extent size is actually 104KB. The value of `MINIMUM EXTENT` or a multiple of `MINIMUM EXTENT` will always be used if specified at the tablespace level. The extent size is 104KB because extents must always be a multiple of `DB_BLOCK_SIZE`. The block size is 8192 bytes or 8KB, therefore 104KB is the closest multiple to 8KB.

The other exception occurs when locally managed tablespaces are used. The size of extents, in locally managed tablespaces, is always taken from the tablespace regardless of storage settings in the `CREATE TABLE` statement. Following is the same example as the previous one, except that this time, instead of specifying `MINIMUM EXTENT`, the tablespace will be a locally managed tablespace.

```
DROP TABLESPACE TEST_STORAGE INCLUDING CONTENTS;

CREATE TABLESPACE TEST_STORAGE
DATAFILE 'C:\CERTDB\DISK6\TEST_STORAGE.DBF' REUSE
EXTENT MANAGEMENT LOCAL
UNIFORM SIZE 100k;

CREATE TABLE TEST_EXT (X NUMBER(1)) TABLESPACE TEST_STORAGE
STORAGE (INITIAL 10k);
```

```
SELECT substr(SEGMENT_NAME,1,15) TABLE_NAME,
       TABLESPACE_NAME, BYTES/1024 "EXT SIZE in K"
FROM   DBA_EXTENTS
WHERE  SEGMENT_NAME='TEST_EXT';
```

TABLE_NAME	TABLESPACE_NAME	EXT SIZE in K
TEST_EXT	TEST_STORAGE	104

Once again, you can see that the size of the extent created is 104KB even though 10KB was specified when the table was being created.

Caching Tables

The CACHE parameter of the CREATE TABLE statement specifies where blocks will be placed on the Least Recently Used (LRU) list of the database buffer cache when read in during a full table scan. The NOCACHE is the default and means that blocks are placed on the least recently used end of the LRU list. If CACHE is specified, blocks are placed at the most recently used end. Blocks at the least recently used end of the LRU list are the first ones to be aged out of the database buffer cache when blocks are needed. Blocks that are read in by the ROWID, as is the case with an index read, go to the most recently used end. The number of blocks read in during each IO pass of a full table scan is determined by the parameter DB_FILE_MULTIBLOCK_READ_COUNT.

Accessing blocks in memory is much faster than having to perform an IO operation to read them in, so tables that you query by a full table scan can benefit from the CACHE option. The likelihood of a memory read versus an IO read increases for cached tables. The only tables that you should cache are small lookup tables. Specifying the CACHE option does not guarantee that tables will always be accessed in memory. The database buffer cache is a finite amount of memory equal to DB_BLOCK_SIZE * DB_BLOCK_BUFFERS, and if memory is required, Oracle will age out blocks regardless of the CACHE parameter. By default, Oracle stores only five blocks in memory from a table that is cached. Using the CACHE option for large tables will have limited benefit for full table scans.

LOGGING versus NOLOGGING

The LOGGING parameter determines whether Oracle will record the table creation in the redo log files. The NOLOGGING option specifies that the initial creation and rows inserted during the creation and any future INSERT APPEND operations will not be logged. You use the NOLOGGING option most commonly when tables you're creating tables using the CREATE TABLE AS SELECT (CTAS) command. The NOLOGGING option speeds up the performance of the table creation and future insert append operations by avoiding the costs of recording change information in the redo log files. The performance can be five to six times faster. Using NOLOGGING has a downside: With initial rows loaded, any future rows inserted during INSERT APPEND operations are not recoverable and force DBAs to perform backups after these types of commands to ensure that no data loss occurred.

The NOLOGGING option applies only to the table creation and certain types of insert statements such as INSERT APPEND. Regular inserts, updates, and deletes are recorded in the redo log files regardless of the LOGGING status. If you do not set the option parameter, it obtains its value from the tablespace where the table is being created.

If tables are created using the NOLOGGING option, ensure that the table is backed up after the operation completes. Doing so will help to prevent data loss.

Copying an existing table

You use the CREATE TABLE AS SELECT (CTAS) command to allow tables to be created based upon a query. You can specify all regular storage parameters and block utilization parameters. Only the data and NOT NULL constraints on columns are carried through to the copied table. All other constraints, triggers, and table privileges are not copied. Here is an example:

```
CREATE TABLE TEMP_INSTRUCTORS TABLESPACE TOOLS
  STORAGE (INITIAL 10k NEXT 10k MINEXTENTS 2 PCTINCREASE 0)
  PCTFREE 20
  PCTUSED 40
  NOLOGGING
AS
  SELECT * FROM INSTRUCTORS;
```

This command creates a new table called TEMP_INSTRUCTORS in the tablespace TOOLS with its own extent storage and block utilization parameters. All the rows from the INSTRUCTORS table are copied to TEMP_INSTRUCTORS; none of the indexes, triggers, privileges, or other constraints is copied, however. To copy the table structure only, without the data, simply append a “WHERE 1=0” clause to the subquery:

```
CREATE TABLE TEMP_INSTRUCTORS TABLESPACE TOOLS
  STORAGE (INITIAL 10k NEXT 10k MINEXTENTS 2 PCTINCREASE 0)
  PCTFREE 20
  PCTUSED 40
  NOLOGGING
AS
  SELECT * FROM INSTRUCTORS
  WHERE 1 = 0;
```

```
SQL> DESC INSTRUCTORS
```

Name	Null?	Type
INSTRUCTORID	NOT NULL	NUMBER(38)
SALUTATION		CHAR(4)
LASTNAME	NOT NULL	VARCHAR2(30)
FIRSTNAME	NOT NULL	VARCHAR2(30)
MIDDLEINITIAL		VARCHAR2(5)
ADDRESS1		VARCHAR2(50)
ADDRESS2		VARCHAR2(50)

CITY		VARCHAR2(30)
STATE		CHAR(2)
COUNTRY		VARCHAR2(30)
POSTALCODE		CHAR(10)
OFFICEPHONE		CHAR(15)
HOMEPHONE		CHAR(15)
CELLPHONE		CHAR(15)
EMAIL		VARCHAR2(50)
INSTRUCTORTYPE	NOT NULL	CHAR(10)
PERDIEMCOST		NUMBER(9,2)
PERDIEMEXPENSES		NUMBER(9,2)
COMMENTS		VARCHAR2(2000)

```
SQL> DESC TEMP_INSTRUCTORS
```

Name	Null?	Type
-----	-----	-----
INSTRUCTORID	NOT NULL	NUMBER(38)
SALUTATION		CHAR(4)
LASTNAME	NOT NULL	VARCHAR2(30)
FIRSTNAME	NOT NULL	VARCHAR2(30)
MIDDLEINITIAL		VARCHAR2(5)
ADDRESS1		VARCHAR2(50)
ADDRESS2		VARCHAR2(50)
CITY		VARCHAR2(30)
STATE		CHAR(2)
COUNTRY		VARCHAR2(30)
POSTALCODE		CHAR(10)
OFFICEPHONE		CHAR(15)
HOMEPHONE		CHAR(15)
CELLPHONE		CHAR(15)
EMAIL		VARCHAR2(50)
INSTRUCTORTYPE	NOT NULL	CHAR(10)
PERDIEMCOST		NUMBER(9,2)
PERDIEMEXPENSES		NUMBER(9,2)
COMMENTS		VARCHAR2(2000)

Copying all columns is not necessary. If you need only a few columns from the INSTRUCTORS table, simply select only those columns:

```
CREATE TABLE TEMP_INSTRUCTORS TABLESPACE TOOLS
STORAGE (INITIAL 10k NEXT 10k MINEXTENTS 2 PCTINCREASE 0)
PCTFREE 20
PCTUSED 40
NOLOGGING
AS
SELECT INSTRUCTORID, SALUTATION, LASTNAME, FIRSTNAME
FROM INSTRUCTORS;
```

```
SQL> DESC TEMP_INSTRUCTORS
```

Name	Null?	Type
------	-------	------

INSTRUCTORID	NOT NULL	NUMBER(38)
SALUTATION		CHAR(4)
LASTNAME	NOT NULL	VARCHAR2(30)
FIRSTNAME	NOT NULL	VARCHAR2(30)

Creating tables using Oracle Schema Manager in OEM

Here are the steps to using the Oracle Schema Manager.

STEP BY STEP: Using the Oracle Schema Manager

1. Launch the DBA STUDIO.
Click Start ⇨ Programs ⇨ Oracle ⇨ Database Administration ⇨ DBA STUDIO.
2. Select the option to connect directly to a database rather than the Management Server and enter a username and password; then click OK.
3. Click the CERTDB database.
4. Enter the username SYSTEM with the password of MANAGER.
5. Select Object ⇨ Create from the menu bar.
6. Choose Table from the list and click the Use Wizard Option; then click Create.
7. Enter the table information, such as the name, column name and datatypes, constraint information, block utilization, and so on.
8. Create the table and then expand the Schema ⇨ Table ⇨ folder to verify that the table exists.

Temporary tables

Oracle introduced global temporary tables in Oracle8.1. The data is temporarily stored in the user's temporary tablespace for the duration of the user's session or transaction. The duration depends upon the option used during creation. The data is stored in the PGA memory for the session or in the user's temporary tablespace if not enough memory is available.

You use the CREATE GLOBAL TEMPORARY TABLE command to create temporary tables. Although the data is temporary, the definition of the table is permanent and can be removed only by using the DROP TABLE command. So, if one user creates a temporary table, all users can use it without any special privileges on the table. All users can insert data into a temporary table, but only that user who inserts data

can see his or her own data, and his or hers only. Because the data is stored in that user's PGA or temporary tablespace, it is not shared. Following is an example of using the CREATE GLOBAL TEMPORARY TABLE command:

```
SQL> CONNECT STUDENT/ORACLE
Connected.
SQL> CREATE GLOBAL TEMPORARY TABLE TEMP_TAB1 (X NUMBER);

SQL> INSERT INTO TEMP_TAB1 VALUES (999);

1 row created.

SQL> SELECT * FROM TEMP_TAB1;

           X
-----
          999
```

Now, from a different session, connect as another user and run the same query:

```
SQL> CONNECT SYSTEM/MANAGER
Connected.

SQL> SELECT * FROM STUDENT.TEMP_TAB1

no rows selected

SQL> INSERT INTO STUDENT.TEMP_TAB1 VALUES (888);

1 row created.

SQL> SELECT * FROM STUDENT.TEMP_TAB1;

           X
-----
          888
```

Now, return to the first session and select from TEMP_TAB1 again:

```
SQL> SELECT * FROM TEMP_TAB1;

           X
-----
          999
```

From this series of examples, you can see that although the table is accessible to all users, the data is private to the user who placed it there. The duration with which the data resides in the table depends upon the option used at creation time. The allowable values are the following: ON COMMIT DELETE ROWS, which specifies the duration of the data to be the duration of the transaction; and ON COMMIT PRESERVE ROWS, which specifies the duration of the data to be for the duration of the session. The default is ON COMMIT DELETE ROWS.

```
SQL> commit;

Commit complete.

SQL> SELECT * FROM TEMP_TAB1;

no rows selected
```

Following is an example of a temporary table using the ON COMMIT PRESERVE ROWS option:

```
SQL> CREATE GLOBAL TEMPORARY TABLE TEMP_TAB2 (X NUMBER)
2 ON COMMIT PRESERVE ROWS;

Table created.

SQL> INSERT INTO TEMP_TAB2 VALUES (777);

1 row created.

SQL> SELECT * FROM TEMP_TAB2;

      X
-----
     777

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM TEMP_TAB2;

      X
-----
     777
```

End the session by simply reconnecting; you can see that the rows are now gone:

```
SQL> CONNECT STUDENT/ORACLE
Connected.
SQL> SELECT * FROM TEMP_TAB2;

no rows selected
```

In the table creation, you can also use a subquery similar to the CTAS command:

```
SQL> CREATE GLOBAL TEMPORARY TABLE TEMP_LOCATIONS
2 ON COMMIT PRESERVE ROWS
3 AS
4 SELECT * FROM STUDENT.LOCATIONS;

Table created.
```

You can also create indexes on temporary tables, and the data for the index exists for the same durations as those for the table data, either for the session or transaction. You can also create views, triggers, procedures, and functions on temporary tables. Neither rollback nor redo information is recorded for temporary tables. You can export definitions but not any data of the temporary tables, even if you use the ROWS=Y option of the export. Currently, no means of exchanging a temporary table for a permanent table exists.

Temporary tables are a great way of storing information for query or batch processing on a temporary basis. If you'll be running large queries joining multiple tables many times in a batch job or report, you'll find it beneficial to run the query once, storing the results in a temporary table and then simply selecting from the temporary table. The query against the temporary table will run exponentially faster than the large query that is accessing multiple tables, because only one table needs to be queried in the case of the temporary table.

There is no such thing as a local temporary table.

Creating tables on clusters

You can use clusters for storing table data in a grouped manner. Tables that share the same cluster share the same data blocks, which is possible only with clusters. Regular tables cannot share data blocks with any other object. Typically, the tables that are clustered are tables that are commonly joined during queries or tables that can be accessed by the same value, called a Cluster Key. The advantage from a query performance standpoint is that fast access to data is provided through the cluster as opposed to an index. Accessing data through a cluster ranks higher in the Rule Based optimizer than do indexes. One type of cluster, a hash cluster, retrieves data at virtually the same speed as a ROWID lookup. So, tremendous performance gains can be achieved if clusters are properly implemented. They do, however, have their limitations and in general are not well received in the Oracle community because of those limitations. The biggest limitation to clusters is that a full table scan will almost always be slower with a cluster. The only time you benefit from clusters is when tables are being joined and the drawbacks of clusters seem to outweigh the improved join performance.

Types of clusters

There are two different types of clusters in Oracle and each is unique. The similarity between the two is that they are both stored in clusters.

Index clusters

Index clusters store data from multiple tables on the same block(s). The tables must have a common column, which is referred to as the Cluster Key. An index supports the Cluster Key values. The tables that are clustered are typically tables that are used together in joins. Figure 11-2 illustrates an index cluster. Good examples of tables that can be clustered are the LOCATIONS and INSTRUCTORS tables. These tables are almost always joined together when queried. If the tables are unclustered

and a query joins the two tables together where the LOCATIONID=100, the block from the LOCATIONS table containing the row for LOCATIONID 100 will need to be read as well as all blocks from the INSTRUCTORS table containing LOCATIONID 100. The index blocks for each table will also need to be read into the cache. As a result, Oracle may need to perform 5 or 6 IOs to satisfy the query. If you store the tables in an index cluster, Oracle scans the index on the Cluster Key to locate the cluster where the rows are stored. Then the block(s) that store the Cluster Key values is read into the database buffer cache. Because the data is grouped by the Cluster Key, multiple blocks will need to be read only if the size of the block is not large enough to support all the rows being stored for the common Cluster Key. The query will perform fewer units of IO being performed to satisfy the query. Some space will also be saved because the Cluster Key value is not repeated in the cluster. For example, if 15 INSTRUCTORS are working out of LOCATIONID 100, the value LOCATIONID=100 is stored only once in the cluster, as opposed to 15 times in the unclustered table example.

Before an index cluster can store table data, both the cluster and the index supporting the cluster must exist. The cluster and the index supporting the cluster should be located on separate tablespaces to minimize disk contention. The table storage parameters are specified at the cluster level and apply to all tables sharing the same cluster. The parameter SIZE can be set only in clusters. It defines the amount of space, in bytes, that all the rows sharing the same Cluster Key require. When you require a new Cluster Key value, such as when you enter a new order, Oracle allocates a chunk of space in the cluster equal to the parameter SIZE for storing all rows sharing this new Cluster Key value. If the SIZE parameter is set too high, space can be wasted. For example, if the SIZE is set to 1MB and on average all rows sharing the same Cluster Key average 100KB, then 900KB of space per Cluster Key is being wasted because the space inside the cluster cannot be shared with different Cluster Key values. So, to avoid wasting space in the cluster, carefully set the SIZE parameter to roughly the amount of space required to store all rows of the same Cluster Key. Following are the steps to create an index cluster.

STEP BY STEP: Creating an Index Cluster on the LOCATIONS and INSTRUCTORS Tables

1. Create the cluster specifying the Cluster Key and the SIZE for all common rows. You can use a bit of an educated guess because the actual row lengths are not always known or accurate.

```
SQL> CREATE CLUSTER CLUSTER_LOC_INST (LOCATIONID
NUMBER(5))
  2  SIZE 200
  3  TABLESPACE TOOLS
  4  STORAGE (INITIAL 40k NEXT 40k PCTINCREASE 0);
```

Cluster created.

2. Create the index for the cluster.

```
SQL> CREATE INDEX I_CLUSTER_LOC_INST
  2 ON CLUSTER CLUSTER_LOC_INST
  3 TABLESPACE INDX;
```

3. Create the tables for the cluster, making sure to specify the Cluster Key column. You do not have to give the column use the same name as that specified in the CREATE CLUSTER statement, but the column must use the same size and datatype.

```
SQL> CREATE TABLE LOCATION_CL (LOCATIONID NUMBER(5),
  2                               NAME VARCHAR2(30))
  3 CLUSTER CLUSTER_LOC_INST (LOCATIONID)
```

Table created.

```
SQL> CREATE TABLE INSTRUCTOR_CL (ID NUMBER(10),
  2                               ENAME VARCHAR2(100),
  3                               LOCATIONID NUMBER(5))
  4* CLUSTER CLUSTER_LOC_INST (LOCATIONID)
```

Table created.

Here are some points to remember about index clusters:

- ♦ Index clusters provide excellent access to multiple tables that share the same Cluster Key. Virtually any Primary Key / Foreign Key relationship can benefit from index clusters.
- ♦ Selects, updates, and deletes that use the Cluster Key process are faster than regular tables.
- ♦ A single insert into a clustered table performs slower than on a regular table.
- ♦ Loading through SQL-Loader or the Import Utility is very slow.
- ♦ Clusters cannot be loaded without the index; therefore, bulk loads must maintain the index information while loading. This will slow down the insert.
- ♦ More rollback and redo information is generated because of the index on the cluster.
- ♦ Tables with index clusters require more space than regular tables because blocks can store only rows sharing the same Cluster Key value.
- ♦ INSERT APPEND operations are not supported.

Hash clusters

Hash clusters offer very fast access to rows when the table is queried using an equality predicate on the Cluster Key column. Hash clusters do not require an index as Index clusters do; rather, Oracle builds the cluster using a predetermined number

of hash buckets, with each bucket being assigned a hash address. When rows are inserted into a table on a hash cluster, Oracle calculates the hash value on the Cluster Key column and determines which hash bucket to store the row in. When you query rows using the Cluster Key column, Oracle recalculates the hash value for the Cluster Key and knows which hash bucket the rows must exist in. Oracle does not need to perform an index or a full table scan. The difference between index clusters and hash clusters when the cluster is being created is that the number of hash buckets must be specified when the cluster is created. With index clusters, Oracle allocates another chunk of space equal to the SIZE parameter of the cluster every time a new Cluster Key is added to the cluster. With hash clusters, Oracle pre-allocates all the space for the cluster. At creation time, the parameter HASHKEYS determines the number of buckets to be created. Each hash bucket is the size specified by the parameter SIZE of the cluster. Therefore, if the SIZE is 1000 bytes and the HASHKEYS is 10000, the total amount of space allocated for this cluster before a single row is inserted into a table is $1000 * 10000 = 10,000,000$ bytes, or roughly 10MB. Figure 11-4 illustrates rows being inserted and queried in a hash cluster.

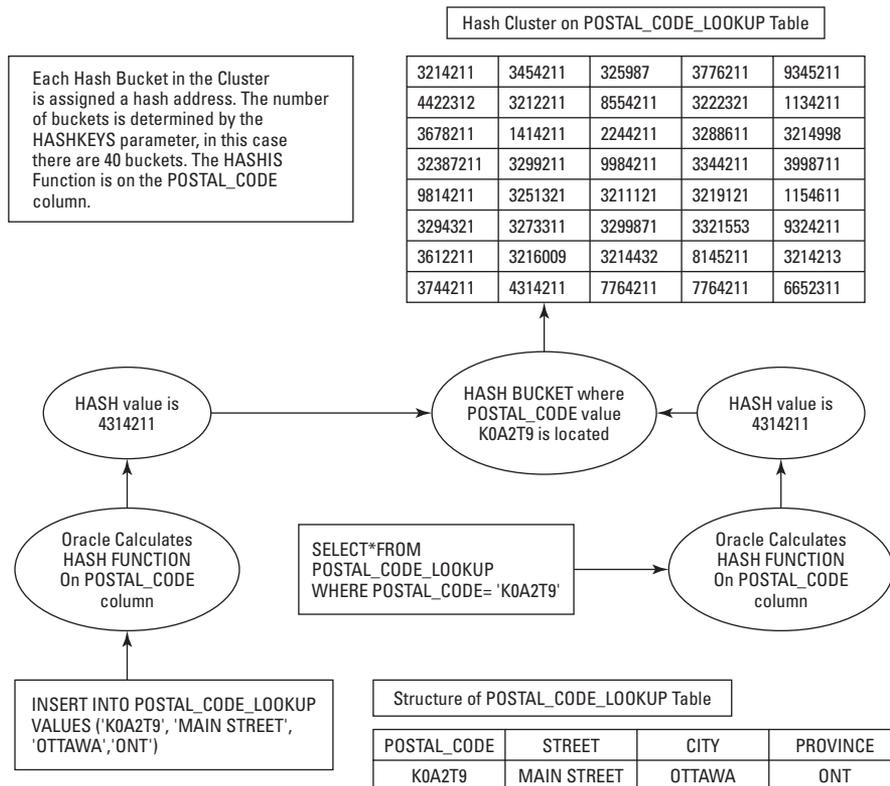


Figure 11-4: Hash clusters

You can see that each hash bucket is assigned a hash address. Oracle calculates the hash addresses when the cluster is created. When you insert rows, the hash address is calculated based upon the Cluster Key, in this case the `POSTAL_CODE` column. Oracle calculates the hash address and stores the row in that hash bucket. When you write a query using an equality predicate, such as `POSTAL_CODE`, on the Cluster Key column, Oracle simply recalculates the hash address and goes to that bucket to retrieve all rows on the bucket. No index is required.

Follow these steps for an example of creating a hash cluster.

STEP BY STEP: Creating and Using Hash Clusters

1. Create the cluster first. By specifying the `HASHKEYS` parameter, you make it a hash cluster. The `HASHKEYS` parameter should equal the number of unique Cluster Key values being stored. In this example, if 40 unique postal codes need to be stored, you should set `HASHKEYS` to 40. If only 20 are to be stored, then set the parameter to 20. Setting the number to a higher value than the actual number required will waste space.

```
CREATE CLUSTER "STUDENT"."POST_CODE_CL"
    (POSTAL_CODE VARCHAR2(6) )
    HASH IS POSTAL_CODE
    SIZE 120 TABLESPACE "CERTDB"
    HASHKEYS 40
```

2. Create the table specifying the Cluster Key column. It does NOT have to be the same name but must have the same datatype.

```
CREATE TABLE POST_CODE_LOOKUP (
    POSTAL_CODE VARCHAR2(6), STREET VARCHAR2(80),
    CITY VARCHAR2(30), PROVINCE VARCHAR2(3))
    CLUSTER POST_CODE_CL (POSTAL_CODE)
```

Hash clusters are complicated. There are several things that DBAs should be aware of with Hash Clusters:

- ♦ Hash clusters provide excellent access to a single row, provided that the query uses the Cluster Key value with an equality predicate. No indexes need to be scanned.
- ♦ Hash clusters provide excellent access to multiple records sharing the same Cluster Key.
- ♦ Selects, Updates, and Deletes that use the Cluster Key run faster on Hash Clusters run faster than on regular tables.
- ♦ Loading tables through SQL-Loader or the Import Utility is slower with Hash Clusters.

If you perform a query on a table in a hash cluster without using an equality predicate on the Cluster Key column, a full table scan is required.

- ♦ Space for the entire cluster is pre-allocated, so space may initially be wasted.
- ♦ The number of hash buckets for the HASHKEYS parameter must be known ahead of creation because adding buckets is impossible without dropping and re-creating the entire cluster.
- ♦ Full table scans will be slower on Hash Clusters.
- ♦ Although they appear seamless to the user, hash clusters are difficult to maintain, modify, or both.

Guidelines for creating tables

Objective

Control the space used by tables

Several rules exist for creating tables, and the following sections describe some that you should follow.

Choose appropriate storage settings

Choosing the correct extent size for a table is very important. The size affects performance in two ways. If the extent size is small in relation to the table, many extents are required. If extents need to be created while users are attempting to insert rows, this process will dramatically slow down those users. You should avoid dynamic extent creation and you can do so by creating larger extents. A myth exists in the Oracle community about the number of extents that a table should have. Many in the Oracle community feel that each table should have only one extent. The idea behind the myth is that with only one extent, queries will be faster and extents will not be dynamically created. In fact, the number of extents has virtually no effect on query performance. If rows are being retrieved via a ROWID, as is the case with index lookups, then Oracle retrieves the row based upon the datafile number, block number, and row number. It makes no difference whether there is one extent or 1000. So, does the myth contain any truth? Well, some, but not for the reasons that most people think. The only time that the extent size effects performance is during full table scans, when every row in a table is read. If the table has one extent, the extent map for the table can be scanned in a single pass. If the table has many extents, Oracle may require multiple scans of the extent map during the full table scan. The problem with full table scan performance occurs when extents are not multiples of the initialization parameter `DB_FILE_MULTIBLOCK_READ_COUNT`. This parameter controls how many blocks are read in every IO pass during a full table scan. If the parameter is set to 20 blocks and extents are not multiples of 20 blocks, Oracle will need to go to multiple extents during each IO request to satisfy the number of blocks required. For example, if the extent size for a table is 15 blocks, the table has 10 extents, and the parameter `DB_FILE_MULTIBLOCK_READ_COUNT` is set to 20 blocks, then the following sequence happens:

1. Every IO pass needs 20 blocks, as determined by `DB_FILE_MULTIBLOCK_READ_COUNT`. This scan will require 8 IO passes to read in all blocks for the table (15 blocks per extent * 10 extents / 20 = 7.5). The first IO pass needs 20 blocks, so the server process goes to the extent map of the table being processed, locates the extent information, and starts reading in blocks. The first extent that it reads, however, has only 15 blocks but needs 20. So, it jumps to the second extent and reads in the last five blocks that it needs for the first IO. It now has 20 blocks.
2. The second IO pass needs 20 blocks as well. It goes to the second extent and reads in 10 blocks before reaching the end of the extent. Still needing 10 more, it goes to the next extent.
3. This process continues a total of 8 times until all blocks have been read. Each IO pass is forced to bounce to multiple extents to satisfy the required number of blocks.

In this example, if the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter had been 15 blocks or the extents were 20 blocks in size, then the bounces to the extent map would have been avoided and the query would have completed quicker.

So, the important thing to remember is that the number of extents is not what affects performance; rather, the size of the extents does. Extents should always be a multiple of `DB_FILE_MULTIBLOCK_READ_COUNT * DB_BLOCK_SIZE`, and `DB_FILE_MULTIBLOCK_READ_COUNT` should almost always be a multiple of 64KB. Consult your OS guidelines for the appropriate setting for this parameter.

When choosing the size of the extents for a table, pick a few standard sizes, such as 128KB, 1MB, and 10MB, and always set `PCTINCREASE` to zero. Set `INITIAL = NEXT` and make sure that their values are multiples of `DB_FILE_MULTIBLOCK_READ_COUNT * DB_BLOCK_SIZE`. By doing so, you ensure that extents are multiples of one another and full table scan performance will be optimized.

Table data and indexes

Tables should always be placed on data-only tablespaces. Avoid mixing tables with indexes, temporary segments, and rollback segments. When you access tables using an index, Oracle reads the index to obtain the ROWID of the row being processed. With the ROWID, Oracle then reads the table. If a table and the indexes for that table are on the same tablespace, IO conflicts may occur. Certain constraints, such as PRIMARY KEY and UNIQUE constraints, are maintained with indexes. Make sure that these indexes are not on the same tablespace as the table. You can use a special `USING INDEX` option of the constraint creation syntax to specify the location of the constraint indexes, or you can simply use the `ALTER INDEX REBUILD` command to relocate indexes onto a different tablespace.



Chapter 13 covers constraints, and Chapter 12 covers the ALTER INDEX REBUILD command.

Rollback segments and temporary segments have a tendency to grow and shrink, leading to fragmentation. Therefore, place tables in their own tablespace.

Use locally managed tablespaces

Locally managed tablespaces offer several benefits. Space is managed at the data-file level in a special bitmap that identifies free space. In dictionary-managed tablespaces, free space is managed in the data dictionary. Moving the free space management to the data-file level reduces contention in the data dictionary when space is required. The other benefit to locally managed tablespaces is that all extent sizes are uniform, regardless of specifications in the CREATE TABLE statement. If the extent size for a locally managed tablespace is 1MB and a table is created specifying an initial extent of 128KB, the extent will be 1MB. The default storage parameters supersede those specified in the table. If an initial extent size is specified at 4MB in the CREATE TABLE statement, then 4 extents of 1MB are created in the locally managed tablespace.



Most DBAs are now choosing to use locally managed tablespaces for simplicity and performance. Create three different types of tablespaces per application and set the size to 128KB, 1MB, and 10MB appropriately; then, ensure that small tables are placed on the 128KB tablespace, tables less than 10MB are placed on the 1MB tablespace, and large tables go on the 10MB tablespace. Placing large tables (tables larger than 100MB) on their own tablespace is also a good idea.

Here are some miscellaneous guidelines to observe:

- ♦ Use multiple free lists for tables that are subject to many concurrent inserts. If a table has multiple users performing concurrent inserts, contention can happen for the free list of that table. You can specify free lists only at table creation time, so adding them requires dropping and re-creating the table. You also can add them using the ALTER TABLE MOVE command.
- ♦ Have one or a couple of users who own tables in the database. The only user who can assign object privileges on a table is the owner of that table. Therefore, if a user creates a table and the DBA wants to grant privileges on that table to another user, the DBA must get permission from the owner of the table. The owner of the table can revoke privileges from other users as well, which may cause problems with applications. Therefore, for ease of maintenance, do not let users create tables in a production environment. Have them provide scripts with the table definitions. Having these scripts will let you specify storage parameters, decide upon a tablespace, and eventually control permissions on the table.
- ♦ Do not use the COMPRESS option of the Export Utility. The compress option will change extent sizes when you import objects.

Setting PCTFREE and PCTUSED

The parameters PCTFREE and PCTUSED control block utilization and storage. They can be specified only at the segment or table level. You can't set defaults for PCTFREE or PCTUSED at the tablespace. If you don't set the parameters, they use the system defaults, as follows: PCTFREE=10 percent and PCTUSED=40 percent.

PCTFREE

PCTFREE controls the amount of free space that is reserved on a block for updates and helps control ROW MIGRATION. You can specify, as a number, a value between 1 and 99 percent. A value of 10 percent for PCTFREE reserves 10 percent of the block as free space. When less than 10 percent of the block is "free," the block is removed from the free list. The free list is a list of blocks for each table that are available for insert. If a block is not on the free list, no more rows will be inserted. Only UPDATE and DELETE operations will be possible.

If the database block size is 8,192 bytes and the PCTFREE value is 10 percent, then roughly 800 bytes are reserved for free space. Subtracting block header information, about 7200 bytes remain available per block for row data. If the average row size is 1200 bytes, the block will support approximately seven rows before the free space left on the block goes down to less than 10 percent or 800 bytes. After exceeding this threshold, Oracle removes the block from the free list. The seven rows on the block can grow, through updates, to a total of 800 additional bytes before no free space is left on the block. If no space is left on a block and a row is updated, Oracle will "migrate" the row to another block, leaving behind a forwarding address to the new block location.



Migration is discussed later in this chapter in the section "Controlling migration and chaining."

If, however, rows are rarely updated, the 800 bytes of free space become wasted space. If a table has a PCTFREE setting of 20 percent, the block size is 8192 bytes, an average row's size is 500 bytes, and the table has 100,000 rows, then the table will have roughly 7820 blocks total. The calculation is as follows:

$$\begin{aligned} & (\text{AVG_ROW_SIZE} * \text{NUM_ROWS}) / ((\text{BLOCKSIZE} - 200 (\text{HEADER}) * (1 - \text{PCTFREE})) \\ & (500 * 100000) / ((8192 - 200) * (1 - .20)) = 7820 \end{aligned}$$

Roughly 1600 bytes of space are reserved on each block for free space. Multiplying 1600 bytes times the number of blocks, 7,820, means that 12, 512, 000 bytes, or roughly 12MB, of space is reserved. If the rows are not updated, this 12MB amounts to wasted space.

You should always pack blocks as tightly as possible, but not so tightly as to increase the likelihood of migration. Following is an example similar to the previous one, this time with PCTFREE set to 2 percent rather than 20 percent because rows are infrequently updated. With a PCTFREE of 2 percent, the total number of blocks is

$$(500 * 100000) / ((8192 - 200) * (1 - .02)) = 6383$$

With this setting, 1437 fewer blocks would need to be scanned during a full table scan.

You must strike a balance between reducing migration and using space efficiently. Migration is very expensive and should always be avoided by setting PCTFREE to an appropriate level. The calculation for PCTFREE is as follows:

$$\frac{(\text{Average Row Size} - \text{Initial Row Size}) * 100}{\text{Average Row Size}}$$

Where Initial Row Size is the average value of rows when they are inserted and Average Row Size is the size of rows after updates have taken place

If you plug some number into the calculation like; Average Row Size = 100 and Initial Row Size = 80 then:

$$\frac{(100 - 80)}{(100)} = 20 \text{ Percent}$$

PCTUSED

After a data block becomes full as determined by PCTFREE, Oracle does not consider the block for the insertion of new rows until the percentage of the block being used falls below the parameter PCTUSED. Before this value is achieved, Oracle uses the free space of the data block only for updates to rows already contained in the data block. If the table does not have deletes or updates, the setting of PCTUSED is insignificant because it controls only when blocks are placed back on the free list. A value of 40 percent means that the block will not be placed back on the free list until less than 40 percent of it is being used. The 40 percent does not include the space used by the block header; 40 percent is the default value.

You want to ensure that blocks are returned to the free list only when enough free space is available for at least one row, and preferably two rows, to be inserted. The average row size is stored in the column AVG_ROW_LEN of the DBA_TABLE view after an ANALYZE is performed. If a block on the free list does not contain enough space for the row being inserted, Oracle continues scanning the free list until Oracle finds either a block with sufficient space or the end of the free list. If the scan reaches the end of the free list, Oracle is forced to put more blocks on the free list. This may involve creating a new extent if no free blocks are available. The DBA_TABLES data dictionary view has a column called FREE_BLOCKS that lists the number of free blocks available for the table.

Figure 11-5 illustrates the effects of the PCTFREE and PCTUSED parameters on block utilization and storage. You can see that when an insert or update leaves less free space on the block than was specified by the PCTFREE parameter, that block is removed from the free list. The remaining free space is used for updates to the

existing rows. When a delete or update operation frees space on the block and less space is being used on the block than that specified by PCTUSED, the block is returned to the free list and is available to Oracle for inserts again.

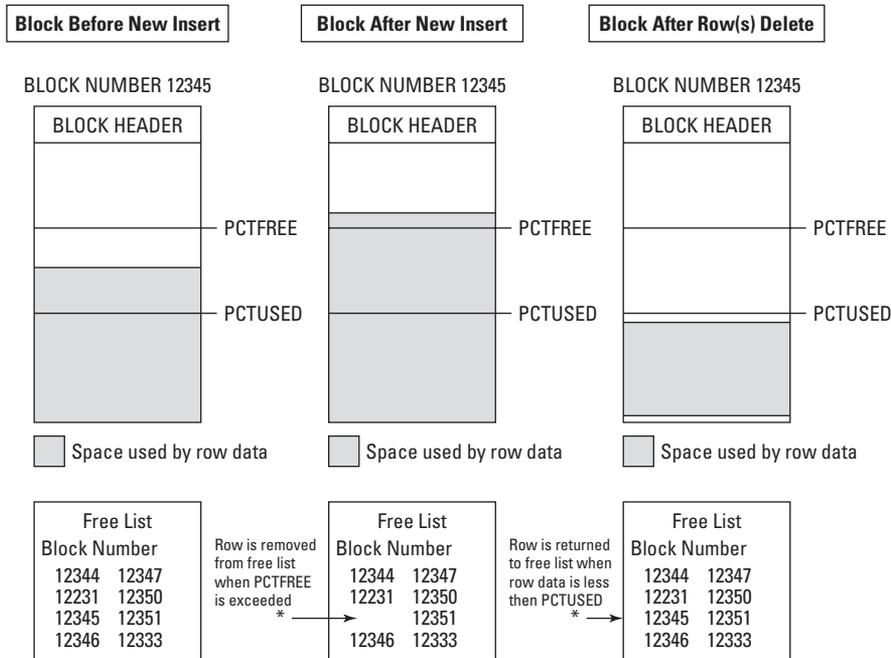


Figure 11-5: Effects of PCTFREE and PCTUSED on block utilization and storage

Row migration and chaining

Objective

Analyze tables to check integrity and migration

Using row migration and chaining means that not all pieces of a row are stored on the same block.

Row migration

Row migration happens when not enough free space is left on a block to satisfy an update. For example, a table that has a VARCHAR2(4000) column can store up to 4000 bytes. Assume that the current value being stored in the column occupies 1000 bytes. Potentially 3000 more bytes of information can be stored in that column. If a user updates the column value, adding 500 bytes to the current value, then 500 bytes of free space must be left on the block to hold the updated value. If only 300 bytes are left on the block, Oracle is forced to migrate the entire row to a new block. Oracle goes to the free list and finds a block with enough free space to hold the entire row. It then moves the entire row to the new block and leaves behind a forwarding address on the original block that points to the new block. So,

the ROWID does not change and the index is not updated. Now when this row is being queried, Oracle will have to go to the original block, see a forwarding address there, and go to the new row location on the new block. This situation creates extra IO that can be avoided with appropriate setting of the PCTFREE value. Migration happens only during updates. Figure 11-6 illustrates row migration.

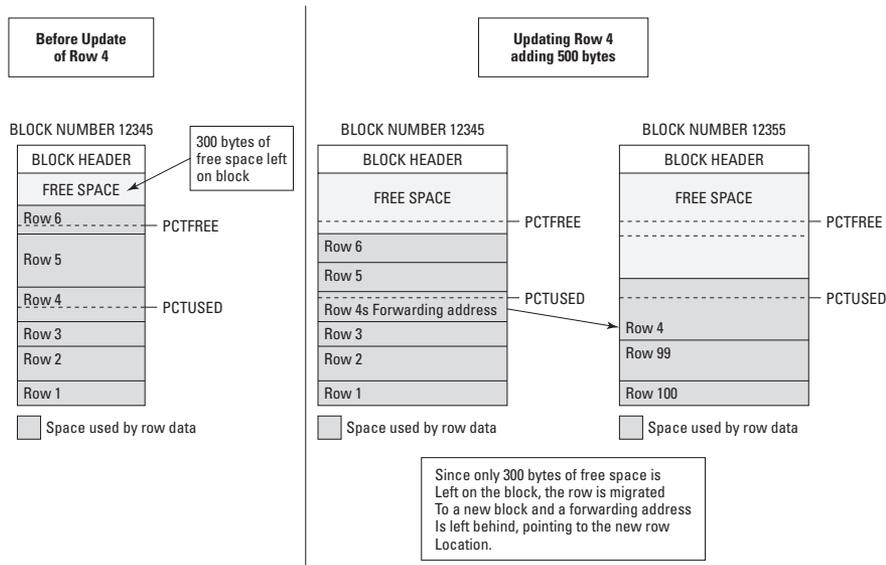


Figure 11-6: Row migration

Row chaining

Row chaining can occur on either inserts or updates. It most often occurs on inserts when the row being inserted is bigger than the block size. For example, a row that is 6,000 bytes in length cannot fit onto a 4,000-byte block. Therefore, Oracle chains the row together over multiple blocks. Pieces of the row will exist on as many blocks as required. Having an update greater than the block size also causes the row to be chained.



Updates cause chaining only when the updated value increases the row size to a value greater than the block size. If the updated value is greater only than the free space left on the block and not the entire row, Oracle will migrate the row, not chain it.

Controlling migration and chaining

You can control migration by appropriately setting the value of PCTFREE. If the value of PCTFREE is low, such as five percent, Oracle reserves only five percent of the block for updates to existing rows. If updates to rows are frequent, that five percent will be quickly used up, leaving no free space on the block. When this happens,

Oracle is forced to migrate. If PCTFREE is set to 30 percent, Oracle reserves 30 percent of the block for updates. Oracle will stop inserting new rows on the block when less than 30 percent is free, which leaves all that space for updates. If you're using a 4KB block size, 30 percent translates to roughly 1200 bytes. Now, migration will not happen until the 1200 bytes are used.



Refer to the section “Setting PCTFREE and PCTUSED,” earlier in this chapter, for the correct setting of the PCTFREE value.

The drawback to setting PCTFREE to higher values is that space is reserved and potentially wasted. If no updates occur to the rows on a block, that 1,200 bytes of free space will never be used. If a table has 10,000 blocks with zero updates, then $10,000 \text{ (blocks)} * 1,200 \text{ (bytes)} = 12,000,000 \text{ bytes}$, or roughly 12MB, of space is wasted. You must choose a trade-off between wasting space on the block and row migration. Because row migration is very expensive in terms of query performance, most DBAs lean toward reserving the free space.

You can control chaining only by increasing the block size for the database or by splitting the table into multiple tables. Increasing the database block size can be accomplished only by rebuilding the entire database.

Detecting row migration and chaining

The DBA_TABLES data dictionary view contains a column called CHAIN_CNT that reports the number of rows that are chained, migrated, or both. Oracle does not differentiate between chaining and migration inside the database. The CHAIN_CNT is populated after Oracle runs the ANALYZE TABLE command. Here is an example of a table with migrated rows:

```
SQL> CREATE TABLE TEST_MIG (X NUMBER(10), Y CHAR(2000)) TABLESPACE TOOLS;
```

```
Table created.
```

```
# Insert 100 rows
```

```
SQL> BEGIN
2   FOR I IN 1..100 LOOP
3     INSERT INTO TEST_MIG VALUES (I,NULL);
4   END LOOP;
5 END;
6 /
```

```
PL/SQL procedure successfully completed.
```

```
SQL> ANALYZE TABLE TEST_MIG COMPUTE STATISTICS;
```

```
Table analyzed.
```

```

SQL> SELECT TABLE_NAME,CHAIN_CNT
       2 FROM   USER_TABLES
       3 WHERE  TABLE_NAME='TEST_MIG';

TABLE_NAME                                CHAIN_CNT
-----
TEST_MIG                                    0

# Update the column Y that will force migration

SQL> UPDATE TEST_MIG
       2 SET   Y=RPAD('X',1000,'X');

100 rows updated.

SQL> ANALYZE TABLE TEST_MIG COMPUTE STATISTICS;

Table analyzed.

# After the update, you can see that there are 97 of the 100
# rows that are chained or migrated.

SQL> SELECT TABLE_NAME,CHAIN_CNT
       2 FROM   USER_TABLES
       3 WHERE  TABLE_NAME='TEST_MIG';

TABLE_NAME                                CHAIN_CNT
-----
TEST_MIG                                    97

```

A number of ways exist to correct migrated rows. The ALTER TABLE *table_name* MOVE command is by far the easiest.



The ALTER TABLE *table_name* MOVE command is covered later in this chapter in the section called “Moving tables.”

Modifying Tables

Modifying tables is one of a DBAs more common tasks. Outlined in the sections below are some of the common modifications that are made to tables.

Altering storage parameters

You can change virtually all storage parameters and block utilization parameters with the three exceptions of INITIAL, FREE LISTS, and the TABLESPACE where the table is located. The important thing to understand is when the storage parameters will take effect. For example, changing the value of PCTREE does not cause Oracle to examine every block in the table to determine whether it should be removed

from the free list. It simply examines all future blocks when it modifies rows. Table 11-5 lists the storage and block utilization parameters that you can change using the ALTER TABLE command. The table also describes when the parameter takes effect.

Table 11-5
Options of the ALTER TABLE Command

<i>Parameter</i>	<i>When it Takes Effect</i>
PCTFREE	Changes in PCTFREE affect only future inserts and updates. Blocks already removed from the free list will not be placed back on if they no longer exceed the value of PCTFREE. Also, blocks currently on the free list will not be taken off immediately if they exceed the value of PCTFREE. The next insert or update statement that exceeds the limit of PCTFREE will force the block to be removed from the free list.
PCTUSED	As with PCTFREE, changing PCTUSED does not have an immediate effect on the blocks. Future update or delete operations will investigate the block to see whether it can now be placed back on the free list.
INITRANS	Changes affect only new blocks.
MAXTRANS	Changes all blocks in the table.
LOGGING	Affects future LOGGING/NOLOGGING operations.
NOLOGGING	Affects future LOGGING/NOLOGGING operations.
CACHE	Affects future full table scans.
NOCACHE	Affects future full table scans.
STORAGE CLAUSE	
NEXT	Affects the next extent created. The second extent for the table will be created the size of NEXT and extents after the third will be created by taking the value of NEXT * PCTINCREASE. If more than two extents already exist for a table, the next extent is created using the value of NEXT, and the subsequent extents are created using the value of at NEXT * PCTINCREASE.
MINEXTENTS	MINEXTENTS must be greater than or equal to the current number of extents and will take effect only when the table is reorganized.
MAXEXTENTS	MAXEXTENTS must be greater than or equal to the current number of extents and affects all future extents.
PCTINCREASE	Affects future extents created after the second. If NEXT is specified along with PCTINCREASE, the subsequent extent is created using the value of NEXT and all other extents are created using NEXT * PCTINCREASE.
BUFFER POOL	A change to the buffer pool is effective only the next time the database is restarted.

The following example modifies the LOCATIONS table, setting the value of the next extent created to 1MB. The maximum number of extents is unlimited and the PCTINCREASE parameter is set to zero. The example also shows modified block storage parameters of PCTFREE and PCTUSED.



If the table is stored in a locally managed tablespace or the tablespace has the MINIMUM EXTENT parameter set, these circumstances will affect the actual size of the extents when created. If the extent size for a locally managed tablespace is set to 10MB and you attempt to set NEXT to 50MB, 5–10MB extents will be created when the next one is required. If you set the value to 100KB, the next extent will be created at 10MB even though you set it to 100KB.

```
ALTER TABLE LOCATIONS
  STORAGE ( NEXT 1M MAXEXTENTS UNLIMITED PCTINCREASE 0)
  PCTFREE 20
  PCTUSED 50;
```

The following example attempts to modify the value of INITIAL for the LOCATIONS table.

```
SQL> ALTER TABLE LOCATIONS
  2  STORAGE ( INITIAL 2M);

STORAGE ( INITIAL 2M)
          *
ERROR at line 2:
ORA-02203: INITIAL storage options not allowed
```

Manually allocating extents

Oracle allocates extents when no more free blocks are available and an extent is available. Allocating and extent happens when Oracle scans the free list looking for a block to insert a row and none is available. If Oracle finds no “empty blocks,” it allocates a new extent. The problem with Oracle allocating extents dynamically is that it is an expensive operation: it slows down the insert. DBAs can manually create additional extents to avoid this dynamic allocation. DBAs will also manually create extents to try to distribute extents over multiple datafiles to reduce IO contention.

One of the DBA’s jobs is to ensure that as little dynamic extension as possible happens inside the database. To determine the likelihood that a table will need to extend, the DBA must consider three factors: the frequency of inserts, the size of the inserted rows, and the number of empty blocks left in the table. The frequency of inserts is usually derived from historical analysis, but the size of the inserted rows and the number of empty blocks is stored in the DBA_TABLES data dictionary view and is populated after the ANALYZE TABLE command is run.

```
SQL> ANALYZE TABLE CLASSENROLLMENT COMPUTE STATISTICS;
```

Table analyzed.

```
SQL> SELECT TABLE_NAME,BLOCKS,EMPTY_BLOCKS,AVG_ROW_LEN
2 FROM USER_TABLES
3 WHERE TABLE_NAME='CLASSENROLLMENT';
```

TABLE_NAME	BLOCKS	EMPTY_BLOCKS	AVG_ROW_LEN
CLASSENROLLMENT	1	3	43

The preceding example contains three empty blocks. When these blocks are used, Oracle will need to dynamically extend. If you expect a very high volume of inserts in the CLASSENROLLMENT table, allocating more space by manually allocating a new extent may be a good idea, as follows:

```
SQL> ALTER TABLE CLASSENROLLMENT ALLOCATE EXTENT (SIZE 40K);
```

Table altered.

```
SQL> ANALYZE TABLE CLASSENROLLMENT COMPUTE STATISTICS;
```

Table analyzed.

```
SQL> SELECT TABLE_NAME,BLOCKS,EMPTY_BLOCKS,AVG_ROW_LEN
2 FROM USER_TABLES
3 WHERE TABLE_NAME='CLASSENROLLMENT';
```

TABLE_NAME	BLOCKS	EMPTY_BLOCKS	AVG_ROW_LEN
CLASSENROLLMENT	1	8	43

With the extent created, eight blocks are now available. Oracle will not need to extend now until those eight blocks are used.

Here is the syntax of the ALTER TABLE command for manually allocating extents:

```
ALTER TABLE [schema.]table
ALLOCATE EXTENT [ ([SIZE integer [K|M]]
[ DATAFILE 'filename' ]) ]
```

If the created extent exceeds the value of MAXEXTENTS, the command will fail. If you don't include the SIZE parameter, the size of the extent is determined by the value of NEXT_EXTENT in the DBA_TABLES data dictionary view. The SIZE can be specified in kilobytes or megabytes and will be rounded up to the next multiple of DB_BLOCK_SIZE if the value is not a multiple of DB_BLOCK_SIZE.

If the `DATAFILE` clause is included then the datafile specified must belong to the same tablespace as the table. If the `DATAFILE` is not included, and there are multiple datafiles for the tablespace, then Oracle creates the extent on the next datafile in the sequence of datafiles. Oracle will try and distribute the extents over all the datafiles as the extents are created.

Moving tables

Oracle8.1 introduced a way of moving tables. You can move a table from one tablespace to another or rebuild it on the same tablespace that reorganizes the table. Reorganizing tables fixes the problems with migration. You also can use the `move table` command to change the values of extent sizes such as `INITIAL` and `NEXT`, and to change the number of `FREE LISTS` for a table. All the storage parameters that you can specify during table creation can be specified using the `MOVE` command, which makes this an extremely powerful tool for DBAs. Here is the syntax of the `ALTER TABLE table_name MOVE` command:

```
ALTER TABLE [schema.] table_name MOVE
  [ INITRANS integer ]
  [ MAXTRANS integer ]
  [ LOGGING | NOLOGGING ]
  [ CACHE | NOCACHE ]
  [STORAGE ( [ INITIAL integer [K|M] ]
             [ NEXT integer [K|M] ]
             [ MINEXTENTS integer ]
             [ MAXEXTENTS integer | UNLIMITED ]
             [ PCTINC [TABLESPACE tablespace_name ]
             [ PCTFREE integer ]
             [ PCTUSED integer ]
           REASE integer ]
             [ FREE LISTS integer ]
             [ BUFFER POOL KEEP | RECYLCE | DEFAULT ]
           )]
```

Following is an example of the command:

```
ALTER TABLE SCHEDULEDCLASSES MOVE;

Table Altered.
```

This example simply re-creates the table on the same tablespace. The existing storage and block utilization parameters will be used.

In the following example, the table `TEST_MIG` that was created in the previous section called “Chaining and Migration” is rebuilt to fix the migration problem:

```
SQL> CREATE TABLE TEST_MIG (X NUMBER(10), Y CHAR(2000))
TABLESPACE TOOLS;

Table created.
```

```

SQL>
SQL>
SQL> BEGIN
  2   FOR I IN 1..100 LOOP
  3     INSERT INTO TEST_MIG VALUES (I,NULL);
  4   END LOOP;
  5 END;
  6 /

```

PL/SQL procedure successfully completed.

```

SQL>
SQL>
SQL> ANALYZE TABLE TEST_MIG COMPUTE STATISTICS;

```

Table analyzed.

```

SQL>
SQL>
SQL> SELECT TABLE_NAME,CHAIN_CNT
  2 FROM   USER_TABLES
  3 WHERE  TABLE_NAME='TEST_MIG';

```

TABLE_NAME	CHAIN_CNT
TEST_MIG	0

```

SQL>
SQL> UPDATE TEST_MIG
  2 SET   Y=RPAD('X',1000,'X');

```

100 rows updated.

```

SQL>
SQL> ANALYZE TABLE TEST_MIG COMPUTE STATISTICS;

```

Table analyzed.

```

SQL> SELECT TABLE_NAME,CHAIN_CNT
  2 FROM   USER_TABLES
  3 WHERE  TABLE_NAME='TEST_MIG';

```

TABLE_NAME	CHAIN_CNT
TEST_MIG	97

ALTER THE TABLE AND MOVE IT TO REORGANIZE.

```

SQL> ALTER TABLE TEST_MIG MOVE
  2 ;

```

Table altered.

```
SQL> ANALYZE TABLE TEST_MIG COMPUTE STATISTICS;
```

NOTICE THAT AFTER THE COMMAND, THERE ARE NO MIGRATED ROWS

```
SQL> SELECT TABLE_NAME,CHAIN_CNT
2 FROM USER_TABLES
3 WHERE TABLE_NAME='TEST_MIG';
```

TABLE_NAME	CHAIN_CNT
TEST_MIG	0

After you have moved a table, all indexes on the table will need to be rebuilt because the ROWIDs have changed; however, all program units such as procedures, packages, functions, and triggers will *not* be invalidated.

```
SQL> ALTER TABLE STUDENT.SCHEDULEDCLASSES MOVE;
```

Table altered.

```
SQL> SELECT CLASSID,COURSENUMBER
2 FROM STUDENT.SCHEDULEDCLASSES
3 WHERE CLASSID = 50;
```

```
SELECT *
*
```

ERROR at line 1:

ORA-01502: index 'STUDENT.PK_CLASSID' or partition of such index is in unusable state

```
SQL> ALTER INDEX STUDENT.PK_CLASSID REBUILD;
```

Index altered.

```
SQL> SELECT CLASSID,COURSENUMBER
2 FROM STUDENT.SCHEDULEDCLASSES
3 WHERE CLASSID = 50;
```

CLASSID	COURSENUMBER
100	

Handling unused space

Objective

Control the space used by tables

From time to time, DBAs may need to deallocate space from a table because too much space was initially allocated or maybe because the nature of the table has changed. The estimated data volume may not be what it was expected, and the DBA wants to use the space allocated for the table for other tables. The DBA is limited to deallocating space above the high-water mark only unless the entire table is being rebuilt.

The high-water mark

The high-water mark for a table indicates the last block that was ever placed on the free list for each table. The high-water mark is the level at which blocks have never been set as ready to receive data. As data is inserted and new blocks are required, the high-water mark will continue to move out. The location of the high-water mark tells Oracle which blocks need to be read during a full table scan. Blocks after the high-water mark cannot contain rows because they would have never been placed on the free list for that table. The high-water mark never moves back when deletes are performed. It represents the “highest” point in the table where data existed at one time or another. The high-water mark is stored in the header of the table and can be determined by one of two ways. The column `BLOCKS` in the `DBA_TABLES` view represents the high-water mark but requires the `ANALYZE TABLE` command to be performed. Or, the `DBMS_SPACE.UNUSED_SPACE` package can be used without the `ANALYZE TABLE`.

The `DBA_TABLES` method

The `BLOCKS` column represents the high-water mark and the `EMPTY_BLOCKS` represents the free blocks, or “unused blocks,” after or above the high-water mark in the `DBA_TABLES` data dictionary view. You must first run the `ANALYZE TABLE` command, as follows:

```
SQL> ANALYZE TABLE TEST_MIG ESTIMATE STATISTICS
```

```
Table analyzed.
```

```
SQL> SELECT TABLE_NAME, BLOCKS, EMPTY_BLOCKS
2 FROM DBA_TABLES
3 WHERE TABLE_NAME='TEST_MIG';
```

TABLE_NAME	BLOCKS	EMPTY_BLOCKS
TEST_MIG	34	1

Because the value of `BLOCKS` is 34, you know that the high-water mark is at 34 blocks. One empty block is available. Now, you can see that the high-water mark will move up or out to the right as rows are added, increasing the number of required blocks.

```
SQL> INSERT INTO TEST_MIG SELECT * FROM TEST_MIG;
```

```
100 rows created.
```

```
SQL> ANALYZE TABLE TEST_MIG ESTIMATE STATISTICS;
```

```
Table analyzed.
```

```
SQL> SELECT TABLE_NAME, BLOCKS, EMPTY_BLOCKS
2 FROM DBA_TABLES
3 WHERE TABLE_NAME='TEST_MIG';
```

TABLE_NAME	BLOCKS	EMPTY_BLOCKS
TEST_MIG	71	0

Notice that even when all rows in the table are deleted, the high-water mark never moves down. The only reason that the high-water mark ever moves down is if the table is reorganized or the table is truncated.

```
SQL> DELETE FROM TEST_MIG;
```

```
200 rows deleted.
```

```
SQL> ANALYZE TABLE TEST_MIG ESTIMATE STATISTICS;
```

```
Table analyzed.
```

```
SQL> SELECT TABLE_NAME, BLOCKS, EMPTY_BLOCKS
2 FROM DBA_TABLES
3 WHERE TABLE_NAME='TEST_MIG';
```

TABLE_NAME	BLOCKS	EMPTY_BLOCKS
TEST_MIG	71	0

The DBMS_SPACE.UNUSED_SPACE method

The DBMS_SPACE.UNUSED_SPACE procedure contains the same information as DBA_TABLES but the ANALYZE TABLE command is not required. Here is a list of the input and output variables required to execute the procedure:

Argument Name	Type	In/Out Default?
SEGMENT_OWNER	VARCHAR2	IN
SEGMENT_NAME	VARCHAR2	IN
SEGMENT_TYPE	VARCHAR2	IN
TOTAL_BLOCKS	NUMBER	OUT
TOTAL_BYTES	NUMBER	OUT
UNUSED_BLOCKS	NUMBER	OUT

UNUSED_BYTES	NUMBER	OUT	
LAST_USED_EXTENT_FILE_ID	NUMBER	OUT	
LAST_USED_EXTENT_BLOCK_ID	NUMBER	OUT	
LAST_USED_BLOCK	NUMBER	OUT	
PARTITION_NAME	VARCHAR2	IN	DEFAULT

Following is an example of using the procedure to determine the location of the high-water mark and the number of free or unused blocks:

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2   V_OWNER          VARCHAR2(30) := 'SYS';
  3   V_SEGMENT       VARCHAR2(30) := 'TEST_MIG';
  4   V_SEGMENT_TYPE  VARCHAR2(30) := 'TABLE';
  5   V_TOTAL_BLOCKS  NUMBER;
  6   V_TOTAL_BYTES   NUMBER;
  7   V_UNUSED_BLOCKS NUMBER;
  8   V_UNUSED_BYTES  NUMBER;
  9   V_LAST_EXT_FILE_ID    NUMBER;
 10  V_LAST_EXT_BLOCK_ID   NUMBER;
 11  V_LAST_USED_BLOCK     NUMBER;
 12  BEGIN
 13    DBMS_SPACE.UNUSED_SPACE (
 14      V_OWNER
 15    , V_SEGMENT
 16    , V_SEGMENT_TYPE
 17    , V_TOTAL_BLOCKS
 18    , V_TOTAL_BYTES
 19    , V_UNUSED_BLOCKS
 20    , V_UNUSED_BYTES
 21    , V_LAST_EXT_FILE_ID
 22    , V_LAST_EXT_BLOCK_ID
 23    , V_LAST_USED_BLOCK
 24    );
 25    DBMS_OUTPUT.PUT_LINE ( V_SEGMENT||' TOTAL BLOCKS:
'|V_TOTAL_BLOCKS||' WITH
'|V_UNUSED_BLOCKS||' EMPTY BLOCKS');
 26  END;
 27  /

```

```
TEST_MIG TOTAL BLOCKS: 88 WITH 20 EMPTY BLOCKS
```

PL/SQL procedure successfully completed.

In this example, you can see that a total of 88 blocks have been allocated to the TEST_MIG table, with 20 free or unused. The high-water mark would therefore be on block 68.

Deallocating unused space above the high-water mark

You can deallocate space only above the high-water mark. You must do it manually for tables because Oracle deallocates space automatically only for rollback segments. Here is the syntax for the DEALLOCATE COMMAND:

```
ALTER TABLE [schema.]table_name
DEALLOCATE UNUSED [KEEP integer [K|M]]
```

The KEEP option specifies the number of bytes, using K (for kilobytes) or M (for megabytes) to retain or keep above the high-water mark. Without the KEEP option, Oracle deallocates all space down to the high-water mark only if the high-water mark is at an extent equal to or above the number of extents specified by the MINEXTENTS parameter. Otherwise, Oracle deallocates, or releases, all space above the MINEXTENTS value.

In the following example, a table called TEST_DEALLOC contains 127 blocks and 16 empty blocks. The only space that can be deallocated, or freed, is the 16 blocks. In this example, the first DEALLOCATE COMMAND uses the KEEP option and the second does not, to demonstrate the differences.

```
# BEFORE SPACE IS DEALLOCATED, 16 EMPTY BLOCKS

SQL> SELECT TABLE_NAME, BLOCKS, EMPTY_BLOCKS
 2 FROM DBA_TABLES
 3 WHERE TABLE_NAME='TEST_DEALLOC';

TABLE_NAME                                BLOCKS  EMPTY_BLOCKS
-----
TEST_DEALLOC                               127      16

# DEALLOCATE SPACE BUT LEAVE 64 KILOBYTES.

SQL> ALTER TABLE TEST_DEALLOC DEALLOCATE UNUSED KEEP 64K;

Table altered.

SQL> ANALYZE TABLE TEST_DEALLOC ESTIMATE STATISTICS;

Table analyzed.

SQL> SELECT TABLE_NAME, BLOCKS, EMPTY_BLOCKS
 2 FROM DBA_TABLES
 3 WHERE TABLE_NAME='TEST_DEALLOC'
 4 /

TABLE_NAME                                BLOCKS  EMPTY_BLOCKS
-----
TEST_DEALLOC                               127      8

# DEALLOCATE ALL SPACE ABOVE THE HIGH-WATER MARK
```

```
SQL> ALTER TABLE TEST_DEALLOC DEALLOCATE UNUSED;
```

Table altered.

```
SQL> ANALYZE TABLE TEST_DEALLOC ESTIMATE STATISTICS;
```

Table analyzed.

```
SQL> SELECT TABLE_NAME, BLOCKS, EMPTY_BLOCKS
2 FROM DBA_TABLES
3 WHERE TABLE_NAME='TEST_DEALLOC'
4 /
```

TABLE_NAME	BLOCKS	EMPTY_BLOCKS
TEST_DEALLOC	127	0

Truncating tables

The `TRUNCATE TABLE` command deletes all rows in the table and corresponding indexes but does not remove the table definition from the data dictionary. Using this command is the fastest way to delete all the rows in a table because no Rollback or Redo information is generated. Some redo is generated, which will be propagated to a standby database. Be careful, though: Because this is a DDL statement, it has no `ROLLBACK` command. You must be the owner of the table or have the `TRUNCATE ANY TABLE` system privilege to truncate a table in another user's schema. All space, including that below the high-water mark, is released and new extents are created. The number of extents created is determined by the parameter `MINEXTENTS`, and the size of the extents is determined by the value of `INITIAL`, `NEXT`, and `PCTINCREASE`. Here is the syntax:

```
TRUNCATE TABLE [schema.] table_name
[ { DROP | REUSE } STORAGE]
```

where the `REUSE` option saves or retains the space used by the table but deletes all rows. The high-water mark is also moved back. If you use the `DROP` option, which is the default, Oracle drops all extents and creates new ones, with the number being determined by `MINEXTENTS`.

The following is a list of things to remember about the `TRUNCATE` command:

- ♦ All rows are deleted.
- ♦ Oracle resets the high-water mark to the first block in the table.
- ♦ The `MINEXTENTS` parameter determines the number of extents.
- ♦ The command is a DDL statement; therefore, you cannot `ROLLBACK` the command.

- ♦ You need to release all space used by indexes on the table.
- ♦ If any referential integrity constraints are pointing the table, the table cannot be truncated. Drop or disable the constraints before issuing the TRUNCATE command.
- ♦ Delete triggers do not fire.

Dropping a table

You can drop tables if you no longer need them or are reorganizing your tables. The DROP TABLE command deallocates all space held by the table and removes the table definition from the data dictionary. You must either be the owner of the table or have the DROP ANY TABLE system privilege to drop a table owned by another user. Here is the syntax for the command:

```
DROP TABLE [schema. ]table_name
[CASCADE CONSTRAINTS]
```

If any FOREIGN KEY constraints are referencing the table, the CASCADE CONSTRAINTS option is required. This option drops all FOREIGN KEY constraints referencing the table.

Dropping a column

New in Oracle8.1 is the ability to drop a column in a table. Before Oracle8.1, all DBAs could do was set the column value to NULL through an update statement. You must be the owner of the table or have the DROP ANY COLUMN privilege to drop a column in another user's table. The DROP COLUMN command both removes the data, freeing up space in a table, and removes the column definition from the data dictionary. Here is the syntax for the DROP COLUMN command:

```
ALTER TABLE [schema. ]table_name
DROP COLUMN column_name
[CASCADE CONSTRAINTS]
[CHECKPOINT integer]
```

The CASCADE CONSTRAINTS option is required if any FOREIGN KEY constraints reference this column. The CHECKPOINT parameter allows you to specify a frequency by which checkpoints will occur during the DROP. When you drop a column, Oracle deletes all the data for that column from the table. On large tables, this can take a lot of time and resources, such as rollback space. By using the CHECKPOINT option, the amount of rollback space required for the command to complete can be significantly reduced. The integer is a number of rows. Here is an example:

```
ALTER TABLE STUDENT.CLASSENROLLMENT
DROP COLUMN COMMENTS
CHECKPOINT 200;
```

This example drops the COMMENTS column from the CLASSENROLLMENT table and performs a checkpoint after every 200 rows. If the table contained thousands of rows, a large rollback segment would be required to complete this command without using the CHECKPOINT option. During the process of dropping the column, the table is locked and marked INVALID until the operation completes. If the database should fail during the operation, you can restart the command by using a special CONTINUE option of the ALTER TABLE command, as follows:

```
ALTER TABLE [schema. ]table_name
DROP COLUMNS CONTINUE;
```

Because the table is locked during this operation and many resources are required, performing this command in off hours when the system is not busy may be advisable. If the column absolutely must be dropped, consider using the SET UNUSED COLUMN option of the ALTER TABLE COMMAND instead. This command simply removes the column definition from the data dictionary without actually removing the data. Initially, this command will not free any space; it will let applications continue processing as if the column does not exist. During off hours, the columns that have been flagged as UNUSED can be dropped. Here is the syntax for setting a column to UNUSED:

```
ALTER TABLE [schema. ]table_name
SET UNUSED COLUMN column_name
[CASCADE CONSTRAINTS]
```

By using a special option of the ALTER TABLE command, you can drop columns that have been set to UNUSED. Here is the syntax:

```
ALTER TABLE [schema. ]table_name
DROP UNUSED COLUMNS
[CHECKPOINT integer]
```

This command will drop ALL columns that have been set to UNUSED. The data dictionary view DBA_UNUSED_COL_TABS lists all tables with UNUSED columns.



After a column has been set to UNUSED, the data is no longer available. This is a DDL statement with no ability to ROLLBACK and without a “set column USED” comment to revert back to using a column. So, even though the data is still in the block, you can’t use it without performing some form of database recovery.

Here is an example of setting a column to a status of UNUSED and then dropping the UNUSED columns:

```
CREATE TABLE EMP
(EMPID NUMBER CONSTRAINT EMP_EMPID_PK PRIMARY KEY,
 ENAME VARCHAR2(50) NOT NULL,
 SAL NUMBER,
 JOB VARCHAR2(15),
 COMMENTS VARCHAR2(100)) TABLESPACE TOOLS
/
```

```
SQL> DESC EMP
Name                               Null?    Type
-----
EMPID                               NOT NULL NUMBER
ENAME                               NOT NULL VARCHAR2(50)
SAL                                  NUMBER
JOB                                  VARCHAR2(15)
COMMENTS                            VARCHAR2(100)
```

```
# INSERT A ROW OF DATA
```

```
SQL> INSERT INTO EMPLOYEEVALUES (
2 123
3 , 'TODD ROSS'
4 ,1000
5 , 'INSTRUCTOR'
6 , 'EMPLOYEE REVIEW DATE IS JANUARY 1ST, 2002');
```

```
# SET THE COMMENTS COLUMN TO BE UNUSED
```

```
SQL> ALTER TABLE STUDENT.EMP
2 SET UNUSED COLUMN COMMENTS;
```

```
Table altered.
```

```
SQL>
SQL> DESC STUDENT.EMP
Name                               Null?    Type
-----
EMPID                               NOT NULL NUMBER
ENAME                               NOT NULL VARCHAR2(50)
SAL                                  NUMBER
JOB                                  VARCHAR2(15)
```

```
# NOW, DROP ALL UNUSED COLUMNS FROM THE TABLE. THIS WILL
# FREE UP ALL THE SPACE BEING USED BY THOSE COLUMNS
```

```
SQL> ALTER TABLE STUDENT.EMP
2 DROP UNUSED COLUMNS;
```

```
Table altered.
```

One advantage of the SET COLUMNS UNUSED command is when multiple columns are being dropped. If you use the DROP COLUMN command, the table and blocks are being processed once for each column being dropped. If three columns are being dropped, the table will be locked three separate times. When the SET COLUMNS UNUSED command is used in conjunction with the DROP UNUSED COLUMNS command, the table is locked and processed only once, with all the UNUSED columns being removed.

You can't do the following with the DROP COLUMN command:

- ♦ Drop all columns in a table
- ♦ Drop a partitioning key column
- ♦ Drop a column from a table owned by SYS
- ♦ Drop a parent key column
- ♦ Drop a column from an indexed-organized table if the column is the PRIMARY KEY
- ♦ Drop a column from an object type table
- ♦ Drop columns from nested tables

Retrieving Table Information

Objective

Retrieve information about tables from the data dictionary

Five main data dictionary views contain table information. The following sections cover these views.

DBA_OBJECTS

The DBA_OBJECTS view contains data on all objects (tables, views, procedures, packages, package bodies, and so on) in the database. Table 11-6 contains the column names, datatypes, and column descriptions for DBA_OBJECTS.

Table 11-6
DBA_OBJECTS Data Dictionary View

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
OWNER	VARCHAR2(30)	Object owner.
OBJECT_NAME	VARCHAR2(128)	Object name.
SUBOBJECT_NAME	VARCHAR2(30)	Subobject name for partitions.
OBJECT_ID	NUMBER	Object number for object.
DATA_OBJECT_ID	NUMBER	Object Number of the segment that contains the object. This is the value stored in the ROWID for each table.
OBJECT_TYPE	VARCHAR2(18)	Object type.
CREATED	DATE	Date created.

Continued

Table 11-6 (continued)

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
LAST_DDL_TIME	DATE	Date of last DDL or DCL statement performed.
TIMESTAMP	VARCHAR2(19)	Timestamp for the specification of the object.
STATUS	VARCHAR2(7)	Status of the object (VALID,INVALID). Invalid objects need to be recompiled or rebuilt in the case of an index.
TEMPORARY	VARCHAR2(1)	Is this a temporary table (Y,N)?
GENERATED	VARCHAR2(1)	Was the name of the object system generated? This is often the case for constraints and indexes.
SECONDARY	VARCHAR2(1)	Is this a secondary object created as part of icreate for domain indexes?

DBA_TABLES

The DBA_TABLES data dictionary view contains all the information about the physical structure of the table and the block utilization parameters. It also contains the statistical information for the table used by the Cost-Based optimizer. The statistics column requires you to run the ANALYZE TABLE command. Table 11-7 contains the column names, datatype and a description of each column for the DBA_TABLES data dictionary view.

Table 11-7
DBA_TABLES Data Dictionary View

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
OWNER	VARCHAR2(30)	Owner of table.
TABLE_NAME	VARCHAR2(30)	Table name.
TABLESPACE_NAME	VARCHAR2(30)	Tablespace where table is located.
CLUSTER_NAME	VARCHAR2(30)	Cluster name where the table is located if this is a clustered table.
IOT_NAME	VARCHAR2(30)	Index-organized table name, if any, for the Overflow Segment.
PCT_FREE	NUMBER	PCTFREE setting for the table.
PCT_USED	NUMBER	PCTUSED setting for the table.

Column Name	Datatype	Description
INI_TRANS	NUMBER	INITRANS setting for the table.
MAX_TRANS	NUMBER	MAXTRANS setting for the table.
INITIAL_EXTENT	NUMBER	Size of INITIAL extent for the table. Size is in bytes.
NEXT_EXTENT	NUMBER	Value for the NEXT extent to be created. The next extent created will be this size, which is expressed in bytes.
MIN_EXTENTS	NUMBER	Number of extents created when the table was created or the number that will be created when the table is rebuilt.
MAX_EXTENTS	NUMBER	Maximum number of extents for this table.
PCT_INCREASE	NUMBER	PCTINCREASE setting for the table. The value is a percentage.
FREELISTS	NUMBER	Number of free lists for the table.
FREELIST_GROUPS	NUMBER	Number of free list groups. This is available only for Oracle Parallel Server.
LOGGING	VARCHAR2(3)	Is logging enabled or disabled for the table? Values: YES NO
BACKED_UP	VARCHAR2(1)	Has the table been backed up since last DDL statement? Values: Y N
NUM_ROWS	NUMBER	Number of rows in the table. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added or deleted.
BLOCKS	NUMBER	Number of blocks that have been placed on the free list. This value also represents the high-water mark for the table. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added.
EMPTY_BLOCKS	NUMBER	Number of blocks that have not been placed on the free list. This value represents the number of blocks after the high-water mark. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added or deleted.
AVG_SPACE	NUMBER	The average available free space per block in the table. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added, updated, or deleted.

Continued

Table 11-7 (continued)

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
CHAIN_CNT	NUMBER	Number of chained or migrated rows. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added, updated, or deleted.
AVG_ROW_LEN	NUMBER	Average row length. This includes row header, column length, and column data in the calculation. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added, updated, or deleted.
AVG_SPACE_FREELIST_BLOCKS	NUMBER	Average free space of all blocks on free list. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added, updated, or deleted.
NUM_FREELIST_BLOCKS	NUMBER	Number of blocks on the free list. This value is populated only during the ANALYZE TABLE command and is not updated as new rows are added, updated, or deleted.
DEGREE	VARCHAR2(10)	The default degree of parallelism for the table.
INSTANCES	VARCHAR2(10)	The number of instances across which the table is to be scanned.
CACHE	VARCHAR2(5)	Is this a CACHED table? Values: Y N
TABLE_LOCK	VARCHAR2(8)	Is table locking enabled or disabled? Values: ENABLED DISABLED
SAMPLE_SIZE	NUMBER	The sample size used during the estimate statistics command.
LAST_ANALYZED	DATE	The date the table was last analyzed.
PARTITIONED	VARCHAR2(3)	Is this a partitioned table? Values: YES NO
IOT_TYPE	VARCHAR2(12)	Type of index-organized table. Values: IOT IOT_OVERFLOW NULL
TEMPORARY	VARCHAR2(1)	Is this a temporary table? Values: Y N
SECONDARY	VARCHAR2(1)	Is the table object created as part of icreate for domain indexes?

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
NESTED	VARCHAR2(3)	Is this a nested table? Values: YES NO
BUFFER_POOL	VARCHAR2(7)	Buffer pool for this table. Values: KEEP RECYCLE DEFAULT
ROW_MOVEMENT	VARCHAR2(8)	If this is a partitioned table, is row movement enabled for the partition key column?
GLOBAL_STATS	VARCHAR2(3)	Are the statistics calculated without merging the underlying partitions?
USER_STATS	VARCHAR2(3)	Were the statistics entered directly by the user? Values: NO YES
DURATION	VARCHAR2(15)	If this is a temporary table, what is the duration of the data? Values: SYS\$SESSION SYS\$TRANSACTION NULL for permanent
SKIP_CORRUPT	VARCHAR2(8)	Should blocks marked as corrupt be skipped? Values: ENABLED DISABLED
MONITORING	VARCHAR2(3)	Should the amount of modification be kept? Values: YES NO

DBA_SEGMENTS

The DBA_SEGMENTS data dictionary view contains a row for every segment in the database. Table 11-8 contains the column names, datatype and a description of each column for the DBA_SEGMENTS data dictionary view.

Table 11-8
DBA_SEGMENTS Data Dictionary View

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
OWNER	VARCHAR2(30)	Segment owner.
SEGMENT_NAME	VARCHAR2(81)	Segment name.
PARTITION_NAME	VARCHAR2(30)	Partition name.

Continued

Table 11-8 (continued)

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
SEGMENT_TYPE	VARCHAR2(18)	Segment type. Values: CACHE CLUSTER INDEX LOBINDEXT LOBSEGMENT NESTED TABLE ROLLBACK TABLE TEMPORARY
TABLESPACE_NAME	VARCHAR2(30)	Tablespace name where the segment is located.
HEADER_FILE	NUMBER	File Id of the file that contains the segment header.
HEADER_BLOCK	NUMBER	Block Id of the block that contains the segment header.
BYTES	NUMBER	Size, in bytes, of the segments. This value is populated only after the ANALYZE TABLE command is used.
BLOCKS	NUMBER	Size, in blocks, of the segment. This value is populated only after the ANALYZE TABLE command is used.
EXTENTS	NUMBER	Number of extents for this segment.
INITIAL_EXTENT	NUMBER	The size of the INITIAL parameter, which is the size of the first extent created for the table. The value is specified in BYTES.
NEXT_EXTENT	NUMBER	Value for the NEXT extent to be created. The next extent created will be this size, which is expressed in bytes.
MIN_EXTENTS	NUMBER	Number of extents created when the table was created, or the number that will be created when the table is rebuilt.
MAX_EXTENTS	NUMBER	Maximum number of extents for this table.
PCT_INCREASE	NUMBER	PCTINCREASE setting for the table. The value is a percentage.
FREELISTS	NUMBER	Number of free lists for the table.
FREELIST_GROUPS	NUMBER	Number of free list groups. This is available only for Oracle Parallel Server.
RELATIVE_FNO	NUMBER	Relative file number containing the segment header.
BUFFER_POOL	VARCHAR2(7)	Buffer pool for this table. Values: KEEP RECYCLE DEFAULT

DBA_EXTENTS

The DBA_EXTENTS data dictionary view contains one row for every extent in the database. The view lists the size of the extent as well as the datafile where the extent resides. Grouping by the OWNER and SEGMENT_NAME provides the number of extents for the table. Table 11-9 contains the column names, datatype and a description of each column for the DBA_EXTENTS data dictionary view.

Table 11-9
DBA_EXTENTS Data Dictionary View

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
OWNER	VARCHAR2(30)	Extent owner, which is also the owner of the segment.
SEGMENT_NAME	VARCHAR2(81)	Segment name that the extent belongs to.
PARTITION_NAME	VARCHAR2(30)	Partition name that the extent belongs to.
SEGMENT_TYPE	VARCHAR2(18)	Segment type that the extent belongs to. Values: CACHE CLUSTER INDEX LOBINDEX LOBSEGMENT NESTED TABLE ROLLBACK TABLE TEMPORARY
TABLESPACE_NAME	VARCHAR2(30)	Tablespace that the extent belongs to.
EXTENT_ID	NUMBER	Unique extent number for the segment.
FILE_ID	NUMBER	File Id that this extent belongs to.
BLOCK_ID	NUMBER	Starting Block Id for this extent.
BYTES	NUMBER	Size of the extent in bytes.
BLOCKS	NUMBER	Size of the extent in blocks.
RELATIVE_FNO	NUMBER	Relative file number that this extent belongs to.

Here is a query that you can run to determine the number of extents that a table has:

```
SELECT      SEGMENT_NAME, COUNT(*) "NUMBER OF EXTENTS"
FROM        DBA_EXTENTS
WHERE       SEGMENT_NAME = 'SCHEDULEDCLASSES'
GROUP BY   SEGMENT_NAME;
```

This query will list all the extents for table, the size of the extents in both bytes and blocks, and the datafile that the extent belongs to.

```

SELECT      SEGMENT_NAME, EXTENT_ID, FILE_ID,
BYTES "SIZE IN BYTES", BLOCKS "SIZE IN BLOCKS"
FROM        DBA_EXTENTS
WHERE       SEGMENT_NAME = 'SCHEDULEDCLASSES';

```

DBA_TAB_COLUMNS

The `DBA_TAB_COLUMNS` view has one row for every column in the database. This includes the datatype as well as statistical information. The statistical information is populated only when the `ANALYZE TABLE` command is run. (See Table 11-10.)

Table 11-10
DBA_COLUMNS Data Dictionary View

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
OWNER	VARCHAR2(30)	Owner of the table.
TABLE_NAME	VARCHAR2(30)	Table name.
COLUMN_NAME	VARCHAR2(30)	Column name.
DATA_TYPE	VARCHAR2(106)	Datatype.
DATA_TYPE_MOD	VARCHAR2(3)	Datatype modifier of the column.
DATA_TYPE_OWNER	VARCHAR2(30)	Owner of the column datatype.
DATA_LENGTH	NUMBER	Length of the column in bytes.
DATA_PRECISION	NUMBER	Length in decimal digits (NUMBER) or binary digits (FLOAT).
DATA_SCALE	NUMBER	Digits to the right of the decimal, or the scale.
NULLABLE	VARCHAR2(1)	Does the column allow nulls?
COLUMN_ID	NUMBER	Sequence number of the column as it is created..
DEFAULT_LENGTH	NUMBER	Length of the default value for the column.
DATA_DEFAULT	LONG	Default value, if any, for the column.
NUM_DISTINCT	NUMBER	Number of distinct values. This column is populated only during the <code>ANALYZE TABLE</code> command.
LOW_VALUE	RAW(32)	Low value for the column represented as a raw number. This column is populated only during the <code>ANALYZE TABLE</code> command.
HIGH_VALUE	RAW(32)	High value for the column represented as a raw number. This column is populated only during the <code>ANALYZE TABLE</code> command.

<i>Column Name</i>	<i>Datatype</i>	<i>Description</i>
DENSITY	NUMBER	Density of the column. This column is populated only during the ANALYZE TABLE command.
NUM_NULLS	NUMBER	Number of nulls in this column. This column is populated only during the ANALYZE TABLE command.
NUM_BUCKETS	NUMBER	Number of buckets for histograms. This column is populated only during the ANALYZE TABLE command.
LAST_ANALYZED	DATE	Date that the column statistics were last generated.
SAMPLE_SIZE	NUMBER	The sample size used during the ESTIMATE STATISTICS command.
CHARACTER_SET_	NAME	VARCHAR2(44) Character set for NCHAR, NVARCHAR2, and NCLOB columns.
CHAR_COL_DECL_LENGTH	NUMBER	Declaration length of the character type column.
GLOBAL_STATS	VARCHAR2(3)	Are the statistics calculated without merging the underlying partitions?
USER_STATS	VARCHAR2(3)	Did the user enter the statistics manually?
AVG_COL_LEN	NUMBER	The average column length, represented in bytes. This column is populated only during the ANALYZE TABLE command.

Here is a query that can be run to list the column for a table:

```
SELECT      COLUMN_NAME
FROM        DBA_TAB_COLUMNS
WHERE      TABLE_NAME = 'SCHEDULEDCLASSES';
```

DBMS_ROWID Package

Objective

Convert between different formats of ROWID

Oracle provides a package for converting Oracle8 ROWIDs to Oracle7, or Oracle7 to Oracle8 ROWIDs. The package also has procedures for extracting individual components of the ROWID. Table 11-11 outlines the procedures of the package.

Table 11-11
DBMS_ROWID Package

<i>Function</i>	<i>Description</i>
ROWID_CREATE	Creates a ROWID from the individual components specified.
ROWID_OBJECT	Returns the object identifier for a ROWID. The object identifier is located in the DATA_OBJECT_ID column of the DBA_OBJECTS view.
ROWID_RELATIVE_FNO	Returns the relative file number for a ROWID. The relative file number identifies the datafile that the row belongs to. The relative file number is relative to the data object id.
ROWID_BLOCK_NUMBER	Returns the block number for a ROWID.
ROWID_ROW_NUMBER	Returns the row number for a ROWID.
ROWID_TO_ABSOLUTE_FNO	Returns the absolute file number for a ROWID. The absolute file number is stored in the FILE_ID column of the DBA_DATA_FILES data dictionary.
ROWID_TO_EXTENDED	Converts a "Restricted" ROWID to an "Extended" ROWID.
ROWID_TO_RESTRICTED	Converts an "Extended" ROWID to a "Restricted" ROWID.

You can use the DBMS_ROWID package in a query to obtain the physical location of the rows in the SCHEDULEDCLASSES table:

```
SQL> SELECT      CLASSID, ROWID,
2      DBMS_ROWID.ROWID_OBJECT(ROWID) "OBJ ID",
3      DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) "REL FNO",
4      DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) "BLOCK",
5      DBMS_ROWID.ROWID_ROW_NUMBER(ROWID) "ROW NUMBER",
6      DBMS_ROWID.ROWID_TO_ABSOLUTE_FNO(ROWID,
7      'STUDENT', 'SCHEDULEDCLASSES') "ABS FNO"
8 FROM          STUDENT.SCHEDULEDCLASSES;
```

CLASSID	ROWID	OBJ ID	REL FNO	BLOCK	ROW	ABS FNO
50	AAAFE1AAHAAAAArAAA	20773	7	43	0	7
51	AAAFE1AAHAAAAArAAB	20773	7	43	1	7
53	AAAFE1AAHAAAAArAAC	20773	7	43	2	7

Key Point Summary

In preparing for the Oracle8i DBA: Architecture and Administration exam, please keep these points in mind regarding managing tables:

- ♦ There is more than one type of table in Oracle. Each type serves a unique purpose. There are regular tables, index-organized tables, partitioned and clustered tables, and temporary tables. Only regular tables and temporary tables are covered on the exam.
- ♦ There are several different data types that are available to a DBA. Be sure to know the advantages of VARCHAR2 and CHAR as well as LONG vs. LOB datatypes.
- ♦ The structure of the ROWID changed adding the object number to the ROWID.
- ♦ When creating tables, it is important to understand how the storage options work.
- ♦ The MINIMUM EXTENT option, if specified at the tablespace level, or locally managed tablespaces are the only two times that storage parameter values are overridden.
- ♦ Small tables can be cached. By caching tables they tend to stay in memory longer resulting in fewer IO operations.
- ♦ The only time it is safe to use the NOLOGGING option when creating tables is if the data can be recovered or if the data does not need to be recovered. The NOLOGGING option is significantly faster than LOGGING for large tables.
- ♦ The CREATE TABLE AS SELECT command can be used to copy an existing table to a new table or to just copy certain columns and/or rows to a new table. This command only copies the table definition and NOT NULL constraints.
- ♦ Temporary tables have permanent definitions in the data dictionary and temporary data. The data stays in the table for either the duration of the transaction or session depending upon what was specified when the table was created.
- ♦ Pay careful attention to the setting of PCTFREE. Setting too high can lead to row migration and chaining, while setting too low can waste space.
- ♦ PCTUSED controls when blocks are made available again for inserts after the block has been removed from the free list.
- ♦ Use the ANALYZE TABLE command to determine the number of chained and or migrated rows. Excessive chaining and migration can lead to poor performance and should be fixed.

- ♦ Tables can be reorganized by exporting and importing them using the Export and Import utility or the table can be rebuilt using the MOVE command.
- ♦ The ALTER TABLE MOVE command allows a table to be rebuilt in a different tablespace for the purpose of redistributing data or in the same tablespace for the purpose of reorganizing it.
- ♦ Unused space, that is space above the high-water mark can only be deallocated. The ALTER TABLE table_name DEALLOCATE UNUSED command to deallocate space.
- ♦ Use the ANALYZE table command along with the DBA_TABLES data dictionary view or the DBMS_SPACE.UNUSED_SPACE package to find the high-water mark.
- ♦ Columns can be dropped or set to a status of UNUSED. UNUSED columns are no longer visible to the user, however, no space is freed up as no data is actually deleted. If and when the column is dropped, the space is freed.
- ♦ There is a comprehensive list of data dictionary views that contain information about tables. They are the DBA_TABLES, DBA_OBJECTS, DBA_SEGMENTS, DBA_EXTENTS, DBA_TAB_COLUMNS.
- ♦ The DBMS_ROWID package can be used to extract information from the ORACLE8.1 ROWID.



STUDY GUIDE

Now that you have learned about tables, you should test your understanding by reviewing the assessment questions and performing the following exercises.

Assessment Questions

1. What is true about columns that use a varchar2 datatype?
 - A. The maximum size is 2000 bytes.
 - B. The maximum size is 4000 bytes.
 - C. The column data is padded out with spaces.
 - D. These columns cannot store numeric data.
 - E. These columns cannot be modified once created.
2. List two advantages that LOB datatypes have over LONG. (Pick two answers.)
 - A. LOB datatypes can be up to 2GB in size.
 - B. A SELECT returns the DATA.
 - C. You can have multiple LOB column datatypes in the same table.
 - D. LOB datatypes can be up to 6GB in size.
 - E. LOB datatypes can be up to 4GB in size.
3. What is not a component of the new ROWID format in Oracle8?
 - A. Data object number
 - B. Row number
 - C. Relative file number
 - D. File number
 - E. Block number
4. What does the PCTFREE block utilization parameter control?
 - A. The percentage of the row that is free
 - B. The percentage of the table that is free
 - C. The amount of free space reserved in a table for rows to grow
 - D. The amount of free space reserved in a block for rows to grow
 - E. When blocks are returned to the free list

5. What is the system default value for PCTFREE?
- A. 40
 - B. 10
 - C. 90
 - D. There is no system default; only tablespace defaults.
 - E. There is no system default; a value can be specified only during table creation or when tables are being altered.
6. What effect does setting the PCTINCREASE value to 100 have on extents belonging to tables created on regular tablespaces that are not locally managed and that do not have the MINIMUM EXTENT parameter defaulted at the tablespace level?
- A. PCTINCREASE is a block utilization parameter and therefore has no effect on extent sizes.
 - B. Setting PCTINCREASE to 100 percent means that every extent will be twice the size of the previous extent.
 - C. Setting PCTINCREASE to 100 percent means that every extent will be 100 times the size of the previous extent.
 - D. Setting PCTINCREASE to 100 percent means that every extent after the second extent will be twice the size of the previous extent.
 - E. Setting PCTINCREASE to 100 percent means that every extent after the second extent will be 100 times the size of the previous extent.
7. What effect does creating a table using the NOLOGGING option have on tables?
- A. No rollback or redo information will ever be kept about the table.
 - B. Inserts, updates, deletes, and the actual CREATE TABLE commands are not recorded in the redo log files.
 - C. The table is created from a script and not from recovery logs.
 - D. The CREATE TABLE statement and certain types of data loads will not be recorded in the redo log files.
 - E. Both A and D.
8. What is true about temporary tables in Oracle?
- A. The table definition and the table data are kept either for the duration of the user's session or the duration of the transaction.
 - B. Temporary tables can only be queried. No DML statements are permitted.
 - C. Temporary tables can only be created by the super user.
 - D. The table definition can be removed only with a DROP TABLE command.

- E. Users can share data in a temporary table, but only the user who inserted that data can perform DML statements on the data.
9. What is true about chaining?
- A. Chaining happens when updates are performed on blocks without enough free space.
 - B. You can control chaining through proper setting of the PCTFREE parameter.
 - C. You can control chaining through proper setting of the PCTUSED parameter.
 - D. Chaining happens when a row is inserted that is bigger than the database block size.
 - E. You can fix chaining by issuing the ALTER TABLE *table_name* MOVE command.
10. What storage parameter cannot be changed using the ALTER TABLE command?
- A. NEXT
 - B. INITIAL
 - C. PCTFREE
 - D. MAXEXTENTS
 - E. MAXTRANS
11. What is the high-water mark?
- A. The high-water mark denotes the maximum number of blocks ever addressed by the table.
 - B. The high-water mark is the end-of-file position in a datafile.
 - C. The high-water mark is the point in a block where the last row is located.
 - D. The high-water mark denotes the last block in an extent.
 - E. The high-water mark determines the last block that Oracle needs to read during an index or ROWID lookup.
12. What is one of the main benefits of the DBMS_SPACE.UNUSED_SPACE procedures?
- A. It will coalesce all free space on a block.
 - B. It will coalesce all free space in a table.
 - C. It can be used to find the high-water mark without your having to run the ANALYZE TABLE command.

- D. It can be used to deallocate space below the high-water mark.
 - E. It can be used to find blocks that are on the free list.
13. What is one difference between the TRUNCATE TABLE command and the DROP TABLE command?
- A. The TRUNCATE TABLE command deletes all rows from a table and deallocates space, whereas the DROP TABLE command, which also deletes all rows from a table and deallocates space, removes the table definition as well.
 - B. The DROP TABLE command repositions the high-water mark to the beginning of the table, whereas the TRUNCATE TABLE command simply deletes all rows from the table.
 - C. The DROP TABLE command can be rolled back; the TRUNCATE COMMAND cannot.
 - D. The TRUNCATE TABLE command can be rolled back; the DROP TABLE command cannot.
 - E. The TRUNCATE TABLE command can be used to rebuild data blocks.
14. Which data dictionary view contains a row for every extent in the database?
- A. DBA_EXTENTS
 - B. DBA_TABLES
 - C. DBA_SEGMENTS
 - D. ALL_DBA_EXTENTS
 - E. DBA_EXTENT_INFO
15. Which function of the DBMS_ROWID package would you use to get the relative file number of a ROWID?
- A. DBMS_ROWID.ROWID_RELATIVE
 - B. DBMS_ROWID.RELATIVE_FNO
 - C. DBMS_ROWID.ROWID_RELATIVE_FNO
 - D. DBMS_ROWID.RELATIVE_FILE_NUMBER
 - E. DBMS_ROWID.ABSOLUTE_FILE_NUMBER

Scenarios

1. You have just been hired as a consultant by DB EXPERTS AND SO MUCH MORE ltd. The company has a client who is concerned about the rapid rate at which his database is growing. The two busiest tables in the database are growing at rates well beyond expectation. The volume of inserts has been as expected but the client appears to have miscalculated the storage requirements. He is concerned that he may need to purchase additional hardware.

The two tables in question are the CUSTOMER table and the SUPPLIER table. Here is a description of the tables:

```
SQL> DESC CUSTOMER
Name                               Null?    Type
-----
CUSTID                             NOT NULL NUMBER
NAME                                CHAR(50)
ADDRESS                             CHAR(100)
ADDRESS2                             CHAR(100)
PHONE                                CHAR(13)
COMMENTS                             CHAR(2000)
```

```
SQL> DESC SUPPLIERS
Name                               Null?    Type
-----
SUPPLIERID                         NOT NULL NUMBER
NAME                                CHAR(50)
ADDRESS                             CHAR(100)
ADDRESS2                             CHAR(100)
PHONE                                CHAR(13)
FAX                                  CHAR(13)
CONTACTNAME                         CHAR(40)
TERMS                                CHAR(10)
EMAIL                                CHAR(50)
COMMENTS                             CHAR(2000)
```

The customer has also noted that the table is always having to be rebuilt because of problems with migration. Here is some of the information about both tables from the DBA_TABLES view:

```
SQL> SELECT TABLE_NAME, PCT_FREE, PCT_USED, AVG_ROW_LEN
2 FROM USER_TABLES
3 WHERE TABLE_NAME IN ('CUSTOMER', 'SUPPLIERS')
4 /
```

TABLE_NAME	PCT_FREE	PCT_USED	AVG_ROW_LEN
CUSTOMER	10	50	2276
SUPPLIERS	10	50	2393

- A.** Based on the information given in the two tables, what recommendations would you make about the space issues?
 - B.** How would you go about implementing the recommendations?
 - C.** How would you go about fixing the customer's problems with migration so that migration does not happen as much?
- 2.** You have been hired as a DBA by WE SELL PARTS Ltd. The company has experienced tremendous growth and realizes the need for a full time DBA. The company's system administrator had been performing the task but quickly realized that she had bitten off more than she could chew. The database being used is the one that was created when Oracle was installed. No additional tablespaces have been created and all the tables have been created "on the fly" as they were needed, without any of the default storage parameters being changed.

Upon further investigation by you, you quickly realize that all objects are being stored in the SYSTEM tablespace. Furthermore, the tables were created using the Oracle default values for the storage and block utilization parameters. No statistics exist for the database, and performance is very poor. IO seems to be the biggest problem. The database has several very large tables and indexes.

- A.** What would you recommend about the configuration of the database in terms of the tablespaces?
- B.** How would you approach the problems with performance and how would you implement your approach?

Lab Exercises

Lab 11-1 Creating and Managing Tables

1. Connect to your instance using SQLPLUS line mode (sqlplus) as the user student using the password oracle.
2. Create the EMPLOYEE table using the following command:

```
CREATE TABLE EMPLOYEE (  
  EMPNO NUMBER CONSTRAINT EMP_EMPNO_PK PRIMARY KEY  
  USING INDEX TABLESPACE INDX,  
  ENAME CHAR(50) NOT NULL,  
  ADDRESS1 CHAR(50) NOT NULL,  
  ADDRESS2 CHAR(50),  
  CITY CHAR(30),  
  PHONE CHAR(12),  
  EMAIL CHAR(50),  
  SAL NUMBER)  
TABLESPACE CERTDB  
STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 100)  
PCTFREE 5;
```

3. Insert 200 rows into the table using the following command:

```
BEGIN  
  FOR i IN 1..200 LOOP  
    INSERT INTO EMPLOYEE VALUES (  
      i,  
      'SAME NAME',  
      '123 MAIN ST.',  
      'ADDRESS LINE 2',  
      'NEW YORK',  
      '555-123-555',  
      'SNAME@EMAIL.COM',  
      1000.00);  
  END LOOP;  
END;  
/
```

4. Compute statistics for the EMPLOYEE table.
5. Query the data dictionary and find the number of blocks, the number of rows, the average row size, and the number of chained or migrated rows for the EMPLOYEE table. Fill in the information for the following table.

<i>BLOCKS</i>	<i>NUMBER OF ROWS</i>	<i>AVERAGE ROW LENGTH</i>	<i>CHAIN COUNT</i>
---------------	-----------------------	---------------------------	--------------------

6. Modify all the columns that have a datatype of CHAR and change them to VARCHAR2, with the same length.
7. Update all the columns modified in the previous question using the TRIM function.
8. Compute statistics on the EMPLOYEE table.
9. Rerun your query from question 5 and compare the value of the average row length taken in question five. Notice how changing from a CHAR to a VARCHAR2 for the column datatypes saves an average of 173 bytes per row.

Lab 11-2 Setting Appropriate Block Utilization Parameters

1. Connect to your instance using SQLPLUS line mode (sqlplus) as the user student using the password oracle.
2. The EMPLOYEE table must be created and populated with data from the previous exercise. Verify its existence.
3. Compute statistics on the EMPLOYEE table.
4. Query the data dictionary and find the number of blocks, the number of rows, the average row size, and the number of chained or migrated rows for the EMPLOYEE table. Fill in the information for the following table.

<i>BLOCKS</i>	<i>NUMBER OF ROWS</i>	<i>AVERAGE ROW LENGTH</i>	<i>CHAIN COUNT</i>
---------------	-----------------------	---------------------------	--------------------

5. Add a column called COMMENTS to the EMPLOYEE table that is a VARCHARS(2000).
6. Update the COMMENTS column in the EMPLOYEE table using the following command:

```
UPDATE EMPLOYEE SET
COMMENTS = RPAD('C',1000,'C');
```

7. Compute statistics on the EMPLOYEE table.
8. Rerun the query that you ran in question 4 and fill in the new values for the selected values.

<i>BLOCKS</i>	<i>NUMBER OF ROWS</i>	<i>AVERAGE ROW LENGTH</i>	<i>CHAIN COUNT</i>
---------------	-----------------------	---------------------------	--------------------

- Check the setting of PCTFREE for the EMPLOYEE table and change it to a more appropriate value of 30.
- Changing the value of PCTFREE will not have an immediate effect. Rebuild the table and check the number of blocks along with the number of migrated or chained rows. Remember that you will need to rebuild all indexes on the EMPLOYEE table as well as compute statistics to see the new table statistics.
- Why are there no more chained or migrated rows and why are there more blocks?

Lab 11-3 Dropping Columns and Deallocating Unused Space

- Connect to your instance using SQLPLUS line mode (sqlplus) as the user student using the password oracle.
- The EMPLOYEE table must be created and populated with data from the previous exercise. Verify its existence.
- Mark the COMMENTS column of the EMPLOYEE table as UNUSED.
- Describe the EMPLOYEE table and note that the COMMENTS column no longer is there — or is it?
- Compute statistics on the EMPLOYEE table, query the data dictionary, and find the number of blocks, the number of rows, the average row size, and the number of chained or migrated rows for the EMPLOYEE table. Fill in the information for the following table.

<i>BLOCKS</i>	<i>NUMBER OF ROWS</i>	<i>AVERAGE ROW LENGTH</i>	<i>CHAIN COUNT</i>
---------------	-----------------------	---------------------------	--------------------

- Why has the average row length not changed after setting a column with an average of 1000 characters in it UNUSED?
- Drop the COMMENTS column.
- Compute statistics on the EMPLOYEE table, query the data dictionary, and find the number of blocks, the number of rows, the average row size, and the number of chained or migrated rows for the EMPLOYEE table. Fill in the information for the following table. Notice the average row size.

<i>BLOCKS</i>	<i>NUMBER OF ROWS</i>	<i>AVERAGE ROW LENGTH</i>	<i>CHAIN COUNT</i>
---------------	-----------------------	---------------------------	--------------------

9. Create a table called DEPARTMENTS using the following command and storage parameters:

```
CREATE TABLE DEPARTMENTS (
  ID      NUMBER(4),
  NAME    VARCHAR2(40))
TABLESPACE CERTDB
STORAGE (INITIAL 10K NEXT 10K MINEXTENTS 5
         PCTINCREASE 100);
```

10. Insert rows in the DEPARTMENTS table using the following command:

```
BEGIN
  FOR i IN 1..500 LOOP
    INSERT INTO DEPARTMENTS VALUES (
      i,
      'DEPARTMENT '||i);
  END LOOP;
END;
/
```

11. Query the data dictionary to find the number of extents for the DEPARTMENTS table as well as the number of blocks used by the table. Remember that you will need to compute statistics first.
12. Assuming that all the rows that the database will ever need are in the DEPARTMENTS table, you may want to deallocate the extra space used by the table. Deallocate the extra extents used by the DEPARTMENTS table and query the data dictionary to see how many extents exist after the command runs.

Lab 11-4 Creating Temporary Tables

1. Connect to your instance using SQLPLUS line mode (sqlplus) as the user student using the password oracle.
2. Create a temporary table called EMPLOYEE_TEMP with the same columns as the EMPLOYEE table.
3. Insert records from the EMPLOYEE table where the EMPNO is less than 51. Use the following command:

```
INSERT INTO EMPLOYEE_TEMP
SELECT *
FROM   EMPLOYEE
WHERE  EMPNO < 51;
```

4. Alter the EMPLOYEE_TEMP table and multiply the SAL column by 150 percent.

5. Select the EMPNO and SAL columns from the EMPLOYEE_TEMP table and note the new values for the SAL column.
6. Issue a COMMIT and then select all rows from the EMPLOYEE_TEMP table.
7. Why are there no rows?

Answers to Chapter Questions

Chapter Pre-Test

1. Any user who has the CREATE TABLE system privilege and a quota on the tablespace where the table is being created can create tables.
2. An Oracle cluster is a special type of storage area for storing tables. Tables in a cluster have their data stored physically together on the same block based upon the value of a Cluster Key. Oracle uses two types of clusters: Index clusters and Hash clusters.
3. A CHAR datatype is a fixed-width datatype where varchar2 is a variable character width. The value “Todd Ross” will occupy 30 spaces in a char(30) column and roughly 9 in a varchar2(30) column. Varchar2 columns use space more efficiently.
4. LOBs in Oracle can be up to 4GB in size. LOBs replaced LONGs, which could store only 2GB.
5. A nested table is a table within a column. A column such as address can be declared with a nested table called ADDRESS_OBJ. The ADDRESS_OBJ will be an actual table that can store multiple values. Therefore, if you wanted to store multiple addresses for customers, instead of storing multiple address lines for the different address, you could simply use a nested table that could have multiple rows for each different address. Using this method does not put a limit on the number of addresses. If the table was initially created with one column for the mailing address and another for the service address, the table is limited to storing only two addresses. If a third address, such as a business address, was required to be stored, the table definition would need to be changed. If a nested table was used, no changes are required.
6. A free list is a list of blocks that are available for insert. Each table will have at least one free list. When blocks fill, they are removed from the free list because they are no longer available for inserts. When space is free in data blocks, the blocks are placed back on the free list and made available for inserts again. The PCTFREE and PCTUSED block storage parameters control when blocks are removed and placed back on the free list.
7. The NOLOGGING parameter improves the performance of tables that are created as subqueries, the CREATE TABLE AS SELECT statement. When you use NOLOGGING, Oracle does not record the rows inserted from the subquery in the redo log files. If you don't use NOLOGGING, Oracle places each row that is inserted into the table in the redo log files also. This parameter helps table

creations perform five to six times faster depending on the number of rows being inserted. The NOLOGGING parameter applies only to the actual table creation; after the table has been created, all future DML statements will be logged regardless of the value of the NOLOGGING parameter.

8. The data for temporary tables is stored in the PGA of the user who is inserting into the table. The PGA is a memory allocation, and if the amount of data the user is inserting exceeds the available memory in the user's PGA, the data is stored in the user's temporary tablespace.
9. The only user who can see data inside a temporary table is the user who placed it there. If there is a temporary table called LOCATIONS_TEMP and user BOB inserts a row, only BOB can see this row. If user DOUG also inserts a row in the LOCATIONS_TEMP table, only user DOUG can see that row.
10. The ALTER TABLE MOVE command will create a copy of the table and then drop the original. This is a way of rebuilding a table to fix problems with fragmentation or chaining. The table can be built in a new tablespace or the same tablespace. If you're rebuilding a table in the same tablespace, you'll need two times the amount of space because the new table is created before the old one is dropped. For example, if a table is 15MB in size and is being rebuilt in the same tablespace, that tablespace must be at least 30MB in size.
11. The high-water mark is the maximum number of blocks that have been on the free list. If 2,000 blocks are being used for a table, the high-water mark will be somewhere around 2,000 blocks. If 1,000 blocks are being used for a table, but at some point 2,000 blocks were being used, the high-water mark will be somewhere around 2,000 blocks. This occurs because the high-water mark can move only up, never down. The high-water mark tells Oracle which blocks need to be read during a full table scan.
12. The DBA_TABLES data dictionary view reports the number of chained rows per table. There is a column called CHAIN_CNT. This column gets populated only when the table is ANALYZED. So, if a table that has not been ANALYZED contains 200 chained rows, the CHAIN_CNT column of DBA_TABLES for that table will be NULL until the table is ANALYZED.

Assessment Questions

1. **B.** The maximum size for varchar2 datatypes is 4,000 bytes. It is the char datatype that is limited to 2,000 bytes. For more information on all scalar datatypes, refer to the section "Oracle Datatypes."
2. **C and E.** Two of the big advantages of LOBs over LONGs are having the capacity for multiple LOBs per table that can be up to 4GB in size. LONG datatypes are limited to 2GB. More information about LONGs and LOBs can be found in the section on "Large object datatypes."
3. **D.** The file number is not part of the new ROWID format. It is the relative file number. It is relative to the object number, whereas the file number is unique in the database. You can extract the file number from the ROWID by using the

DBMS_ROWID package, but it is not part of the extended ROWID. For more information, refer to the section entitled “The ROWID and UROWID datatypes.”

4. **D.** The PCTFREE parameter controls the amount of space that is reserved on every block for rows on that block to grow. When the PCTFREE threshold is exceeded for a block, the block is removed from the free list. If a block is removed from the free list, no more inserts are permitted—only updates. Refer to the section on “Creating Tables” for more information.
5. **B.** The default value for PCTFREE is 10 percent. The default value is used if a value is not specified when the table is created. None of the block utilization parameters can be specified as tablespace defaults. Refer to the section on “Creating Tables” for more information.
6. **D.** PCTINCREASE controls extent sizes only after the second extent. The PCTINCREASE value is multiplied with the previous extent size to determine the size of the next extent. This information is stored in the NEXT_EXTENT column of the DBA_TABLES data dictionary view. You can find more information on PCTINCREASE in the section “Creating Tables.”
7. **E.** NOLOGGING specifies that the creation of the table and certain types of data loads, such as a direct load insert using SQL loader or an INSERT APPEND command, will not be logged in the redo log files.
8. **D.** With temporary tables, the definition is permanent. That means that you can use only the DROP TABLE command to remove it. The data is temporary; that is, when a user inserts data, it stays in the temporary table for the duration of the user’s session or the duration of the user’s transaction. This is determined when the temporary table is created. You can find more information in the section “Temporary tables.”
9. **D.** Chaining happens when a row is larger than a block. If the database block size is 2KB, determined by the DB_BLOCK_SIZE parameter, and a row is 3KB, Oracle chains multiple blocks together to store the entire row. The only way to fix chaining is by splitting the table up into multiple tables or increasing the database block size. You can increase the database block size only by rebuilding the entire database. Refer to the section “Row migration and chaining.”
10. **B.** You can specify the value of INITIAL only at table creation time. Refer to the section “Altering storage parameters” for more information.
11. **A.** The high-water mark is the last block to ever be addressed by the table with data. It represents the position in the table that Oracle needs to read to during a full table scan. The section “The high-water mark” contains detailed information.
12. **C.** This package’s main benefit is that it can be used to report the high-water mark without the need to run the ANALYZE TABLE command. Refer to the section “Handling unused space” for more information.

13. **A.** The difference between the TRUNCATE TABLE command and the DROP TABLE command is that the DROP TABLE command removes the table definition from the data dictionary. Both commands remove all rows and deallocate space. Refer to the “Modifying Tables” section for more information.
14. **A.** The DBA_EXTENTS view contains a row for every extent in the database. You can find more information in the “Retrieving Table Information” section.
15. **C.** You can use the DBMS_ROWID.ROWID_RELATIVE_FNO function to obtain the relative file number for a ROWID. For more information, refer to the section “DBMS_ROWID Package.”

Scenarios

1. **A.** Based on the information provided, you should recommend that the columns defined as CHAR be converted to VARCHAR2. VARCHAR2 columns require less storage than do CHARs. CHAR columns pad data with nulls, which can result in wasted space.
1. **B.** An ALTER TABLE command is required to change the column datatypes. When changing the datatypes from CHAR to VARCHAR2, the columns will need to be updated to remove the padded nulls.
1. **C.** The problems with migration stem from the low value set for PCTFREE in both tables. Not enough space is being reserved on the blocks for updates, which is resulting in rows migrating. Use the ALTER TABLE command to change the value of PCTFREE and then use the ALTER TABLE MOVE command to rebuild it. Doing so will eliminate migrated rows and help reduce the chance of rows migrating from future updates. Increasing the value of PCTFREE will result in more blocks being required for the table; however, changing the column datatypes to VARCHAR2 will dramatically reduce the space used by both tables.
2. **A.** You should start by creating additional tablespaces for storing DATA, INDEXES, ROLLBACK SEGMENTS, and TEMPORARY SEGMENTS. After the tablespaces have been created, you should relocate the tables, Indexes, Rollback Segments, and Temporary Segments to the appropriate tablespaces. Splitting up table data from index data should help with some of the IO problems.
2. **B.** To determine the true magnitude of the performance issues, statistics will need to be generated. These statistics will provide valuable information about how effective the storage and block utilization parameters are. Because the database default values were used when the tables were created initially, consider adjusting the parameters based on the table’s size and activity. Large tables will require extent sizes bigger than the 5 block database default value. Also, because the default value for PCTINCREASE is 50 percent, all the large tables will be fragmented. Consider some larger standard extent sizes for these tables. The default value of 10 for PCTFREE may not be sufficient for the volatile tables. Check the statistics for tables with migration and consider adjusting the PCTFREE value and rebuilding these tables.

Lab Solutions

Lab 11-1

4.

```
ANALYZE TABLE EMPLOYEE COMPUTE STATISTICS;
```

5.

```
SQL> SELECT TABLE_NAME,BLOCKS,NUM_ROWS,AVG_ROW_LEN,CHAIN_CNT  
2 FROM USER_TABLES  
3 WHERE TABLE_NAME='EMPLOYEE';
```

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	CHAIN_CNT
EMPLOYEE	10	200	257	0

6.

```
ALTER TABLE EMPLOYEE MODIFY  
ENAME VARCHAR2(50);
```

```
ALTER TABLE EMPLOYEE MODIFY  
ADDRESS1 VARCHAR2(50);
```

```
ALTER TABLE EMPLOYEE MODIFY  
ADDRESS2 VARCHAR2(50);
```

```
ALTER TABLE EMPLOYEE MODIFY  
CITY VARCHAR2(30);
```

```
ALTER TABLE EMPLOYEE MODIFY  
PHONE VARCHAR2(12);
```

```
ALTER TABLE EMPLOYEE MODIFY  
EMAIL VARCHAR2(50);
```

7.

```
UPDATE EMPLOYEE SET  
ENAME = TRIM(ENAME),  
ADDRESS1 = TRIM(ADDRESS1),  
ADDRESS2 = TRIM(ADDRESS2),  
CITY = TRIM(CITY),  
PHONE = TRIM(PHONE),  
EMAIL = TRIM(EMAIL);
```

8.

```
ANALYZE TABLE EMPLOYEE COMPUTE STATISTICS;——
```

9.

```
SQL> SELECT TABLE_NAME,BLOCKS,NUM_ROWS,AVG_ROW_LEN,CHAIN_CNT
2 FROM USER_TABLES
3 WHERE TABLE_NAME='EMPLOYEE';
```

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	CHAIN_CNT
EMPLOYEE	7	200	84	0

Lab 11-2

2.

```
SQL> DESC EMPLOYEE
Name                               Null?   Type
-----
EMPNO                               NOT NULL NUMBER
ENAME                               NOT NULL VARCHAR2(50)
ADDRESS1                            NOT NULL VARCHAR2(50)
ADDRESS2                             VARCHAR2(50)
CITY                                 VARCHAR2(30)
PHONE                                 VARCHAR2(12)
EMAIL                                VARCHAR2(50)
SAL                                   NUMBER
```

3.

```
ANALYZE TABLE EMPLOYEE COMPUTE STATISTICS;
```

Table Analyzed.

4.

```
SQL> SELECT TABLE_NAME,BLOCKS,NUM_ROWS,AVG_ROW_LEN,CHAIN_CNT
2 FROM USER_TABLES
3 WHERE TABLE_NAME='EMPLOYEE';
```

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	CHAIN_CNT
EMPLOYEE	7	200	84	0

5.

```
SQL> ALTER TABLE EMPLOYEE ADD COMMENTS VARCHAR2(2000);
```

Table altered.

6.

```
SQL> UPDATE EMPLOYEE SET  
2   COMMENTS = RPAD('C',1000,'C');
```

200 rows updated.

7.

```
ANALYZE TABLE EMPLOYEE COMPUTE STATISTICS;
```

Table analyzed.

8.

```
SQL> SELECT TABLE_NAME,BLOCKS,NUM_ROWS,AVG_ROW_LEN,CHAIN_CNT  
2   FROM   USER_TABLES  
3   WHERE  TABLE_NAME='EMPLOYEE';
```

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	CHAIN_CNT
EMPLOYEE	29	200	1093	173

9.

```
SQL> SELECT PCT_FREE  
2   FROM   USER_TABLES  
3   WHERE  TABLE_NAME = 'EMPLOYEE';
```

```
   PCT_FREE  
-----  
          5
```

```
SQL> ALTER TABLE EMPLOYEE PCTFREE 30;
```

Table altered.

10.

```
SQL> ALTER TABLE EMPLOYEE PCTFREE 30;
```

Table altered.

```
SQL> ALTER TABLE EMPLOYEE MOVE;
```

Table altered.

```
SQL> ALTER INDEX EMP_EMPNO_PK REBUILD;
```

Index altered.

```
SQL> ANALYZE TABLE EMPLOYEE COMPUTE STATISTICS;
```

Table analyzed.

```
SQL> SELECT TABLE_NAME,BLOCKS,NUM_ROWS,AVG_ROW_LEN,CHAIN_CNT
2 FROM USER_TABLES
3 WHERE TABLE_NAME='EMPLOYEE';
```

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	CHAIN_CNT
EMPLOYEE	40	200	1087	0

- 11.** The MOVE command rebuilds the table, which will fix all migrated rows. The rows were not chained rows. Changing the value of PCTFREE means that Oracle will reserve more space on each block for updates. This means that there are fewer rows per block, which means that there must be more blocks. As you can see, a trade-off takes place between eliminating migration and reducing the number of blocks used by a table.

Lab 11-3

2.

```
SQL> DESC EMPLOYEE
Name                               Null?    Type
-----
EMPNO                               NOT NULL NUMBER
ENAME                               NOT NULL VARCHAR2(50)
ADDRESS1                            NOT NULL VARCHAR2(50)
ADDRESS2                             VARCHAR2(50)
CITY                                 VARCHAR2(30)
PHONE
VARCHAR2(12)
EMAIL                                VARCHAR2(50)
SAL                                  NUMBER
COMMENTS                             VARCHAR2(2000)
```

3.

```
SQL> ALTER TABLE EMPLOYEE SET UNUSED COLUMN COMMENTS;
```

Table altered.

4.

```
SQL> DESC EMPLOYEE
Name                               Null?    Type
-----
EMPNO                               NOT NULL NUMBER
ENAME                               NOT NULL VARCHAR2(50)
ADDRESS1                            NOT NULL VARCHAR2(50)
ADDRESS2                             VARCHAR2(50)
CITY                                 VARCHAR2(30)
PHONE                                VARCHAR2(12)
EMAIL                                VARCHAR2(50)
SAL                                  NUMBER
```

5.

```
ANALYZE TABLE EMPLOYEE COMPUTE STATISTICS;
```

```
SQL> SELECT TABLE_NAME,BLOCKS,NUM_ROWS,AVG_ROW_LEN,CHAIN_CNT
2 FROM USER_TABLES
3 WHERE TABLE_NAME='EMPLOYEE';
```

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	CHAIN_CNT
EMPLOYEE	40	200	1087	0

6. No space is released when the SET UNUSED command is issued. The SET UNUSED command just modifies the data dictionary. The column must be dropped to release the space used by the column COMMENTS. The SET UNUSED command is used to quickly drop a column and to avoid locking the table, which the DROP COLUMN does. If the table has a large number of rows, the DROP COLUMN command may lock the table for a long time.

7.

```
SQL> ALTER TABLE EMPLOYEE DROP UNUSED COLUMNS;
```

```
Table altered.
```

8.

```
SQL> ANALYZE TABLE EMPLOYEE COMPUTE STATISTICS;
```

```
Table analyzed.
```

```
SQL> SELECT TABLE_NAME,BLOCKS,NUM_ROWS,AVG_ROW_LEN,CHAIN_CNT
2 FROM USER_TABLES
3 WHERE TABLE_NAME='EMPLOYEE';
```

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	CHAIN_CNT
EMPLOYEE	40	200	84	0

11.

```
SQL> ANALYZE TABLE DEPARTMENTS COMPUTE STATISTICS;
```

```
Table analyzed.
```

```
SQL> SELECT BLOCKS
2 FROM USER_TABLES
3 WHERE TABLE_NAME='DEPARTMENTS';
```

BLOCKS
3

```
SQL> SELECT COUNT(*)
       2 FROM   USER_EXTENTS
       3 WHERE  SEGMENT_NAME = 'DEPARTMENTS';

COUNT(*)
-----
          5
```

12.

```
ALTER TABLE DEPARTMENTS DEALLOCATE UNUSED KEEP 0;
```

```
SQL> SELECT COUNT(*)
       2 FROM   USER_EXTENTS
       3 WHERE  SEGMENT_NAME = 'DEPARTMENTS';

COUNT(*)
-----
          2
```

Lab 11-4**2.**

```
SQL> 1
      2 CREATE GLOBAL TEMPORARY TABLE EMPLOYEE_TEMP
      3   ON COMMIT DELETE ROWS
      4   AS
      5     SELECT *
      6     FROM EMPLOYEE
      7     WHERE 1=0
```

3.

```
INSERT INTO EMPLOYEE_TEMP
SELECT *
FROM   EMPLOYEE
WHERE  EMPNO < 51;
```

4.

```
SQL> UPDATE EMPLOYEE_TEMP
      2 SET SAL = SAL * 1.5;
```

5.

```
SELECT EMPNO, SAL
FROM   EMPLOYEE_TEMP
```

6.

```
COMMIT;  
  
SELECT *  
FROM   EMPLOYEE_TEMP;
```

7. No rows are in the temporary table because a COMMIT was issued. The COMMIT deletes all rows in temporary tables when created using the ON COMMIT DELETE ROWS option. Without this option, the data is deleted only when the user disconnects his or her session.

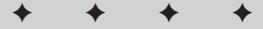
Managing Indexes

EXAM OBJECTIVES

- ◆ Managing Indexes
 - List the different types of indexes and their uses
 - Create B-tree and bitmap indexes
 - Reorganize indexes
 - Drop indexes
 - Get index information from the data dictionary

12

CHAPTER



CHAPTER PRE-TEST

1. What is the main benefit of an index?
2. What is a concatenated index?
3. What is a Reverse Key index?
4. What is stored in an index?
5. What columns should be indexed?
6. If an indexed column has a null, is that stored in the index?
7. What are the three methods for reorganizing an index?
8. What data dictionary view contains one row for every index in the database?
9. What data dictionary view can be used to list the columns that are indexed?

This chapter focuses on the DBA's job with regard to indexing. DBAs must be well versed in the different types of indexes and when these indexes should and should not be used. Application developers have a tendency to want to over index. The first part of this chapter covers the different classifications of indexes as well as the costs and benefit of each type of index. Indexes are considered segments. Each segment in an Oracle database requires storage and depending upon the number of columns and the number of indexes being created, the storage requirements for an index can exceed that of the table the indexes are being created against. The second part of the chapter discusses creating indexes and the special storage requirements that need to be considered when creating indexes.

Another important and probably the most common type of DBA duty is maintaining indexes. Indexes tend to get disorganized much quicker than tables and most need to be reorganized on a regular basis. Extracting this information from the data dictionary and performing the index reorganization is the last section covered in this chapter.

Indexes can be used to improve the performance of queries and updates. The columns that should be indexed are those that appear in the WHERE clause, the ORDER BY clause, the GROUP BY clause, and columns referenced in a join. You must be careful about selecting the columns to index because indexes slow down INSERT and DELETE operations. On INSERTs and DELETEs, Oracle not only modifies the table but also the indexes. When deciding to index, pay careful attention to the type of index to use. Oracle has several different types of indexes that can be used in certain situations. A column that has only a couple of distinct values like GENDER would not be a good candidate for a regular index. It is, however, a good candidate for a bitmap index. Understanding the benefits of each type of index is a must for all DBAs.

Indexes do require storage because they are stored on blocks and in extents like all other objects in the database. When creating indexes, DBAs must ensure that they do not conflict with other objects and that they are sized optimally. Some of the block utilization parameters, like PCTFREE and PCTUSED, have different uses for indexes than they do for tables. Also, since indexes are stored in a sorted order and must remain that way, they tend to need more attention than do tables. DBAs must monitor indexes and reorganize them when needed.

Index Basics

An index is a tree structure that stores the indexed value and the ROWID from the row in the table that the index is based upon. The values are stored in a sorted order, which provides quick access to indexed values. It is similar to the index or glossary of a book. The index or glossary stores keywords with a reference to a page number in the book. This means you do not need to scan every page of a book looking for a particular subject. That task has already been done. The cost of this type of index is that extra pages are required for the book. The costs of an index in

a database are twofold. First, there is the storage cost. All indexes are stored in the database. Second is the cost of maintenance. With the glossary of a book, somebody has had to reference keywords with page numbers. This would have been done once. Books are static so the glossary does not need to be maintained. However, with indexes in a database, they do need to be maintained. As rows are added and deleted from a table that has indexes, the index entries must also be either added or deleted.

The index of a book references a page number. Find the keyword you are looking for, which is pretty easy since this index is stored in alphabetic order, and then go to the page number referenced. For database indexes, the process is similar except instead of page numbers they store ROWIDs. The ROWID is a page number for Oracle. It contains the file number, the block number, and the row number where the row can be found in the table. This means that the table does not need to be scanned completely looking for specific rows. This improves queries and updates that reference the indexed columns. If there is an index on the LASTNAME column of the INSTRUCTORS table, a query written this way can use the index:

```
SELECT *
FROM   INSTRUCTORS
WHERE  LASTNAME = 'Williams';
```

Since the index is on LASTNAME and the WHERE clause is based upon that column, Oracle can use the index. However, the following query is not able to use the index:

```
SELECT LASTNAME, EMAIL
FROM   INSTRUCTORS
WHERE  EMAIL = 'williams@mycorp.com';
```

This is because the index is on LASTNAME and the WHERE clause is using EMAIL. The other elements of a SQL statement that can benefit from an index would be the ORDER BY, GROUP BY, and columns referenced in a table join.

Oracle has several different types of indexes. Indexes can be classified based upon their logical design or on their physical implementation. The logical classification groups indexes generically based upon design requirements. These are usually application-related while the physical classification comes from how the indexes are stored.

Logical elements of indexes

The following discusses the logical elements of indexes in Oracle.

Single column or concatenated

Single-column indexes are indexes on only one column of a table. The index stores the value being indexed and the corresponding ROWID.

Concatenated indexes — also known as compound indexes — are indexes on multiple columns of a table. The columns do not need to be adjacent in the table and can be specified in any order. You could create a compound index on the `FIRSTNAME` and `LASTNAME` columns from the `INSTRUCTORS` table or on the `EMAIL` and `LASTNAME` columns. The maximum number of columns is 32 or the combined size of all the columns cannot exceed one-third the size of the database block. While the order with which you specify the columns for the compound index is not important, it is very important how you reference them in your queries. The first column used in the concatenated index must be included as one of the predicates in the SQL statement in order to use the index. For example, if there is a concatenated index on the `LASTNAME` and `FIRSTNAME` columns and in that order, then the `LASTNAME` column must be included in the statement. If a query is written that only references the `FIRSTNAME`, then it cannot use the index. With a concatenated index on the `LASTNAME` and `FIRSTNAME` columns and in that order, here are some examples of queries that will benefit and not benefit from the index:

```
SELECT *
FROM   INSTRUCTORS
WHERE  LASTNAME = 'Williams'
AND    FIRSTNAME = 'Geoff';
```

This example uses the index because the `LASTNAME` column is included in the `WHERE` clause:

```
SELECT *
FROM   INSTRUCTORS
WHERE  LASTNAME = 'Williams';
```

The preceding query also uses the index, even though the `FIRSTNAME` column is not included. Because the `LASTNAME` column is included and it is the first column in the index, Oracle can use the index.

```
SELECT *
FROM   INSTRUCTORS
WHERE  FIRSTNAME = 'Geoff';
```

The preceding query does not use the index because only the `FIRSTNAME` column is included in the `WHERE` clause. Since the `FIRSTNAME` column is the second column in the index, Oracle cannot use the index for this query. With the absence of any other indexes, this query will be resolved by a full tablescan.

Unique and nonunique indexes

Unique indexes guarantee that no duplicate values are stored in the index. Unique indexes are common on columns that uniquely identify rows in a table. For example, the `INSTRUCTORID` column of the `INSTRUCTORS` table would be a good candidate for a unique index. You would not want two rows in the table that have the same `INSTRUCTORID` or else it would be difficult to identify one specific row. Many

of the unique indexes in a database are associated with a PRIMARY KEY or UNIQUE constraint although they do not have to be. Each entry in a unique index points to only one row in a table.

Nonunique indexes can store duplicates. Each row in the table has an entry in the index. If there were two instructors in the INSTRUCTORS table with a last name of Williams, there would be two rows in the table and two entries in the index—one for each row in the table. The keys are repeated in the index.

Function-based indexes

Function-based indexes are a welcomed addition to a DBA's toolkit. These indexes are created using a function, such as LOWER(LASTNAME), or an expression like (SAL * 12). The index precomputes the values and stores the computed product in the index. This helps DBAs overcome many previous indexing problems. Previously, if the left side of a predicate was dirty, an index on that column could not be used. For example, in previous versions of Oracle if the statement were written this way:

```
SELECT *
FROM   INSTRUCTORS
WHERE  UPPER(LASTNAME) = 'WILLIAMS';
```

an index on the LASTNAME column could not be used. The left side of the predicate, the = sign, is dirty. The same condition is true with the following query:

```
SELECT *
FROM   INSTRUCTORS
WHERE  SALARY * 12 > 60000;
```

This is a common query that tries to find all employees whose salary is greater than 60,000. Again, the left side of the predicate, the > sign is dirty, which means that an index on the SALARY column could not be used.

DBAs can now overcome this limitation by creating function-based indexes. You want to ensure when creating the function-based indexes that the function or the expression matches the one used in the query. In the first example from above if the following index were created:

```
CREATE INDEX INSTR_UPPER_LNAME ON INSTRUCTOR (UPPER(LASTNAME));
```

Oracle would perform this function on all employee last names and store the value in the index. If the same function is used in the statement, the function-based index can be used.



Caution

There are some other criteria that need to be true in order for function-based indexes to be used. The INIT.ORA parameter ENABLE_QUERY_REWRITE must be set to true. This parameter can be modified using an ALTER SESSION command.

Physical elements of indexes

The physical elements of an index define how the index is stored. All indexes in Oracle are stored in a B-tree index. Storage in the database is either partitioned or nonpartitioned.



The B-tree index structure is covered in the section called “Types of Indexes in Oracle.”

Partitioned indexes are generally used with partitioned tables. Partitioning allows tables and indexes to be split into several different segments spanning different tablespaces. The partitioning is done based upon a column value where range, hash, and composite (range and hash) partitioning are supported. By splitting up the table in this way, tables and indexes can be spread across several different tablespaces on several different disks, decreasing any IO bottlenecks, increasing manageability and improving query and certain types of DML performance. Index partitions are usually created on partitioned tables; however, they can be created on nonpartitioned tables as well. A nonpartitioned index is just a regular index and can be on partitioned and nonpartitioned tables.



Partitioning is not included on the exam.

Types of Indexes

Objective

List the different types of indexes and their uses

Oracle now has several different indexing options. The following outlines the different types.

B-tree indexes

All indexes in Oracle are stored in a B-tree index. The term B-tree is generally associated with indexes that store the column value(s) and the ROWID for each key. The structure of the B-tree index is illustrated in Figure 12-1.

Structure of a B-tree index

The first level of a B-tree index is called the Root Level. The root level contains pointers to the next level, the Branch Level. The Branch Level has pointers to the next level, which can be another Branch Level or the Index Leaf Blocks as in the example in Figure 12-1. The Leaf Blocks, also known as Leaf Nodes, contain the actual index entries. Each index entry contains the column value as well as a ROWID. The ROWID is the pointer to the complete row in the table that the index is based upon. The Leaf Blocks are also doubly linked to one another to facilitate scanning the Leaf Blocks in ascending or descending order. That is what the arrows below the Index Blocks in Figure 12-1 indicate.

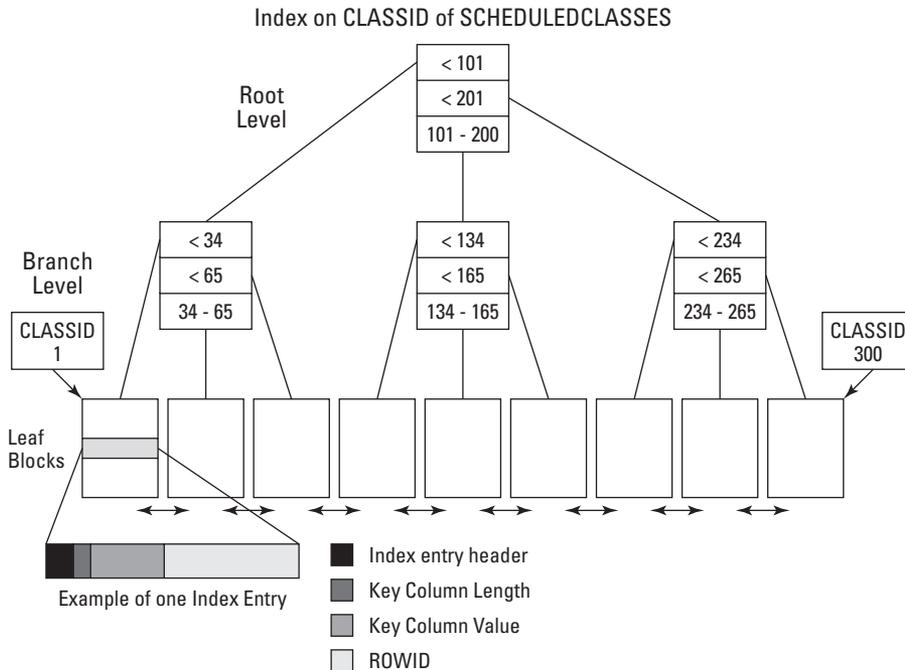


Figure 12-1: Structure of B-tree index

Format of the index Leaf Blocks

The index entries are made up of the following components:

- ♦ An entry header that stores the number of columns being indexed as well as locking information. The locking information is required for DML entries that affect the index.
- ♦ Key column length-value pairs that store the length of the column value in the index. This is required for variable character columns. There will be a key column length for each column being indexed. If it is a concatenated index, there will be a length identified for each column being indexed.
- ♦ Key column value stores the actual data. This is the value of the column as it appears in the table. This is updated if the column is updated in the table. If this is a nonunique index, there will be an entry for each value corresponding to the rows in the table.
- ♦ ROWID stores the ROWID from the table that the column references. The ROWID acts as a pointer to locate the specified row. Oracle uses the restricted ROWID format in indexes because all index entries belong to the same segment. For indexes on partitioned tables, Oracle uses the extended ROWID format since index entries can point to different segments of a partitioned table. Oracle needs the Object Identifier to identify the segment.



The restricted and extended ROWIDs are discussed in more detail in Chapter 11 of this book.

Nulls are not stored in the index, therefore, the number of index entries will only be equal to the number of rows in a table if the column being indexed has a NOT NULL constraint.



It is generally advisable to add a NOT NULL constraint to columns being indexed. When the NOT NULL constraint exists, Oracle can use the index for many more query operations, generally resulting in improved performance.

How indexes are used by SQL statements

If Oracle detects that there is an index on a column referenced in the SQL Statement, it will generally use it. (The algorithm applied is beyond the scope of this book.) When it uses the index, it first goes to the Root Level of the index being used and starts climbing the Branches until it finds the column value being indexed. For example, if the following query were written and using Figure 12-1 as a reference:

```
SELECT *
FROM   SCHEDULEDCLASSES
WHERE  CLASSID = 50;
```

Oracle would start at the Root Level and determine which Branch it needed to go to next. In Figure 12-1 you can see that the left most branch contains the key values for CLASSID where CLASSID is less than 101 that satisfy the query. The query is looking for CLASSID 50. At the left most branch it again performs a scan to see which block, at the next level, it needs to go to for CLASSID 50. The second most left Leaf Block contains entries where CLASSID is between 34 and 65, which satisfies the query. Since this is the last level, the Leaf Block level, it knows that CLASSID 50 is going to be on that Leaf Block. It then scans that block looking for CLASSID 50 and grabs the associated ROWID. It then uses the ROWID to extract the row from the SCHEDULED-CLASSES table where CLASSID = 50. It does not need to scan the SCHEDULED-CLASSES table because the ROWID contains the data file number, block number, and row number for that particular row.

Effect of DML on indexes

Oracle maintains all indexes for you. There is nothing that the user or developer needs to do when performing DML operations on a table with indexes. This maintenance adds overhead to all DML operations against a table with indexes. Here is how Oracle handles the different types of DML statements individually.

- ♦ Inserts result in Oracle inserting the entry on the specific Leaf Block. Since indexes are sorted, the entry must go to a specific block. Problems can occur if Oracle attempts to insert a new index value and there is no room left on the Leaf Block. When this happens, Oracle performs an Index Block Split. It takes

half the rows from the specified block and relocates them onto a new block and then adjusts the pointers at the Branch Level. Now the block that Oracle is attempting to insert onto is half full and the insert should complete. These index Leaf Block splits cause indexes to become disorganized and have a negative impact on performance. Many block splits require indexes to be reorganized.



Reorganizing indexes is covered in section “Modifying Indexes.”

- ♦ Deletes are treated as a logical delete in the index. The space used by the deleted row can only be reused after another entry with the same key value is inserted or updated. Index blocks, unlike table blocks, do not use PCTUSED. PCTUSED for a table indicates when a block should be put back on the free list and made available again for inserts. Index blocks are not placed back on the free list until the entire block is empty. Reorganizing indexes helps resolve problems of wasted space caused by deletes.
- ♦ Updates to the index value are treated as a delete and insert. The key value is deleted from its current location and inserted into its new location. Unlike tables, PCTFREE for indexes is only applicable at the time of index creation. In tables, PCTFREE controls when blocks are removed from the free list and no longer available for inserts. With indexes, PCTFREE controls how much room is reserved on the block at index creation time for future inserts.



Setting PCTFREE for indexes is covered in more detail in the section “Guidelines for Creating Indexes.”

Reverse Key indexes

Reverse Key indexes store the bytes of the key value or column value being indexed in reverse order. For example, a Reverse Key index on the CLASSID column of the SCHEDULEDCLASSES table would store CLASSID 109 as 901. Oracle performs the bytes flip when inserting rows in the index as well as when it queries the index.

Reverse Key indexes help overcome a common problem with regular B-tree indexes. If an index exists on a column that uses a sequence number or some other means of generating a sequential number, then all the index entries go on the same block—the index block that is on the farthest right. This causes contention for this block as all entries are attempting to insert there. Figure 12-2 helps illustrate the IO bottleneck that this situation creates.

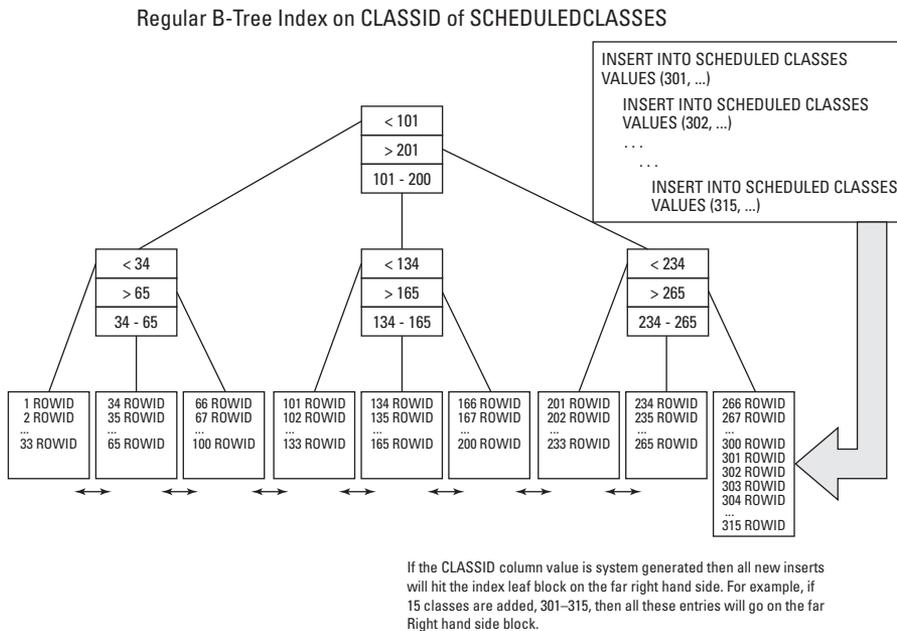


Figure 12-2: IO bottleneck in B-tree indexes

If a Reverse Key index were used, the IO bottleneck would disappear. Reverse Key indexes are more evenly distributed at the Leaf Block level. Instead of all entries being inserted on the one block that is farthest to the right of the index, inserts are distributed of all blocks evenly. This better distribution means that the indexes do not need to be reorganized as frequently as regular B-tree indexes. With a regular B-tree index the Leaf Block on the far right eventually becomes full. When this happens, the block is split into two blocks—half the information on one and half the information on the other. Once again, all inserts go on the far right-hand block. Eventually, this block also fills and splits. The left side block created as a result of the split will be half full (or for the pessimist, half empty). Since the index is being populated sequentially, those blocks will never be inserted into again. They remain half used until the index is reorganized. This evolution causes the index to consist of many more blocks than would be required if they were all packed evenly. The frequency of the rebuild depends upon the volume of transactions.

Figure 12-3 illustrates how a Reverse Key index evenly distributes IO for the index.

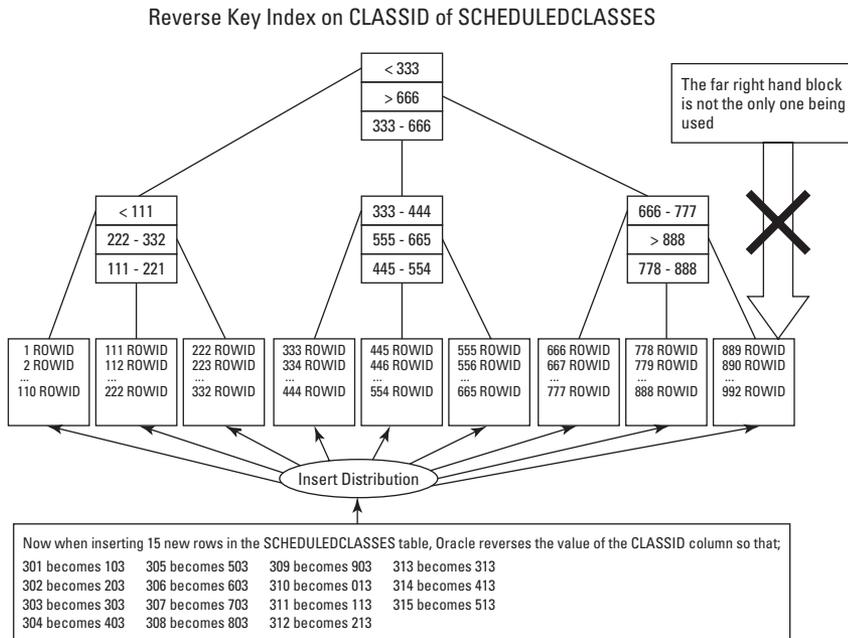


Figure 12-3: Reverse Key indexes

There are several important things to remember about reverse key indexes:

- ♦ Reverse Key indexes can only be used by queries that contain equality predicates. Any query that uses some sort of a range scan cannot use a Reverse Key index. The following query can use the index:

```
SELECT *
FROM SCHEDULEDCLASSES
WHERE CLASSID = 295;
```

When Oracle processes the query, it simply flips the 295 to 592 and retrieves the value and ROWID from the Reverse Key index. However, the following query cannot take advantage of the Reverse Key index:

```
SELECT *
FROM SCHEDULEDCLASSES
WHERE CLASSID BETWEEN 290 AND 295;
```

Since the values 290, 291, 292, 293, 294, and 295 become 092, 192, 292, 392, 492, 592 in the Reverse Key index, they will not be sitting side by side in the index.

- ♦ Reverse Keys start to become unbalanced when the values being inserted add a digit to the length of the index string, or when the values in the index increase by the power of 10. For example, if there were 9800 entries in the SCHEDULEDCLASSES table, then the index would be balanced over 4 digits, 1–9999. However, when the 10,000 class is added, which adds a digit to the index string, the values will no longer be balanced. All the new index entries will be inserted in the right-most Leaf Block again. However, after rebuilding the index, it should not need rebuilding again until the 100,000 class is added and after that the 1,000,000 class.

Function-based indexes

Function-based indexes are a welcomed relief for DBAs, designers, and developers. Prior to function-based indexes, certain types of queries were difficult to tune and often required adding unnecessary columns to a table to ensure that an index would be used. For example, if users are constantly looking up customer names it would make sense to add an index to the customer name column. However, queries must have clean predicates in the where clause in order to use index. This meant that users needed to know how the data was stored in the table to ensure the index was used. For example, if you were looking for a customer named Jacob Ross, how would the query look?

```
SELECT *
FROM   CUSTOMER
WHERE  NAME = 'Jacob Ross';
```

OR

```
SELECT *
FROM   CUSTOMER
WHERE  NAME = 'JACOB ROSS';
```

OR

```
SELECT *
FROM   CUSTOMER
WHERE  NAME = 'jacob ross';
```

You needed to know how the data was stored to resolve this query using an index because, if the query were written like this:

```
SELECT *
FROM   CUSTOMER
WHERE  UPPER(NAME) LIKE 'JACOB ROSS';
```

the index cannot be used. The predicate in the WHERE clause is dirty. You cannot use a function or expression on the left side of a predicate; in this case the LIKE is the predicate, and uses the index. Oracle stores the data in the database the same

way that it is entered. If it were entered as Jacob Ross, that is how Oracle stores it, even in the index. Therefore, the index on the NAME column of the CUSTOMER table stores the values exactly the same as they are stored in the table. This means that Oracle cannot guarantee that a lowercase j and an uppercase J will be side by side on the index. They are 26 characters apart on the ASCII chart so chances are good that they will not be side by side.

To fix this problem, a column could be added to the table. There would be a column called NAME for displaying the customer name, printing the customer name, and so forth, and another column called QNAME for query purposes. The QNAME column could store the customer name in all uppercase, lowercase or with the first character capitalized. Knowing how the data is stored ensured that users could always look up a customer using all uppercase or whatever case convention was chosen. The problem with this is one of maintenance. The column QNAME will need to be changed whenever the NAME column changes. This added complexity for developers and possible data corruption. This is an example of denormalizing a database. But it had to be done!

In Oracle8.1, function-based indexes came to the rescue. Function-based indexes can be created on functions or expressions. To resolve the problem of the customer name, you could create a function based index on UPPER(NAME) of the CUSTOMER table. A function-based index reduces the amount of work that Oracle needs to do when processing a SELECT or DELETE statement. The index stores the computed or materialized expression. The result of the function UPPER (if that function is used) will be stored in the index. Function-based indexes can also be created using expressions. If your employee table stores the salary as a yearly figure but you have queries that break it down to a monthly amount, it may be beneficial to create a function-based index.

```
SELECT *
FROM   EMPLOYEE
WHERE  SAL / 12 < 1000;
```

In this query, the left side of the predicate, in this case the < sign, is dirty. SAL / 12 is an expression and Oracle cannot use the index to resolve the query. However, if a function-based index were created using the expression (SAL / 12), then it could be used. Here are two examples of function-based indexes:

```
CREATE INDEX EMPLOYEE_SAL_MONTHLY ON EMPLOYEE (SAL / 12);

CREATE INDEX CUST_NAME_UPPER ON CUSTOMER (UPPER(NAME));
```

Function based indexes require special attention. There is an INIT.ORA PARAMETER that needs to be turned on in order for a query to use a function-based index. Set QUERY_REWRITE_ENABLED = TRUE for either the entire instance or it can be set for certain sessions. You may also find even after enabling this parameter that the function-based index is still not being used. The decision to use the index is made

by Oracle, and just because an index exists does not guarantee that it will be used. Generally speaking if it makes sense, the index will be used, unless Oracle calculates a better execution plan to use.

Bitmap indexes

Bitmap indexes can be used to overcome some deficiencies of regular B-tree indexes. The situations where bitmap indexes are more efficient than B-tree indexes are as follows:

- ♦ If a column has a low cardinality of values (that is, few unique values stored in the column or columns) and there are many rows in the table, bitmap indexes are more efficient than B-tree. For example, a sales table for an auto manufacturer will have millions of rows with only a few distinct values for the make of the car and the color of the car. If there were five million rows in the table, there may be ten different models of cars sold with ten different colors. If a query were written looking for all car models of LIBRA, the number of rows returned may be hundreds of thousands. If there were five hundred thousand model LIBRA cars sold, then the query would return five hundred thousand rows. In this situation, an index lookup would result in one million logical units of IO — five hundred thousand for the index and another five hundred thousand for the table. In this situation, it would be faster to perform a full table scan than use the index. If a bitmap index existed, it would be faster still.
- ♦ Queries that have multiple predicates joined by either an AND or OR or AND/OR are problematic for B-tree indexes. These are efficiently resolved with bitmap indexes. If a query were written to look for all RED LIBRAs sold, B-tree indexes would be inefficient whereas bitmap indexes are perfectly suited for this type of query.
- ♦ Bitmap indexes are not as efficient as B-tree indexes for most DML operations so they tend to be more efficient for static tables.

Bitmap indexes store separate bitmaps for every distinct column value in the index. In the example of the auto manufacturer, there are ten different models of car sold, which means that there would be ten different bitmaps stored in the index, one for each model. The bitmap index also stores the start and end ROWIDs for the rows in the table. Oracle can then dynamically build ROWIDs using the bitmap and the start and end ROWIDs. Figure 12-4 illustrates how bitmaps are stored. The example in Figure 12-4 assumes there are only 20 rows in the table and four distinct values. The distinct values are the models of cars: LIBRA, SPOCK, DRAGON, and FISTER. The start ROWID is the ROWID of the first row in the table. The end ROWID is the last ROWID in the table.

Bitmap on MODEL Column of the SALES Table

Key Value	Start ROWID	End ROWID	Bitmaps for Each Key Value																				
LIBRA	30.1.1	30.6.2	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	
SPOCK	30.1.1	30.6.2	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0
DRAGON	30.1.1	30.6.2	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0	1	0
FISTER	30.1.1	30.6.2	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0

```
SELECT*
FROM SALES
WHERE MODEL = 'FISTER';
```

Oracle takes the bitmap where MODEL = FISTER and dynamically builds the ROWIDs using theStart ROWID and End ROWID. A value of 1 in the bitmap indicates that the corresponding ROWID is that Key Value. 0 is false.

ROW NUMBER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
FISTER	30.1.1	30.1.20	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0

The 6th, 12th and 19th bits are on – set to 1. This means that those rows have the MODEL column equal to FISTER. Oracle then dynamically builds the ROWIDs.

- The ROWIDs where model equal FISTER are
- 30.1.6
 - 30.1.12
 - 30.1.19

Figure 12-4: Bitmap indexes

Bitmap indexes are efficient for resolving queries with multiple predicates like AND and OR, AND/OR, and NOT IN. Figure 12-5 illustrates how bitmap indexes resolve these types of queries. Again, the bitmap index is on the MODEL column of the SALES table.

If an AND predicate is used, bitmaps need to exist on both columns referenced in the WHERE clause. Oracle will perform a bit and operation on those bitmaps to once again derive a single bitmap. With that bitmap, Oracle can then dynamically construct the ROWIDs for the rows that match the statement.

Structure of a bitmap

Bitmaps are stored in B-tree indexes. A bitmap is stored for each distinct column value in the index. The Leaf Blocks for bitmap indexes consist of:

- ♦ An entry header, containing the number of columns in the index as well as lock information.
- ♦ Key values — however these key values are only stored once. The example in Figure 12-5 has four key values; LIBRA, SPOCK, DRAGON, and FISTER.
- ♦ Start ROWID — the start ROWID is only stored once per distinct key. Oracle uses the start ROWID to construct ROWIDs based upon the bitmap information.
- ♦ End ROWID — the end ROWID is only stored once per distinct key.

- ♦ A bitmap segment consisting of one bit for every row in the table. The bit is set to 1 when the corresponding column value matches the key value. It is unset when the row does not match the key column value. The bitmap segments are compressed by Oracle when stored and must be uncompressed when used. Oracle uses the start ROWID and end ROWID to dynamically construct ROWID, based upon the bit values in the bitmap segments (as seen in Figure 12-4).

Using a bitmap index

Figures 12-4 and 12-5 explain how bitmaps are used by Oracle for processing queries. Oracle uses the same method if retrieving rows for an UPDATE or DELETE operation. When changes are made to the key column value in the table, Oracle is forced to lock the entire bitmap segment. This level of locking is required because there is no bit level locking. It is for this reason that DML operations performed against key columns can lead to locking problems. Because locks are acquired against the entire bitmap segment, a row covered in the bitmap segment cannot be updated by other transactions until the first transaction ends.

Key Value	Start ROWID	End ROWID	Bitmaps for Each Key Value																			
LIBRA	30.1.1	30.6.2	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0
SPOCK	30.1.1	30.6.2	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0
DRAGON	30.1.1	30.6.2	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0	1
FISTER	30.1.1	30.6.2	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0

WHERE CLAUSE:

WHERE MODEL = 'LIBRA' OR MODEL = 'FISTER'

OR Predicates

LIBRA	30.1.1	30.6.2	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0
FISTER	30.1.1	30.6.2	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0

Oracle performs a BIT OR operation and merges the two bitmaps for LIBRA and FISTER into one

Becomes

1	0	1	0	1	1	0	1	1	0	0	1	0	0	0	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

WHERE CLAUSE:

WHERE MODEL NOT IN = 'LIBRA'

NOT IN Predicates

LIBRA	30.1.1	30.6.2	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0
-------	--------	--------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

With NOT IN clauses Oracle just flips the bits.

A 1 becomes a zero. It takes the bitmap where the MODEL = 'LIBRA' and creates a new Bitmap flipping the values.

Becomes

0	1	0	1	0	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Figure 12-5: Bitmap indexes with multiple predicates



The locking problems with bitmaps has been improved in Oracle8.1 and should continue to improve in future releases of Oracle. Oracle has improved its compression decompression algorithm for bitmaps, which means that bitmap segments are locked for a shorter period of time. Many DBAs drop bitmap indexes prior to performing batch updates or inserts on tables with bitmap indexes and then re-create the bitmap when the DML operation completes.

There are two INIT.ORA parameters that affect the performance of bitmaps: `CREATE_BITMAP_AREA_SIZE` and `BITMAP_MERGE_AREA_SIZE`. `CREATE_BITMAP_AREA_SIZE` specifies the amount of memory a bitmap is given when created. Increasing the memory for this type of operation can dramatically improve the creation time. The `BITMAP_MERGE_AREA_SIZE` parameter specifies the amount of memory allocated for merging bitmaps when a bit and or a bit or operation is performed.

Comparing B-tree and bitmap indexes

The following is a comparison of bitmap and B-tree indexes:

- ♦ Bitmap indexes are good for indexes on columns with a low cardinality of values such as gender, marital status, area code, postal code, and so forth. The number of rows returned by these queries is high relative to the number of rows in a table. B-tree indexes are most efficient when there is a high cardinality of the values. The highest cardinality happens on unique indexes where the number of rows returned is always one. Nonunique indexes will have a lower cardinality, but if the cardinality is around one percent, then nonunique indexes are usually more efficient than bitmap indexes.
- ♦ Bitmap indexes are much more compact than B-tree indexes. Bitmap indexes only store the key column value once and does not store the ROWID. The ROWID occupies six bytes of space per row in B-TREE indexes.
- ♦ Bitmap indexes are not as good as B-tree indexes for updates performed against key column values. Updates on B-tree indexes are not forced to lock the entire segment whereas they are for bitmaps.
- ♦ Bitmaps are much more efficient for statements that have multiple predicates such as an AND or OR, AND/OR, and NOT IN. Any of these statements run against columns that have a conventional index usually result in full table scans.
- ♦ For the aforementioned reasons, bitmaps tend to be more beneficial for static data where DML statements are minimal. Also, bitmap indexes work best with large volumes for data where the cardinality is low. Conversely, B-tree indexes require a high cardinality of values. Therefore, B-tree indexes are more suited for OLTP environments and bitmap indexes for Decision Support System (DSS). Although, these guidelines are becoming less strict as Oracle continues to improve bitmap indexes.

Creating B-Tree Indexes

Objective

Create B-tree and bitmap indexes

The same user who owns the table usually creates indexes, however, other users can also create indexes — provided they have the required privileges. Either the INDEX object privilege or the CREATE ANY INDEX system privilege is required to create indexes on another user's table. The user creating the index also needs a quota in the tablespace where the index is being created. Indexes are just segments and are stored the same way as all other segments.

The CREATE INDEX command

The following is the syntax for the CREATE INDEX command:

```
CREATE [ UNIQUE ] INDEX [schema.] index_name
ON [schema.] table
(column [ ASC | DESC ] [, column [ASC | DESC ] ] . . .)
[TABLESPACE tablespace_name]
  [ PCTFREE integer ]
  [ INITRANS integer ]
  [ MAXTRANS integer ]
  [ storage-clause ]
  [LOGGING | NOLOGGING]
  [NOSORT]
```

The keywords from the syntax above have the following meaning:

UNIQUE	Specifies that this is a unique index
Schema	The owner of the table or index
index_name	The name of the index
Table	The name of the table
Column	The column(s) name(s)
ASC/DESC	ASC, the default specifies that the index is created in ascending order. DESC specifies the index is created in descending order. If this is a concatenated index then each column can be either ASC or DESC.
TABLESPACE	Identifies the tablespace where the index will be created. The default if a tablespace is not specified is the user's default tablespace. The user's default tablespace can be obtained from the DBA_USERS data dictionary view.

PCTFREE	The amount of space reserved on the index Leaf Blocks after index creation. PCTFREE for a table controls the minimum amount of free space that must exist on a block before it is removed from the free list. Once removed from the free list, the free space is reserved for future updates. PCTFREE for an index only applies at index creation time. It controls the amount of space reserved on a Leaf Block for future inserts, not updates. Indexes are sorted so newly inserted rows must go to a specific block. If there is not enough free space on that block to hold the newly inserted index value then the index Leaf Block is forced to split. PCTUSED does not apply to index Leaf Blocks. An index Leaf Block can only go back to the free list once all the rows on the block have been deleted.
INITRANS	Controls the number of permanent transaction slots created in the index Leaf Block headers. The default and minimum value is two.
MAXTRANS	Controls the maximum number of transaction slots in the index Leaf Block header. The default is 255.
Storage-clause	Identifies that storage clause that determines how extents are created as well as the number created. The available storage-clause options are INITIAL, NEXT, MINEXTENTS, MAXEXTENTS, PCTINCREASE, BUFFER POOL, and CACHE/NOCACHE.
LOGGING	Specifies if the index creation is to be logged in the redo log files. This is the default value.
NOLOGGING	Specifies that the index creation is not to be logged in the redo log files. This is generally a much quicker way to create indexes. NOLOGGING only applies to the index creation as additional INSERTS, UPDATES, and DELETES will always be LOGGED regardless of the setting of this parameter.
NOSORT	Indicates that the data is already in the index sorted order and does not need to be sorted when the index. If the data is not in the correct order, the index creation will fail.

Here is an example of creating an index on LASTNAME column of the STUDENT table:

```
CREATE INDEX STUDENTS_LNAME_IND ON STUDENTS(LASTNAME)
TABLESPACE INDX
STORAGE (INITIAL 1M NEXT 1M MINEXTENTS 5 PCTINCREASE 0)
PCTFREE 30
INITRANS 3
NOLOGGING;
```

In this example, the index name is STUDENTS_LNAME_IND and five extents are created. Thirty percent of the blocks are going to be reserved for future inserts when the index is being created. The index is created in the INDX tablespace.

Here is an example of creating a concatenated index on LASTNAME and FIRSTNAME columns of the STUDENT table:

```
CREATE INDEX STUDENTS_LNAME_FNAME_IND ON
    STUDENTS(LASTNAME, FIRSTNAME)
    TABLESPACE INDX
    STORAGE (INITIAL 1M NEXT 1M MINEXTENTS 5 PCTINCREASE 0)
    PCTFREE 30
    INITRANS 3
    NOLOGGING;
```

Guidelines for creating indexes

When creating indexes, there are some important guidelines that you should follow.

- ♦ Set the INIT.ORA PARAMETER SORT_AREA_SIZE to a high value for the session creating the index. Since index creation requires sorting the data, increasing this parameter can result in dramatic improvements in index creation times. A value of 30MB can yield excellent results for large indexes. Use the following command to set SORT_AREA_SIZE for your session to 50MB.

```
ALTER SESSION SET SORT_AREA_SIZE = 52428800
```

- ♦ Ensure that your temporary tablespace is a true temporary tablespace. Also, make sure that the tablespace where the index is being created and the temporary tablespace do not share the same disks.
- ♦ Do not over index. Indexes speed up SQL statements only if the column being indexed is referenced in the WHERE clause, ORDER BY clause, GROUP BY clause, or a column that is joined. Indexes slow down certain types of DML statements so indexes that are not being used frequently can negatively affect performance.
- ♦ Create separate tablespaces for indexes. Ensure that indexes do not share tablespaces with rollback segments, temporary segments, or the tables they are indexing. It is OK to have indexes share tablespaces with tables, so long as the table being indexed is not on the same tablespace. But you could have the index for the SCHEDULEDCLASSES table in one tablespace with the STUDENT table and have the indexes for the STUDENT table and the SCHEDULEDCLASSES table data in another tablespace.
- ♦ Use standard extent sizes for your indexes making sure that they are a multiple of 5 * DB_BLOCK_SIZE.
- ♦ Use the NOLOGGING option for creating indexes. This can dramatically speed up index creation especially for large indexes. If the index is lost because it was not logged, it can simply be re-created. There is no risk of losing any data by NOLOGGING an index. The LOGGING/NOLOGGING attribute is inherited from the tablespace if it is not specified so consider setting the NOLOGGING default attribute for your index tablespaces.
- ♦ Ensure that INITRANS is higher than it is for tables. Index Leaf Blocks have many more entries than do tables so the likelihood of contention is increased for indexes.

- ♦ Set PCTFREE to a low value for indexes on columns that uses sequences. You want to ensure that blocks are packed as tightly as possible when the index is created because sequences always use the right-hand most Leaf Block. The fewer number of Leaf Blocks, the more efficient the index. For nonunique indexes on columns that have a more even distribution of values, consider setting PCTFREE to a higher value. This helps prevent index block splits.

Creating bitmap indexes

The syntax for creating a bitmap index is similar to that of a B-tree index with two exceptions: They cannot be unique and you must include the key word BITMAP. The CREATE_BITMAP_AREA_SIZE parameter controls the amount of memory that is allocated for the creation of the bitmap index. The default is 8MB but should be increased if the cardinality for the column is high. If there are many distinct key values, try setting the parameter to 15MB or 20MB. If cardinality is low — that is, there are only a few distinct values — then the default of 8MB is more than ample.

```
CREATE BITMAP INDEX [schema.] index_name
ON [schema.] table
(column [ ASC | DESC ] [, column [ASC | DESC ] ] . . .)
[TABLESPACE tablespace_name]
  [ PCTFREE integer ]
  [ INITRANS integer ]
  [ MAXTRANS integer ]
  [ storage-clause ]
  [LOGGING | NOLOGGING]
  [NOSORT]
```

Here is an example of creating a bitmap index on the DAYSDURATION column of the SCHEDULEDCLASSES table:

```
CREATE BITMAP INDEX SCHED_DAYS DUR_BM ON
SCHEDULEDCLASSES (DAYSDURATION)
TABLESPACE INDX
STORAGE (INITIAL 1M NEXT 1M MINEXTENTS 5 PCTINCREASE 0)
PCTFREE 30
INITRANS 3
NOLOGGING;
```

Creating Reverse Key indexes

The syntax for creating a Reverse Key index is almost identical to that of creating a regular B-tree index. The only two differences are that NOSORT cannot be specified and the keyword REVERSE must be included. Here is the syntax:

```
CREATE [ UNIQUE ] INDEX [schema.] index_name
ON [schema.] table
(column [ ASC | DESC ] [, column [ASC | DESC ] ] . . .)
```

```
[TABLESPACE tablespace_name]
 [ PCTFREE integer ]
 [ INITRANS integer ]
 [ MAXTRANS integer ]
 [ storage-clause ]
 [LOGGING | NOLOGGING]
 REVERSE
```

Here is an example of creating a Reverse Key index on the CLASSID column of the SCHEDULEDCLASSES table:

```
CREATE UNIQUE INDEX SCHED_CLASSID_UN_R ON
SCHEDULEDCLASSES (CLASSID)
TABLESPACE INDX
STORAGE (INITIAL 1M NEXT 1M MINEXTENTS 5 PCTINCREASE 0)
PCTFREE 30
INITRANS 3
NOLOGGING
REVERSE;
```

Creating function-based indexes

Once you have identified the required functions in need of an index, all that needs to be done is to create the index(es). Only privileged users can create function-based indexes. The syntax is similar to the regular B-tree indexes. Here is the syntax:

```
CREATE INDEX [schema.] index_name
ON [schema.] table
([FUNCTION | EXPRESSION])
[TABLESPACE tablespace_name]
 [ PCTFREE integer ]
 [ INITRANS integer ]
 [ MAXTRANS integer ]
 [ storage-clause ]
 [LOGGING | NOLOGGING]
```

Here is an example of creating a function-based index on UPPER(LASTNAME) of the STUDENTS table:

```
CREATE INDEX STUDENT_UPPER_LNAME ON
STUDENT(UPPER(LASTNAME));
```

Creating indexes using OEM

Use the following steps to create indexes using OEM:

STEP BY STEP: Creating Indexes with DBA Studio

1. Launch DBA Studio and select the Launch DBA Studio in Stand Alone option. Start ⇨ Programs ⇨ Oracle OEM (This name may be different on your installation) ⇨ Database Administration ⇨ DBA Studio.
2. Select the CERTDB database and enter the login information if required.
3. Select Object from the menu bar.
4. Select Create.
5. Select Index and click on the Create Button.
6. On the General page, seen in Figure 12-6, enter the index name, the tablespace, the table, the columns, and the type of index to create.

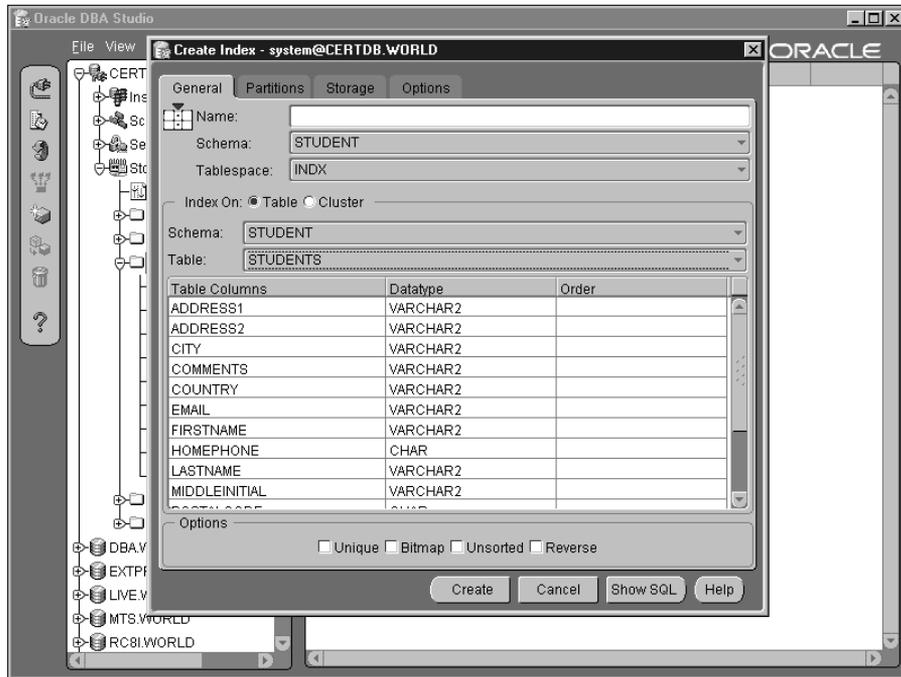


Figure 12-6: Enter index creation information

7. Select the Storage page, seen in Figure 12-7, and enter the values for Initial Size, Next Size and Increase Size by fields.

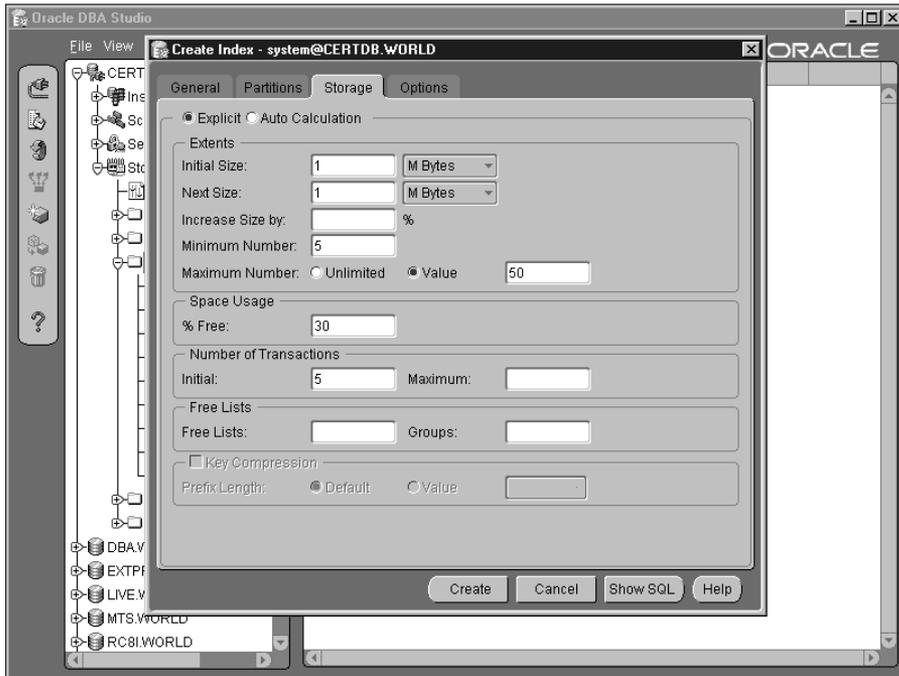


Figure 12-7: Enter index storage Information

8. Select the Options page, seen in Figure 12-8, and specify whether the index creation is to be LOGGED or NOLOGGED.

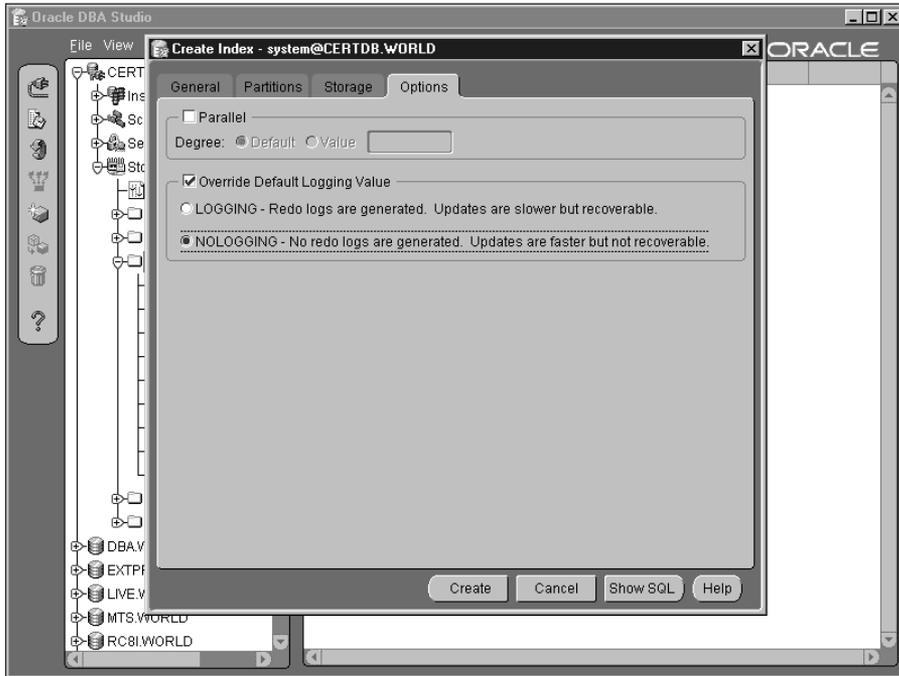


Figure 12-8: Index options

9. Click the Create button when done.

Modifying Indexes

Objective

Reorganize indexes

Altering storage parameters

A limited number of parameters can be changed by modifying an index. Changing these parameters is similar to that of changing parameters for a table. The effect is not immediate. To have the parameters take affect, the index should be reorganized. You must be the owner of the index or have the ALTER ANY INDEX system privilege to modify an index in another user's schema. Here is the syntax of the ALTER INDEX command:

```
ALTER INDEX [schema.] index_name
  [storage-clause]
  [INITRANS integer]
```

```
[MAXTRANS integer]
```

Allocating and deallocating index space

Like all other segments, indexes may need to have additional extents created or have extra extents deallocated to free up disk space. Oracle adds extents dynamically for indexes, but it is always better to pre-allocate extents to avoid expensive dynamic extent allocation. Here is the syntax for manually allocating extents:

```
ALTER INDEX [schema.]index
  ALLOCATE EXTENT ([SIZE integer [K|M]]
    [DATAFILE 'filename'])
```

If the `SIZE` option were not specified, the size used would be the size of the next extent if it were created automatically. The size of the next extent is stored in the `NEXT_EXTENT` column of the `DBA_INDEXES` data dictionary view.

Manually deallocating space enables you to deallocate to a specific size or to deallocate down to the high-water mark of an index. You can never deallocate space below the high-water mark. Reorganizing the index is the only way to lower the high-water mark. Here is the syntax for deallocating extents:

```
ALTER INDEX [schema.]index
  DEALLOCATE UNUSED ([KEEP integer [K|M]])
```

Rebuilding indexes

Rebuilding indexes is one of a DBA's main duties. Certain indexes need to be rebuilt more regularly than others. Indexes on volatile tables are usually the ones that get rebuilt the most. There are other situations in which indexes need to be rebuilt. Here are some of the situations:

- ♦ Indexes should be on their own tablespaces or at a minimum on different tablespaces from rollback segments, temporary segments, and the tables they are indexing. This helps reduce IO bottlenecks that indexes can create. If an index and the table it is indexing share the same tablespace, then index lookups cause IO to be performed on the same tablespace, which are usually the same physical disks. One of the index rebuild options is to relocate the index to a different tablespace.
- ♦ Indexes on volatile tables that are subject to many inserts, deletes, and updates need to be rebuilt more frequently. Space on indexes is not reused efficiently simply because the data must be sorted. This limits the amount of block reuse that Oracle can perform. If a table has many deletes, the index blocks cannot be reused until all the index entries have been deleted. This is much different than tables where `PCTUSED` controls when space on blocks can be reused. `PCTUSED` cannot be specified for an index. If the percentage of deleted rows (`DEL_LF_ROWS` column of `INDEX_STATS` data dictionary view) is

20 percent of the total rows (LF_ROWS column of INDEX_STATS data dictionary view), then the index should be rebuilt. Also, when the height of an index gets above four, the index should be rebuilt. The height of an index is stored in the BLEVEL column of DBA_INDEXES and is only populated when index statistics are calculated. Use the ANALYZE INDEX index_name COMPUTE STATISTICS command to generate index statistics.

- ♦ Converting regular B-tree indexes to Reverse Key or vice versa can be accomplished through a rebuild.
- ♦ If storage parameters are changed for an index, many parameters will only take effect after the index is rebuilt.
- ♦ If a table is moved using the ALTER TABLE table_name MOVE command, all the indexes on that table are unusable and must be rebuilt.

The following are the characteristics of index rebuilds:

- ♦ Index rebuilds use the existing index to create the new one. This avoids having to perform a full table scan on the table and then sort an operation that is expensive. By using the existing index, Oracle simply performs an index scan and creates a new index. You will require twice as much space, however, because the old index is not dropped until the new one is created. If the index is 10MB in size, you will require 20MB of disk space.
- ♦ The new index does not contain any deleted entries, and the new storage parameters for PCTFREE, INITRANS, and MAXTRANS are used. This results in fewer blocks being needed to store the index information. Fewer index blocks always yield faster index retrievals.
- ♦ The old index can still be used while the new index is being built. DML operations are not permitted.

Here is the syntax of the ALTER INDEX index_name REBUILD command:

```
ALTER INDEX [schema.] index_name REBUILD
[ TABLESPACE tablespace_name ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ storage-clause ]
[ LOGGIN | NOLOGGING ]
[ REVERSE | NOREVERSE ]
```

Here is an example of rebuilding the PK_INSTRUCTORID index on the INSTRUCTORID column of the INSTRUCTORS table:

```
ALTER INDEX PK_INSTRUCTORID REBUILD
PCTFREE 5;
```

This rebuilds the index on the same tablespace but changes the PCTFREE value to five percent.

Here is an example of rebuilding the PK_CLASSID index on the CLASSID column of the SCHEDULEDCLASSES table. The index is currently a normal B-tree index and it is going to be rebuilt as a Reverse Key index.

```
ALTER INDEX PK_CLASSID REBUILD  
REVERSE;
```

Assuming in this example that a new tablespace was created called INDEX_NEW for storing indexes, the following command could be used to relocate indexes to the new tablespace.

```
ALTER INDEX PK_CLASSID REBUILD  
TABLESPACE INDEX_NEW;
```

Rebuilding indexes online

The only problem with rebuilding indexes is that DML operations cannot be performed because the table is locked. Queries are allowed, but if your database is a 7-and-24 database, restricting DML is not usually an option. This limits the number of times that an index could be rebuilt and when a DBA could perform the command. In Oracle8.1, Oracle introduced online index rebuilds.

An online index rebuilds the index without restricting query or DML operations. When the online index rebuild is started, any DML statements that affect the index are journalled. When the index rebuild is done, Oracle then merges all the journalled changes into the index. The DML locks are still used, which prevents DDL commands against the index or table.

There are some restrictions to online index rebuilds:

- ♦ Indexes on temporary tables cannot be rebuilt online.
- ♦ Indexes on an entire partition indexed must be rebuilt partition by partition.
- ♦ You cannot also deallocate unused space.
- ♦ You cannot change the value of PCTFREE for the index as a whole.

Here is an example of rebuilding the PK_CLASSID index on the CLASSID column of the SCHEDULEDCLASSES table. This example rebuilds the index on a new tablespace called INDEX_NEW, as well as rebuilding it as a Reverse Key index.

```
ALTER INDEX PK_CLASSID REBUILD ONLINE  
TABLESPACE INDEX_NEW  
REVERSE;
```

To perform an online rebuild of the same index without changing the tablespace or making it a Reverse Key index, use the following command:

```
ALTER INDEX PK_CLASSID REBUILD ONLINE;
```

Coalescing indexes

Coalescing index Leaf Blocks just reorganizes the Leaf Blocks by merging adjacent blocks. If two adjacent blocks are merged, when one becomes free, it can be used by other index entries as new Leaf Blocks are needed. The index itself is not rebuilt so you cannot specify storage parameters or a new tablespace. This is merely a way of reorganizing the Leaf Blocks. This is just another option instead of rebuilding. Here is an example of coalescing the PK_CLASSID index:

```
ALTER INDEX PK_CLASSID COALESCE;
```

The end result is that Leaf Blocks should be packed better and blocks should be freed making the index more efficient. This operation is quicker than creating or rebuilding an index.

Dropping indexes

Objective

Drop indexes

Many indexes are created temporarily. If month end reports are running, they may benefit from indexing, however, the day-to-day operations will be slowed by the indexes. To offset this effect, DBAs often create indexes for month end reporting and then simply drop them when the reports are complete. Another common use for dropping indexes is when large DML jobs are to be run or when performing bulk loads of data. When large DML statements or bulk loads are running, not only are they updating and inserting and deleting from the table, they are also maintaining the index. The total time taken to drop the index, load the data, and then re-create the index is usually much quicker than just loading the data with the indexes in place.

Indexes that are no longer needed because alternative methods for retrieving data have been found or the statement or statements that required the index are no longer run should be deleted. Indexes may also become corrupt and need to be dropped. Whatever the reason for the drop, the syntax for the command is simple:

```
DROP INDEX [schema.] index_name;
```

Indexes can also be dropped automatically by Oracle when other commands eliminate the need for the index.

- ♦ When a table is dropped, all its associated indexes are dropped.
- ♦ When a table is truncated, the index is not dropped but all the space is deallocated.
- ♦ Indexes on PRIMARY KEY or UNIQUE constraints are dropped when the constraint is disabled or dropped, but only if Oracle created the index when the

constraint was created. If this occurred before the constraint as a non-unique or unique index, then the index is not dropped when the constraint is dropped.

- ♦ When you drop a column, all indexes referencing that column are dropped.

Getting Information on Indexes

Objective

Get index information from the data dictionary

Data dictionary views

The two main views for extracting index information are `DBA_INDEXES` and `DBA_IND_COLUMNS`. Both these views have columns that become populated when the index is created or altered, but they also have some columns that only become populated when the index is analyzed, similar to a table. Since indexes are just segments, the `DBA_OBJECTS`, `DBA_SEGMENTS`, and `DBA_EXTENTS` views also hold index information. The `INDEX_STATS` view only contains one row, the row from the index that had the last `VALIDATE STRUCTURE` command issued against it. The `INDEX_STATS` view has statistical information about the Leaf Blocks.

DBA_INDEXES

The `DBA_INDEXES` data dictionary view contains one row for every index in the database. Here is a list of the column names and descriptions for the `DBA_INDEXES` data dictionary view.

<i>COLUMN_NAME</i>	<i>DESCRIPTION</i>
OWNER	Username of the owner of the index
INDEX_NAME	Name of the index
INDEX_TYPE	Type of index
TABLE_OWNER	Owner of the indexed object
TABLE_NAME	Name of the indexed object
TABLE_TYPE	Type of the indexed object
UNIQUENESS	UNIQUE or NONUNIQUE
COMPRESSION	Is Compressions ENABLED or DISABLED? This is used for concatenated indexes
PREFIX_LENGTH	Number of columns in the prefix of the key used for compression
TABLESPACE_NAME	Name of the tablespace containing the index
INI_TRANS	Number of transaction slots in the Leaf Block headers

Continued

<i>COLUMN_NAME</i>	<i>DESCRIPTION</i>
MAX_TRANS	Maximum number of transaction slots in the Leaf Block header
INITIAL_EXTENT	Initial extent size in bytes
NEXT_EXTENT	Size of the next extent to be created. This will increase if PCTINCREASE is a non zero value
MIN_EXTENTS	Number of extents created when index was created
MAX_EXTENTS	Maximum number of allowable extents for the index
PCT_INCREASE	Percentage increase of the extent size
PCT_THRESHOLD	Threshold percentage of block space allowed per index entry
INCLUDE_COLUMN	User column ID for last column to be included in index organized table top index
FREELISTS	Number of freelists for the index
FREELIST_GROUPS	Number of freelist groups for the index
PCT_FREE	Minimum percentage of free space in a block reserved when index is created. This does not have the same meaning as it does for tables.
LOGGING	Logging Attribute for the index. If the index is set to NOLOGGING then certain DIRECT load insert operations are not logged in the redo log files. Regular DML statements are always logged.
BLEVEL	This is the depth of the index from the root block to the Leaf Block. If the blevel is greater than 4 it is time to rebuild the index.
LEAF_BLOCKS	Number of Leaf Blocks in the index
DISTINCT_KEYS	Number of distinct keys in the index
AVG_LEAF_BLOCKS_ PER_KEY	The average number of Leaf Blocks per distinct key
AVG_DATA_BLOCKS_ PER_KEY	The average number of data blocks per distinct key
CLUSTERING_FACTOR	A measurement of the amount of (dis)order of the table data relative to the index
STATUS	Status of the index: VALID or UNUSABLE. UNUSABLE index will need to be rebuilt.
NUM_ROWS	Number of index entries
SAMPLE_SIZE	Sample size used in the last ANALYZE command
LAST_ANALYZED	Date the index was last analyzed
DEGREE	The default degree of parallelism for the index
INSTANCES	Number of instances across which the index is to be scanned

<i>COLUMN_NAME</i>	<i>DESCRIPTION</i>
PARTITIONED	Set to YES if this is a partitioned index
TEMPORARY	Is this a temporary or permanent index? The term temporary refers to the data in the index as opposed to the definition of the index.
GENERATED	Was the name of the index system generated?
SECONDARY	Is the index object created as part of icreate for domain indexes?
BUFFER_POOL	To which buffer pool does the index belong?
USER_STATS	Were the statistics entered directly by the user?
DURATION	The duration of the index data for temporary indexes
PCT_DIRECT_ACCESS	If index is an IOT, then this is the percentage of rows with a valid estimate.
ITYP_OWNER	If domain index, then this is the indextype owner
ITYP_NAME	If domain index, then this is the indextype name
PARAMETERS	If domain index, then this is the parameter string
GLOBAL_STATS	Were statistics captured globally for all partitions?
DOMIDX_STATUS	Is the indextype of the domain index valid?
DOMIDX_OPSTATUS	Status of the operation on the domain index
FUNCIDX_STATUS	Is the function-based index ENABLED or DISABLED?

DBA_IND_COLUMNS

The DBA_IND_COLUMNS view contains one row for every column referenced in an index. For indexes on single columns, there is only one row. For concatenated columns there is one row for every column in the index.

<i>COLUMN_NAME</i>	<i>DESCRIPTION</i>
INDEX_OWNER	Username of the owner of the index
INDEX_NAME	Name of the index
TABLE_OWNER	Owner of the indexed object
TABLE_NAME	Name of the indexed object
COLUMN_NAME	Name of column
COLUMN_POSITION	Position of column attribute within index
COLUMN_LENGTH	Length of the column
DESCEND	Was the index created as a descending index?

Indexes become INVALID usually when the table they are created against is rebuilt or when the indexes are temporarily disabled for a bulk load or update command. Use this query to list all the INVALID indexes:

```
SQL> SELECT      INDEX_NAME
  2  , TABLESPACE_NAME
  3  , INDEX_TYPE
  4  , UNIQUENESS
  5  , STATUS
  6  FROM DBA_INDEXES
  7  WHERE STATUS <> 'VALID'
  8  /
```

INDEX_NAME	TABLESPACE_NAME	INDEX_TYPE	UNIQUENESS	STATUS
PK_CLASSID	CERTDB	NORMAL	UNIQUE	UNUSABLE

To list all the indexes owned by the user STUDENT along with some index characteristics, run the following query:

```
SQL> SELECT      INDEX_NAME
  2  , INDEX_TYPE
  3  , UNIQUENESS
  4  , STATUS
  5  , BLEVEL
  6  , LEAF_BLOCKS "L_BLOCKS"
  7  , DISTINCT_KEYS "D_KEYS"
  8  FROM DBA_INDEXES
  9  WHERE OWNER = 'STUDENT'
 10  /
```

INDEX_NAME	INDEX_TYPE	UNIQUENESS	STATUS	BLEVEL	L_BLOCKS	D_KEYS
BATCHJOBS_JOBID_PK	NORMAL	UNIQUE	VALID	0	1	3
COURSEAUDIT_PK	NORMAL	UNIQUE	VALID			
EMP_EMPNO_PK	NORMAL	UNIQUE	VALID	0	1	200
I_CLUST_EMP_DEPT	CLUSTER	UNIQUE	VALID			
PK_CLASSID	NORMAL	UNIQUE	UNUSABLE	0	1	3
PK_CLASSID_STUDENT NUMBER	NORMAL	UNIQUE	VALID	0	1	7
PK_COURSENUMBER	NORMAL	UNIQUE	VALID			
PK_INSTRUCTORID	NORMAL	UNIQUE	VALID			
PK_LOCATIONID	NORMAL	UNIQUE	VALID	0	1	3
PK_STUDENTNUMBER	NORMAL	UNIQUE	VALID			
STUDENT_LNAME_IND	NORMAL	NONUNIQUE	VALID			
SYS_C001395	NORMAL	UNIQUE	VALID			

SYS_C001410	NORMAL	UNIQUE	VALID	0	1	1
SYS_C001411	NORMAL	UNIQUE	VALID	0	1	1
SYS_IL0000020697C0	LOB	UNIQUE	VALID			
0003\$\$	LOB	UNIQUE	VALID			

Notice that the BLEVEL, Leaf Blocks (L BLOCKS), and Distinct Keys (D_KEYS) columns are null for some indexes. These columns are only populated when statistics are calculated for the index. Once the statistics are calculated, these columns will have values. Use the ANALYZE INDEX index_name COMPUTE STATISTICS command to generate index statistics.

The following query reports indexes that share the same tablespace as the table they are indexing. You should always avoid this and re-create these indexes into index only tablespace.

```
SQL> SELECT      OUTER.OWNER
  2 ,            OUTER.INDEX_NAME
  3 ,            OUTER.TABLE_NAME
  4 ,            OUTER.TABLESPACE_NAME
  5 FROM          DBA_INDEXES OUTER
  6 WHERE         EXISTS (SELECT 'A'
  7                FROM    DBA_TABLES INNER
  8                WHERE   INNER.OWNER = OUTER.OWNER
  9                AND     INNER.TABLE_NAME = OUTER.TABLE_NAME
 10                AND     INNER.TABLESPACE_NAME = OUTER.TABLESPACE_NAME
 11                AND     INNER.OWNER NOT IN ('SYS','SYSTEM'))
SQL> /
```

OWNER	INDEX_NAME	TABLE_NAME	TABLESPACE_NAME
-----	-----	-----	-----
OUTLN	OL\$NAME	OL\$	SYSTEM
OUTLN	OL\$SIGNATURE	OL\$	SYSTEM
OUTLN	OL\$HNT_NUM	OL\$HINTS	SYSTEM
SCOTT	PK_DEPT	DEPT	SYSTEM
SCOTT	PK_EMP	EMP	SYSTEM
STUDENT	PK_COURSENUMBER	COURSES	CERTDB
STUDENT	PK_INSTRUCTORID	INSTRUCTORS	CERTDB
STUDENT	PK_LOCATIONID	LOCATIONS	CERTDB
STUDENT	PK_STUDENTNUMBER	STUDENTS	CERTDB
STUDENT	PK_CLASSID	SCHEDULEDCLASSES	CERTDB
STUDENT	PK_CLASSID_STUDENT	CLASSENROLLMENT	CERTDB
	NUMBER		
STUDENT	BATCHJOBS_JOBID_PK	BATCHJOBS	CERTDB
STUDENT	COURSEAUDIT_PK	COURSEAUDIT	CERTDB
STUDENT	SYS_IL0000020697C0	ORD	CERTDB
	0003\$\$		

STUDENT	SYS_C001395	ORDERS	CERTDB
STUDENT	SYS_C001410	CUSTOMER	CERTDB
STUDENT	SYS_C001411	SUPPLIERS	CERTDB

The following query lists all the indexes whose names have been generated by Oracle. This query includes the column name as well, and these indexes should be renamed with a more meaningful and intuitive name.

```
SQL> SELECT DI.INDEX_NAME
2 ,      DI.TABLE_NAME
3 ,      DIC.COLUMN_NAME
4 ,      DI.OWNER
5 ,      DI.GENERATED
6 FROM   DBA_INDEXES DI
7 ,      DBA_IND_COLUMNS DIC
8 WHERE  DI.GENERATED = 'Y'
9 AND    DI.OWNER = DIC.INDEX_OWNER
10 AND   DI.TABLE_NAME = DIC.TABLE_NAME
11 AND   DI.INDEX_NAME = DIC.INDEX_NAME
12 AND   DI.OWNER NOT IN ('SYS','SYSTEM')
13 /
```

INDEX_NAME	TABLE_NAME	COLUMN_NAME	OWNER	G
SYS_C001395	ORDERS	SYS_NC0000300004\$	STUDENT	Y
SYS_C001410	CUSTOMER	CUSTID	STUDENT	Y
SYS_C001411	SUPPLIERS	SUPPLIERID	STUDENT	Y

The following query lists the table name, column name, and index name for all indexes owned by the user STUDENT:

```
SQL> SELECT DI.INDEX_NAME
2 ,      DI.TABLE_NAME
3 ,      DIC.COLUMN_NAME
4 ,      DI.OWNER
5 FROM   DBA_INDEXES DI
6 ,      DBA_IND_COLUMNS DIC
7 WHERE  DI.OWNER = DIC.INDEX_OWNER
8 AND    DI.TABLE_NAME = DIC.TABLE_NAME
9 AND    DI.INDEX_NAME = DIC.INDEX_NAME
10 AND   DI.OWNER = 'STUDENT'
11 /
```

INDEX_NAME	TABLE_NAME	COLUMN_NAME	OWNER
BATCHJOBS_JOBID_PK	BATCHJOBS	JOBID	STUDENT
COURSEAUDIT_PK	COURSEAUDIT	COURSENUMBER	STUDENT

COURSEAUDIT_PK	COURSEAUDIT	CHANGE	STUDENT
COURSEAUDIT_PK	COURSEAUDIT	DATECHANGED	STUDENT
EMP_EMPNO_PK	EMPLOYEE	EMPNO	STUDENT
I_CLUSTER_EMP_DEPT	CLUSTER_EMP_DEPT	DEPTNO	STUDENT
PK_CLASSID	SCHEDULEDCLASSES	CLASSID	STUDENT
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	CLASSID	STUDENT
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	STUDENTNUMBER	STUDENT
PK_COURSENUMBER	COURSES	COURSENUMBER	STUDENT
PK_INSTRUCTORID	INSTRUCTORS	INSTRUCTORID	STUDENT
PK_LOCATIONID	LOCATIONS	LOCATIONID	STUDENT
PK_STUDENTNUMBER	STUDENTS	STUDENTNUMBER	STUDENT
STUDENT_LNAME_IND	STUDENTS	LASTNAME	STUDENT
SYS_C001395	ORDERS	SYS_NC0000300004\$	STUDENT
SYS_C001410	CUSTOMER	CUSTID	STUDENT
SYS_C001411	SUPPLIERS	SUPPLIERID	STUDENT

Getting index information using OEM

The following steps are used to extract index information from OEM.

STEP BY STEP: Getting Index Information with DBA Studio

1. Launch DBA Studio and select the Launch DBA Studio in Stand Alone option.
Start ⇨ Programs ⇨ Oracle OEM (This name may be different on your installation) ⇨ Database Administration ⇨ DBA Studio.
2. Select the CERTDB database and enter the login information if required.
3. Click on Schema.
4. Click on Index.
5. Click on the user STUDENT.
6. Double-click on the index you want information on.
7. You can now toggle through the General, Storage, Options, and Statistics pages to extract index information as seen in Figure 12-9.

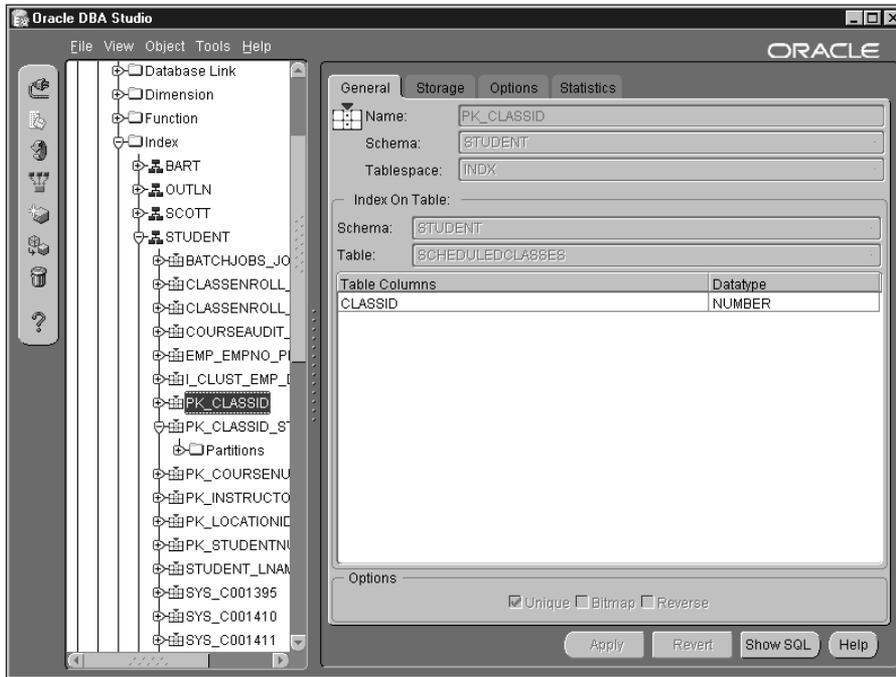


Figure 12-9: Getting index information from DBA Studio

Key Point Summary

In preparing for the Oracle®i DBA: Architecture and Administration exam, please keep these points in mind regarding managing indexes:

- ♦ Indexes generally speed up queries and slow down DML statements. In order for a query to benefit from an index, the index should generally exist on a column referenced in either the WHERE, GROUP BY, ORDER BY, or join condition. There are some exceptions to these rules.
- ♦ The logical elements of indexes in Oracle are: Single column or Concatenated, Unique or non-unique, or function- or expression-based.
- ♦ Indexes are just segments and like all other segments require storage. Indexes can be partitioned or nonpartitioned and are stored in tablespaces like all other segments. Indexes should be stored in their own tablespaces and on separate file systems from the tables that the indexes are indexing.
- ♦ The different types of indexes are: B-tree, Function Based, Reverse Key, and Bitmap.

- ♦ B-tree indexes are most useful on columns with a high cardinality of values where Bitmaps are more beneficial on low cardinality columns.
- ♦ Reverse Key indexes are beneficial for monotonically increasing unique indexes such as indexes on Primary Key columns where an Oracle Sequence generates the key.
- ♦ Function-based indexes store the result of a function or expression in the index, making the index more useful if the function or expression is regularly used to retrieve data.
- ♦ When creating indexes, consider using the `nologging` option to speed up the index creation. You should also increase the `SORT_AREA_SIZE` parameter and be sure that the tablespace the index is being created on is not the same as the temporary tablespace used for sorting the index.
- ♦ Indexes should be analyzed and information from the data dictionary views `DBA_INDEXES`, `DBA_IND_COLUMNS`, and `INDEX_STATS` used to determine when the index should be rebuilt.
- ♦ Always rebuild indexes versus dropping and recreating them. It is much faster.
- ♦ When creating or rebuilding an index, consider using the `ONLINE` option to avoid locking the base table of the index.

STUDY GUIDE

Now that you have learned about managing indexes, you should test your understanding by reviewing the assessment questions and performing the exercises below.

Assessment Questions

1. What is the maximum number of columns allowed in a concatenated index?
 - A. 32
 - B. 16
 - C. 8
 - D. 4
 - E. 2
2. The format of an Index Leaf Entry is made up of an Entry Header, the Key Column Length, Key Column Value, and the _____. (Select the answer that completes this statement.)
 - A. Table Name
 - B. Column Name
 - C. ROWID
 - D. Row Number
 - E. Number of Distinct Keys
3. What storage parameter specifies the amount of space reserved on an Index Leaf Block for future INSERTS?
 - A. INITIAL
 - B. NEXT
 - C. FREE_SPACE
 - D. INTRANS
 - E. PCTFREE

4. What characteristic of Reverse Key indexes is true?
 - A. The index entries are smaller.
 - B. The indexes are stored in descending order.
 - C. They can only be stored in bitmap indexes.
 - D. PCTFREE cannot be specified.
 - E. The key values are reversed in the index.

5. What is true of bitmap indexes?
 - A. They are not very useful in read-only databases or tables.
 - B. They require more storage than a B-tree index.
 - C. They are not very efficient for multiple predicate SQL statements.
 - D. They are very efficient for multiple predicate SQL statements.
 - E. No DML is allowed in tables that use bitmap indexes.

6. If using the NOSORT option of the CREATE INDEX statement, what condition must be true?
 - A. The table data must already be in the sorted order that the index needs.
 - B. The index will not be in a sorted order.
 - C. The index will be in a sorted order but only if the UNIQUE option is also included.
 - D. Prevents sorts from happening to disk during index creation.
 - E. The NOSORT option can only be used when rebuilding indexes.

7. What is true about rebuilding indexes ONLINE?
 - A. The database does not need to be shutdown to rebuild the index.
 - B. Queries are permitted.
 - C. DDL statements are permitted against the table but no DML statements.
 - D. Both query and DML operations are permitted.
 - E. The index does not need to be taken offline to rebuild it.

8. Which command would you use to rebuild an index called IND1 onto tablespace IND_TS?
 - A. ALTER TABLE EMP REBUILD INDEX IND1 TABLESPACE IND_TS;
 - B. ALTER INDEX IND1 REBUILD IND_TS;
 - C. ALTER INDEX IND1 REBUILD STORAGE (TABLESPACE IND_TS);
 - D. ALTER INDEX IND1 IND_TS TABLESPACE IND1;
 - E. ALTER INDEX IND1 REBUILD TABLESPACE IND_TS;

9. What data dictionary view contains a row for every index in the database?
- A. DBA_INDEXES
 - B. DBA_INDEX
 - C. V\$INDEXES
 - D. DBAINDEXES
 - E. DBA_IND_COLUMNS
10. What data dictionary view contains a row for every column referenced by an index in the database?
- A. DBA_INDEXES
 - B. DBA_INDEX
 - C. DBA_IND_COLUMN
 - D. DBA_INDEX_COLUMNS
 - E. DBA_IND_COLUMNS

Scenarios

1. You are a DBA and your users are starting to complain about the performance of the database. Certain INSERT operations on busy tables are taking longer than normal to complete. You have determined that the table that is having the problems is the SCHEDULEDCLASSES table. The table has a column called CLASSID that is populated by a SEQUENCE and there are 15,000 rows in the table. It is a regular B-tree index that you have been rebuilding regularly. This determination stops the user complaints for a short period, but performance gradually gets worse to a point where you are forced to rebuild the index. You are always hesitant about rebuilding the index because the table is locked to prevent users from performing DML statements while the index is rebuilding. Both the table and the index are on the CERTDB tablespace and you notice that most of the queries using the CLASSID column use an equality predicate.
- A. What changes should be made to the index to improve the performance and reduce the frequency of the rebuilds?
 - B. What can be done to prevent users from not being able to perform DML operations while the index is being rebuilt?
2. You have been hired as a consultant by HANNAH and JACOB enterprises to try and improve the time that it takes to create some large indexes. The creation of the indexes is taking 14 hours to complete, which is causing some tremendous locking performance problems. The indexes cannot be rebuilt because there is not enough physical space to perform this operation. The indexes must be dropped first and then re-created.

- A. What recommendations would you make to improve the creation time of the large indexes?

Lab Exercises

Lab 12-1 Creating Indexes

1. Using SQLPLUS line mode (sqlplus), connect to your instance as user STUDENT/ORACLE.
2. Query the data dictionary to find out what indexes exist as well as the type of index for the CLASSENROLLMENT table.
3. Create a normal B-tree index on the ENROLLMENTDATE column of the CLASSENROLLMENT table. Make sure that the index is created on the INDX tablespace and that PCTFREE value is set to 30. The index name should be CLASSENROLL_ENROLLDATE and you should create 4 extents all 100KB in size.
4. Query the data dictionary to verify the creation of the index.
5. Create a bitmap index on the STATUS column of the CLASSENROLLMENT table using the same parameters as question 3. The index name should be CLASSENROLL_STATUS_BM.
6. Query the data dictionary to verify the creation of the index.
7. Create a function-based index on LASTNAME column of the INSTRUCTORS table using the UPPER function. The index should be called INSTRUCTOR_LNAME_UPPER and have the same storage parameters as the index created in question 3.
8. What happens and why?
9. Using SQLPLUS line mode (sqlplus), connect to your instance as a user with SYSDBA privileges.
10. Attempt to create the function-based index again, making sure that the index is created under the STUDENT schema.

Lab 12-2 Managing Indexes

1. Using SQLPLUS line mode (sqlplus), connect to your instance as user STUDENT/ORACLE.
2. Query the data dictionary and find all the indexes owned by the user STUDENT that share the same tablespace as the table they are indexing.
3. Relocate all the indexes for the SCHEDULEDCLASSES and CLASSENROLLMENT tables to the INDX tablespace.

4. Rebuild the PK_INSTRUCTORID index changing it to a Reverse Key index and locating it on the INDX tablespace.
5. Rebuild the PK_COURSENUMBER index on the COURSES table ONLINE as well as relocating it to the INDX tablespace.

Answers to Chapter Questions

Chapter Pre-Test

1. The main benefit of an index is that it can improve query performance. Indexes are stored in a sorted order; therefore queries do not need to scan every row in a table. Indexes also improve the performance of joins and the order by and group by queries. To improve query performance, the index generally needs to be on the columns referenced in the where clause.
2. A concatenated index is an index on more than one column. For example, you could have an index on lastname and firstname of the INSTRUCTORS table if they were queried together.
3. A Reverse Key index stores the indexed values in reverse order. If the CLASS-SID of the SCHEDULEDCLASSES table was being stored in a Reverse Key index then CLASSID 51 would be stored as 15. Reverse Key indexes provide for a better distribution of index values for indexes on monotonically increasing columns, columns that are populated using sequences like a PRIMARY KEY.
4. Indexes store the column data that is being indexed and the ROWID of the row the column belongs to. If there was an index on the LASTNAME column of the INSTRUCTORS table then the index would store the values like Cross, Williams, and Keele, along with the ROWID these values belong to from the INSTRUCTORS table.
5. The columns that should be indexed are those used regularly in the WHERE clause, the ORDER BY clause, the GROUP BY clause, and columns that are frequently joined. You should be careful not to index too many columns. Index only those columns that are regularly used in the aforementioned clauses of a SQL statement. Indexes slow down DML statements so you want to ensure that Oracle will use the index if it is created. Indexes help prevent full table scans, but you should be careful because full table scans are often quicker than index scans.
6. Indexes do not store NULLs. This is why when queries are written using WHERE VALUE1 IS NULL, Oracle cannot use the index because these null values are not stored. This results in a full table scan. It is generally a good idea to use NOT NULL constraints on indexed columns. If a column does not accept null values, then more queries can take advantage of the index because Oracle knows that all rows are going to be included. This query SELECT

COUNT(LASTNAME) FROM INSTRUCTORS could only use the index on LASTNAME if the column had a NOT NULL constraint. Otherwise, this query results in a full table scan.

7. The three main methods for reorganizing indexes are to drop and re-create them, rebuild them, or use the coalesce command.
8. The DBA_INDEXES data dictionary view contains one row for every index in the database.
9. The DBA_IND_COLUMNS data dictionary view contains one row for every column that has an index. If there is a concatenated index on LASTNAME and FIRSTNAME, DBA_IND_COLUMNS would have two rows — one for LASTNAME and one for FIRSTNAME.

Assessment Questions

1. **A.** 32 is the maximum number of columns in a concatenated index. For information see the “Logical Elements of Indexes” section.
2. **C.** Index entries store the Entry Header, the Key Column Length, the Key Column Value, and the ROWID. The ROWID is used to retrieve the actual row from the table. Refer to the “Types of Indexes in Oracle” section for more information.
3. **E.** PCTFREE specifies the amount of free space on an Index Leaf Block to reserve for future indexes at index creation time. For more information, refer to the section on “Creating Indexes.”
4. **E.** Reverse Key indexes store the key values in reverse order. Refer to section “Reverse Key Indexes” for more information.
5. **D.** Bitmap indexes are efficient at resolving queries with multiple predicates. Refer to section “Bitmap Indexes” for more information.
6. **A.** The NOSORT option requires that the data being indexed be in the correct order for the index to use. This means that the creation of the index does not require that a sort be performed. However, if the data is not in the correct order, the creation will fail. For more information, refer to section “Creating Indexes.”
7. **D.** Rebuilding an index ONLINE allows DML operations to continue against the table. Queries are permitted against the table whether the index is being rebuilt ONLINE or not. For more information, refer to section “Rebuilding Indexes.”
8. **E.** For more information, refer to section “Rebuilding Indexes.”
9. **A.** Refer to the section on “Getting Information on Indexes” for more information.
10. **E.** Refer to the section on “Getting Information on Indexes” for more information.

Scenarios

1. **A.** Since the CLASSID column is being populated by a sequence and most of the queries using the index use an equality predicate, the index should be changed to a Reverse Key index. The index should also be relocated onto a different tablespace from the table to reduce IO contention.
1. **B.** When rebuilding the index, use the ONLINE option. This enables users to continue processing DML statements while the index is being rebuilt.
2. **A.** The first thing you should do is ensure that the user performing the index rebuild is using a true temporary tablespace and that this tablespace resides on different disks from the index tablespaces. Also, ensure that the tablespace(s) where the indexes are being stored do not contain any rollback segments, temporary segments, or the tables that the indexes are indexing. The NOLOGGING option should be used to eliminate the index creation statement from being recorded in the redo log files. This should dramatically improve the creation performance. Finally, you should ensure that the user performing the index creation has a large amount of sort memory specified by the INIT.ORA parameter SORT_AREA_SIZE. Consider setting the value anywhere from 20MB to 50MB, depending upon available memory resources. The combination of all these should drastically reduce the amount of time the indexes take to create. If this does not resolve the problem, consider adding more disk space so that the indexes can be rebuilt rather than dropped and re-created.

Lab Exercises

Lab 12-1

2.

```
SQL> col index_name for a20
SQL> col table_name for a20
SQL> col column_name for a15
SQL> col index_type for a15
SQL> SELECT DI.INDEX_NAME
2      ,      DI.TABLE_NAME
3      ,      DIC.COLUMN_NAME
4      ,      DI.INDEX_TYPE
5 FROM      USER_INDEXES DI
6      ,      USER_IND_COLUMNS DIC
7 WHERE     DI.TABLE_NAME = DIC.TABLE_NAME
8 AND      DI.INDEX_NAME = DIC.INDEX_NAME
9 AND      DI.TABLE_NAME = 'CLASSENROLLMENT'
10 /
```

INDEX_NAME	TABLE_NAME	COLUMN_NAME	INDEX_TYPE
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	CLASSID	NORMAL
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	STUDENTNUMBER	NORMAL

```
SQL> col index_name clear
SQL> col table_name clear
SQL> col column_name clear
SQL> col index_type clear
```

3.

```
SQL> CREATE INDEX STUDENT.CLASSENROLL_ENROLLDATE ON
 2 CLASSENROLLMENT (ENROLLMENTDATE)
 3 TABLESPACE INDX
 4 PCTFREE 30
 5 STORAGE (INITIAL 100k
 6          NEXT 100k
 7          MINEXTENTS 4
 8          MAXEXTENTS UNLIMITED
 9          PCTINCREASE 0)
10 ;
```

Index created.

4.

```
SQL> col table_name for a20
SQL> col column_name for a15
SQL> col index_type for a15
SQL> SELECT DI.INDEX_NAME
 2 , DI.TABLE_NAME
 3 , DIC.COLUMN_NAME
 4 , DI.INDEX_TYPE
 5 FROM USER_INDEXES DI
 6 , USER_IND_COLUMNS DIC
 7 WHERE DI.TABLE_NAME = DIC.TABLE_NAME
 8 AND DI.INDEX_NAME = DIC.INDEX_NAME
 9 AND DI.TABLE_NAME = 'CLASSENROLLMENT'
10 /
```

INDEX_NAME	TABLE_NAME	COLUMN_NAME	INDEX_TYPE
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	CLASSID	NORMAL
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	STUDENTNUMBER	NORMAL
CLASSENROLL_ENROLLDATE	CLASSENROLLMENT	ENROLLMENTDATE	NORMAL

```
SQL> col index_name clear
SQL> col table_name clear
SQL> col column_name clear
SQL> col index_type clear
```

5.

```
SQL> CREATE BITMAP INDEX CLASSENROLL_STATUS_BM
2   ON CLASSENROLLMENT (STATUS)
3   TABLESPACE INDX
4   PCTFREE 30
5   STORAGE ( INITIAL 100k
6             NEXT      100k
7             MINEXTENTS 4
8             MAXEXTENTS UNLIMITED
9             PCTINCREASE 0);
```

Index created.

6.

```
SQL> col table_name for a20
SQL> col column_name for a15
SQL> col index_type for a15
SQL> SELECT DI.INDEX_NAME
2   ,      DI.TABLE_NAME
3   ,      DIC.COLUMN_NAME
4   ,      DI.INDEX_TYPE
5 FROM   USER_INDEXES DI
6   ,    USER_IND_COLUMNS DIC
7 WHERE  DI.TABLE_NAME = DIC.TABLE_NAME
8 AND    DI.INDEX_NAME = DIC.INDEX_NAME
9 AND    DI.TABLE_NAME = 'CLASSENROLLMENT'
10 /
```

INDEX_NAME	TABLE_NAME	COLUMN_NAME	INDEX_TYPE
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	CLASSID	NORMAL
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	STUDENTNUMBER	NORMAL
CLASSENROLL_ENROLLDATE	CLASSENROLLMENT	ENROLLMENTDATE	NORMAL
CLASSENROLL_STATUS_BM	CLASSENROLLMENT	STATUS	BITMAP

```
SQL> col index_name clear
SQL> col table_name clear
SQL> col column_name clear
SQL> col index_type clear
```

7.

```
CREATE INDEX INSTRUCTOR_LNAME_UPPER ON
INSTRUCTORS(UPPER(LASTNAME));
```

8. The index creation fails because you need to be a privileged user to create function based indexes.

10.

```
CREATE INDEX STUDENT.INSTRUCTOR_LNAME_UPPER ON
STUDENT.INSTRUCTORS(UPPER(LASTNAME));
```

Lab 12-2

2.

```
SQL> col index_name for a30
SQL> col table_name for a20
SQL> col tablespace_name for a20
SQL> SELECT      OUTER.INDEX_NAME
 2 ,            OUTER.TABLE_NAME
 3 ,            OUTER.TABLESPACE_NAME
 4 FROM          USER_INDEXES OUTER
 5 WHERE         EXISTS (SELECT 'A'
 6                FROM   USER_TABLES INNER
 7                WHERE  INNER.TABLE_NAME = OUTER.TABLE_NAME
 8                AND   INNER.TABLESPACE_NAME = OUTER.TABLESPACE_NAME)
 9 /
```

INDEX_NAME	TABLE_NAME	TABLESPACE_NAME
BATCHJOBS_JOBID_PK	BATCHJOBS	CERTDB
COURSEAUDIT_PK	COURSEAUDIT	CERTDB
PK_CLASSID	SCHEDULEDCLASSES	CERTDB
PK_CLASSID_STUDENTNUMBER	CLASSENROLLMENT	CERTDB
PK_COURSENUMBER	COURSES	CERTDB
PK_INSTRUCTORID	INSTRUCTORS	CERTDB
PK_LOCATIONID	LOCATIONS	CERTDB
PK_STUDENTNUMBER	STUDENTS	CERTDB
SYS_C001395	ORDERS	CERTDB
SYS_C001410	CUSTOMER	CERTDB
SYS_C001411	SUPPLIERS	CERTDB

INDEX_NAME	TABLE_NAME	TABLESPACE_NAME
SYS_IL0000020697C00003\$\$	ORD	CERTDB

12 rows selected.

```
SQL> col index_name clear
SQL> col table_name clear
SQL> col tablespace_name clear
```

3.

```
SQL> ALTER INDEX PK_CLASSID REBUILD TABLESPACE INDX;
```

Index altered.

```
SQL> ALTER INDEX PK_CLASSID_STUDENTNUMBER REBUILD TABLESPACE
INDX;
```

Index altered.

4.

```
SQL> ALTER INDEX PK_INSTRUCTORID REBUILD REVERSE TABLESPACE
INDX;
```

Index altered.

5.

```
SQL> ALTER INDEX PK_COURSENUMBER REBUILD ONLINE
2 TABLESPACE INDX;
```

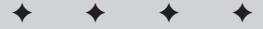
Index altered.



Maintaining Data Integrity

EXAM OBJECTIVES

- ◆ Maintaining Data Integrity
 - Implement data integrity constraints
 - Maintain integrity constraints
 - Obtain constraint information from the data dictionary



CHAPTER PRE-TEST

1. Name the five types of Oracle database constraints?
2. What is a PRIMARY KEY?
3. What is the difference between a primary key constraint and unique constraint?
4. What is a DEFERRABLE constraint?
5. What are the benefits to having constraints disabled?
6. What are the two main data dictionary views that contain constraint information?
7. What is the difference between an in-line constraint and an out-of-line constraint?
8. What is a database trigger?
9. What is the EXCEPTIONS table used for in regard to constraints?

The focus of this chapter is on the implementation and maintenance of database constraints. While other constraint options, such as database triggers and application constraints, are discussed, it is critical for all DBAs to have a strong understanding of how database constraints work inside of Oracle. The first major section of this chapter focuses on the five database constraints highlighting both their benefits and costs. We then discuss the various constraint states and how the state of a constraint impacts the database. This is followed by something a little different, deferrable constraints. Deferrable constraints allow the DBA to change when a constraint will be checked, and you need to know how Oracle has had to change to facilitate this feature. The last section of this chapter focuses on maintaining constraints, which covers adding, modifying, enabling, and disabling constraints. This section also outlines the data dictionary views where constraint information can be found.

Deciding which constraint to use is generally one of the easier tasks for a DBA. Most of these decisions are made during the design stage of database development, since constraints form an integral component of all relational database designs. The tricky part for DBAs, as is typical, is understanding what is happening in the background. The ability to disable a constraint can improve the performance of a bulk load in magnitudes of hundreds and sometimes thousands, allowing batch processes to complete in their allotted time. The downside of disabling constraints is that the DBA must always remember to enable them. If constraints are left disabled, you run the risk of corrupting the database. If a constraint is disabled, it is no longer enforced by Oracle and this can have damaging effects on the validity of the data stored within. Mastering constraints can and will make your job as a DBA much simpler and less troublesome.

Overview of Data Integrity in Oracle

An integrity constraint is a way of enforcing a rule. There are several different ways of maintaining data integrity inside an Oracle database, and deciding upon which method depends almost entirely on the situation. Application code, database triggers, and database constraints are the three available options for maintaining data integrity. The decision about which constraint method to use to enforce these rules is the responsibility of the database and application designers. One of the jobs of a DBA is to maintain these decisions. Constraints act as police. They enforce the rules that the decision makers have implemented. If a user violates one of the rules, the constraint reports back to the user who is performing an illegal action. In general, any of the three constraint methods can be used and all work. The following example illustrates that fact.

Suppose there is a rule that department numbers must be between 10 and 199. No other values are allowed. Implementing this rule through application code would probably be the easiest way to accomplish this by simply adding a check on the department number field validating the range through the front-end application. When a user types in a value, the check is performed prior to submitting the request to the database to be processed. You would need to add this check or validation to all applications that reference a department number.

If you decided to enforce this rule using triggers, you could add a BEFORE INSERT or UPDATE trigger to all tables storing the department number. This would check the value specified ensuring that the number did not violate the valid range. If it did, the trigger could deny the update or insert.

To enforce this rule using database constraints, you would add a check constraint to the department number column in the department table and a foreign key constraint on all other tables that use the department number. The check constraint in the department table ensures that the valid range of department numbers is between 10 and 199 and the foreign key constraint in all tables that use the department number ensures that only numbers that already exist in the department table can be used. Since it is validated by the check constraint in the department table, it will also be validated in all the tables that use the department number.

From these examples, you can see that any of the three methods could be used, but is one better than the others? Well, let's look at what happens when the rule changes. What would happen if the department range changed to 10 to 999? With the application code method, all applications that referenced the department number would need to be changed, recompiled, and redistributed to all users. It may be several applications that need to be changed. The scope of the change depends on the number of places in the application where this rule is enforced. This is often difficult to estimate and changing application code is not easy, especially if you do not have it! What would happen if you had purchased the software from a third party, Data Base Gurus Ltd? You would need to make a change request from Data Base Gurus Ltd., which is going to cost time and money, then wait for them to deliver. They may even decide not to make the change if it is too difficult or if it conflicts with other components of the application. Chances are that they have several customers using this software and must also consider those customers' needs whenever making any change to the applications. While this situation does not state that application code cannot or should not be used, the costs of changing a rule can be expensive.

If using database triggers, you would simply need to change all triggers and recompile. This would be straightforward — provided you had a developer available to make the change. Remember also that there may be more than one trigger that needs to be modified.

When using the database constraint method, the DBA just needs to change the check constraint. Change the upper bound of the valid range of departments to 999 in the department table only. This is done in one command while the database is still up and running. This definitely appears to be the easiest and cheapest way to implement changes.

Database constraints are generally preferred by most DBAs because they are easy to implement and maintain. Application designers love them because it makes for much easier development. They can write fairly generic code that is much more flexible and let each business implement its own rules at the database level. Generally the only time changes need to be made to the application code is when new features are being added and not when rules are being changed. You will usually find a combination of the three implemented as not all rules can be enforced with database constraints. There are also performance considerations, such as network traffic. The fewer calls made to the database, the less network traffic. With application code, you can almost guarantee that the database is not going to reject the request.

It is common to see a combination of all three implemented because they all have their strengths and weaknesses.



Maintaining constraints through application code or triggers is not covered on the exam.

Triggers resolve some of the limitations of database constraints. Things like conditional updates or FOREIGN KEY constraints through database links are not possible with database constraints. But triggers offer a database much more than just data integrity; they can be a valuable auditing tool. Here is an example of a trigger for auditing changes to the SALARY column of the EMPLOYEE table.

```
CREATE TABLE AUD_SAL(C_DATE DATE, C_USER VARCHAR2(30),
                     BEFORE_SALARY NUMBER(10),
                     EMPNO NUMBER(10),
                     NEW_SALARY NUMBER(10));

CREATE OR REPLACE TRIGGER AUD_SAL_T
  BEFORE
  UPDATE ON EMPLOYEES
  FOR EACH ROW
  BEGIN
    INSERT INTO AUD_SAL VALUES
      (SYSDATE,USER,:OLD.EMPNO, :OLD.SALARY, :NEW.SALARY);
  END;
```

```
SQL> UPDATE EMPLOYEES
      2 SET     SALARY = 100
      3 WHERE  EMPNO  = 10;
```

1 row updated.

```
SQL> SELECT * FROM AUD_SAL;
```

C_DATE	C_USER	EMPNO	BEFORE_SALARY	NEW_SALARY
26-FEB-01	STUDENT	10	9	100

The Oracle database engine is very efficient at validating database constraints, much quicker than application code or triggers, so whenever possible, constraints should be used. Here are some of the additional benefits of database constraints:

- ♦ Improve performance because the database is specifically designed to perform these types of checks. Can also be temporarily disabled, improving performance of batch processing.
- ♦ Easy to declare and change. Require no programming knowledge to implement and can be changed without having to change application code, allowing application code to be generic. Changes in rules can be made while the database is still being used.
- ♦ Centralize the business rules in the database. Fully documented in the database, providing a one-stop shop for all business rule information.
- ♦ Integrated with many design tools for easy implementation. When the database design is completed, many of the tools used to perform the design will create the database tables as well as the integrity constraints.

Integrity Constraints in Oracle

This section focuses on the five database constraints available in Oracle. It discusses the various constraint states, how they are used by Oracle as well as highlighting special considerations for implementing these constraints.

Types of constraints

This section looks at the five different types of database constraints in Oracle explaining how they work as well as practical uses.

NOT NULL

Understanding the use of NOT NULL constraints requires an understanding of what a null is or isn't, and why it presents particular problems for Oracle. A null is defined as the absence of a value. It is not a zero or a null string such as ''. It is simply not there. By default, when creating tables, nulls are permitted. Therefore when data is being updated or inserted Oracle does not complain about nulls. The problem with nulls is that they cannot be used in mathematical calculations or be equal to anything. To Oracle, nulls in mathematical expressions cannot be resolved. It is not $1 + \text{NULL}$ it is actually $1 + (\text{something that is not there})$ and is an invalid mathematical expression.

Here is an example of why nulls are problematic to Oracle. Note that NULL is a reserved word in Oracle and is used to insert or update null values.

```
SQL> DESC EMP
Name                               Null?    Type
-----
EMP_ID                             NUMBER(3)
NAME                                VARCHAR2(40)
SAL                                  NUMBER(8,2)
JOB                                  VARCHAR2(15)
```

You can see that in this definition of the EMP table that there are four columns. Now let's insert some data.

```
SQL> INSERT INTO EMP VALUES (123,'TODD ROSS',1000,
'INSTRUCTOR');1 row created.
SQL> INSERT INTO EMP VALUES (456,'ETIENNE KERR',10000,
'OWNER');
1 row created.
SQL> INSERT INTO EMP VALUES (789,'ERIN LEE',10, 'SALES');
1 row created.
SQL> INSERT INTO EMP VALUES (124,'TIM MABEY',NULL, 'JANITOR');
1 row created.
SQL> INSERT INTO EMP VALUES (458,'MIA HEMPEY',2000,
'PRESIDENT');
1 row created.
SQL> INSERT INTO EMP VALUES (780,'ROBERT BEDFORD',50000,
'SALES');
1 row created.
SQL> INSERT INTO EMP VALUES (125,'MIKE HUNT',NULL, 'SALES');
1 row created.
```

Note that for employees TIM MABEY and MIKE HUNT a NULL was inserted for their salaries. When the data is selected, the null appears as an empty column.

```
SQL> SELECT * FROM EMP;
```

EMP_ID	NAME	SAL	JOB
123	TODD ROSS	1000	INSTRUCTOR
456	ETIENNE KERR	10000	OWNER
789	ERIN LEE	10	SALES
124	TIM MABEY		JANITOR
458	MIA HEMPEY	2000	PRESIDENT
780	ROBERT BEDFORD	50000	SALES
125	MIKE HUNT		SALES

```
7 rows selected.
```

The problem that Oracle has with a NULL is that it cannot perform certain operations. Suppose you wanted to find the average salary between two employees, TODD ROSS and TIM MABEY. You could resolve this quickly in your head and it would be 500. Since TIM MABEY does not have a salary specified the equation is $1000 / 2 = 500$. However, when you ask Oracle to perform the same calculation it yields different results.

```
SQL> SELECT AVG(SAL) FROM EMP WHERE NAME IN ('TODD ROSS','TIM MABEY');
```

```
AVG(SAL)
-----
      1000
```

What is going on? Well the answer is simple. When performing these types of calculations, Oracle only considers columns that have values because it does not know about the ones that don't. Since the SAL column for TIM MABEY is NULL, it is not part of the calculation. When the salary is changed to be 0 instead of a NULL, we can see that the calculation works as expected.

```
UPDATE EMP SET SAL=0 WHERE SAL IS NULL;
```

```
SQL> SELECT AVG(SAL) FROM EMP WHERE NAME IN ('TODD ROSS','TIM MABEY');
```

```
AVG(SAL)
-----
       500
```

That's a little better!

There are other issues with columns that accept nulls. Nulls are not stored inside of indexes. Therefore, certain types of operations are forced to use more expensive full table scans than index scans because Oracle cannot guarantee that all the data will exist on the index. If the column has a NOT NULL constraint, then Oracle uses the indexes for many more operations and this generally results in better response time and throughput for the database.

If a column has a NOT NULL constraint, Oracle rejects all attempts to enter NULL values.

```
SQL> ALTER TABLE EMP MODIFY SAL NOT NULL;
```

```
SQL> UPDATE EMP SET SAL=NULL WHERE NAME='TIM MABEY';
```

```
UPDATE EMP SET SAL=NULL WHERE NAME='TIM MABEY'
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01407: cannot update ("SYS"."EMP"."SAL") to NULL
```

It is common to have default values for columns defined with a NOT NULL, such as 0 for numeric fields and 'N/A' for character ones.

UNIQUE

UNIQUE constraints designate a column or columns as being unique within the table or NULL. This means that no two rows can share the same value for the column or columns designated with the UNIQUE constraint. This constraint is used to enforce business rules like "No two people can have the same social security

number.” Maybe a payroll application uses the social security numbers for reporting tax information to the government. If by accident the social security numbers for two employees are the same, the information would be incorrectly reported.

Oracle maintains UNIQUE constraints with an index, which means that Oracle can validate the constraint quickly. If an index does not already exist on the UNIQUE constraint column(s), Oracle automatically creates one and gives it the same name as the constraint. Without the index, Oracle would need to perform a full table scan every time a user inserted a new row. In order to ensure uniqueness in the table, Oracle must ensure that the same value does not already exist. If the table contained thousands of rows, the validation of a UNIQUE constraint would get slower and slower with every new row inserted as Oracle performed the scan of the entire table. By using the index, whenever a new row is inserted, Oracle simply scans the index to ensure that there are no duplicates. With a large number of rows Oracle can always perform index scans for duplicates faster than full table scans.

The fact that Oracle uses indexes for maintaining UNIQUE constraints imposes two problems. The first is that the DBA must manage the index. The second is that indexes do not store nulls. Therefore, you could have multiple rows with null values for the columns designated with the UNIQUE constraint and this would not violate the constraint. To resolve this problem, you could simply combine a NOT NULL constraint with a UNIQUE constraint or define a PRIMARY KEY constraint.

PRIMARY KEY

A PRIMARY KEY constraint is simply a combination of a UNIQUE and NOT NULL constraint. Virtually all good database designs include a column or collection of columns that are unique for each table. This enables users to pinpoint every row in a table. Let’s say that you wanted to update an employee’s salary. Without a unique way of doing this, you cannot guarantee that you are just updating one employee record. What happens when you update Joe Smith’s salary? You issue the following command:

```
UPDATE EMP
SET     SAL = 100
WHERE  NAME = 'JOE SMITH';
```

But what if there were two employees named Joe Smith? Both employees would now have the same salary of 100. The presence of a PRIMARY KEY or UNIQUE constraint resolves this problem. The presence of a PRIMARY KEY or UNIQUE constraint guarantees that only one row is updated. If there were a PRIMARY KEY on the EMPLOYEE_NO column and that column were used to perform the update, then only one row would be updated.

Since a PRIMARY KEY is just a combination of UNIQUE and NOT NULL, the same indexing issues exist as they do with UNIQUE constraints. If an index does not already exist on the UNIQUE constraint column(s), Oracle automatically creates one and gives it the same name as the constraint.

The designers are responsible for designating the PRIMARY KEY within a table and Oracle only permits one per table. However, you can have as many combinations of UNIQUE and NOT NULL constraints as needed.

There can be a bit of a dilemma when you choose which column(s) to designate as the PRIMARY KEY. Perhaps either the employee id column or the social security number column can be the PRIMARY KEY. In those situations, it is always best to designate the column that is going to be used most often in queries and DML statements as the PRIMARY KEY.

FOREIGN KEY

FOREIGN KEY constraints on a column or columns restrict the allowable values for the column(s) to value in referenced table. This is called *referential integrity*. FOREIGN KEY constraints are key elements of all relational databases as they enforce the relationship between two tables. If the employee table is related to the department table by the department number column using a FOREIGN KEY, then only department numbers that exist in the department table or a NULL are allowed in the employee table. This ensures that employees do not work for non-existent departments. Figure 13-1 demonstrates this relationship.

The relationship in this example is between the EMPLOYEE table and the DEPARTMENT table. The only valid values accepted in the EMPLOYEE table for the DEPT_NO column are either 10, 20, 30, or NULL. If you were to try to insert an employee record with DEPT_NO = 40, the insert would fail. Oracle checks the table referenced, in this case the DEPARTMENT table, for a value. If one is not found it rejects the statement. This also applies to updates. You could not move JOHN WILSON to department 40, since it does not exist in the DEPARTMENT table.

The gotcha is that this relationship works both ways. Once the FOREIGN KEY has been defined, it also puts rules on the DEPARTMENT table apart from the PRIMARY KEY rules. If a PRIMARY KEY or UNIQUE constraint is referenced by a FOREIGN KEY, then it must comply with the rules of the FOREIGN KEY constraint. What this means is that you cannot do anything to the DEPT_NO column in the DEPARTMENT table that would invalidate this relationship. An example of this would be changing department number 10 to 40 in the DEPARTMENT table. While this does not violate the PRIMARY KEY on this column, it does violate the FOREIGN KEY referencing this column. Oracle checks if there are any rows in the EMPLOYEE table where the DEPT_NO column is 10. If there are, then the update of the DEPARTMENT table would cause all those rows to be orphaned, invalidating the relationship. However, if there are no employees in department 10, then the constraint is not violated and Oracle allows it. Changes to the NAME column in the DEPARTMENT table are

permitted, since they do not form part of the relationship. Oracle also rejects deletes on the DEPARTMENT table if the row being deleted is referenced from the EMPLOYEE table. Attempting to delete department 30 from the DEPARTMENT table raises an exception. Inserting new rows in the DEPARTMENT table is allowed, since they cannot violate the constraint.

FOREIGN KEY constraints must reference a PRIMARY KEY or UNIQUE constraint. FOREIGN KEY constraints can reference other columns in the same table. An employee table can contain a column for storing manager ids that is just a FOREIGN KEY pointing to the employee table's employee id column. This is called a *self-referencing foreign key* and is most often used in hierarchical relationships such as employees and managers. FOREIGN KEY constraints accept null values, so it is not uncommon to see FOREIGN KEY constraints combined with NOT NULL constraints on the same column.

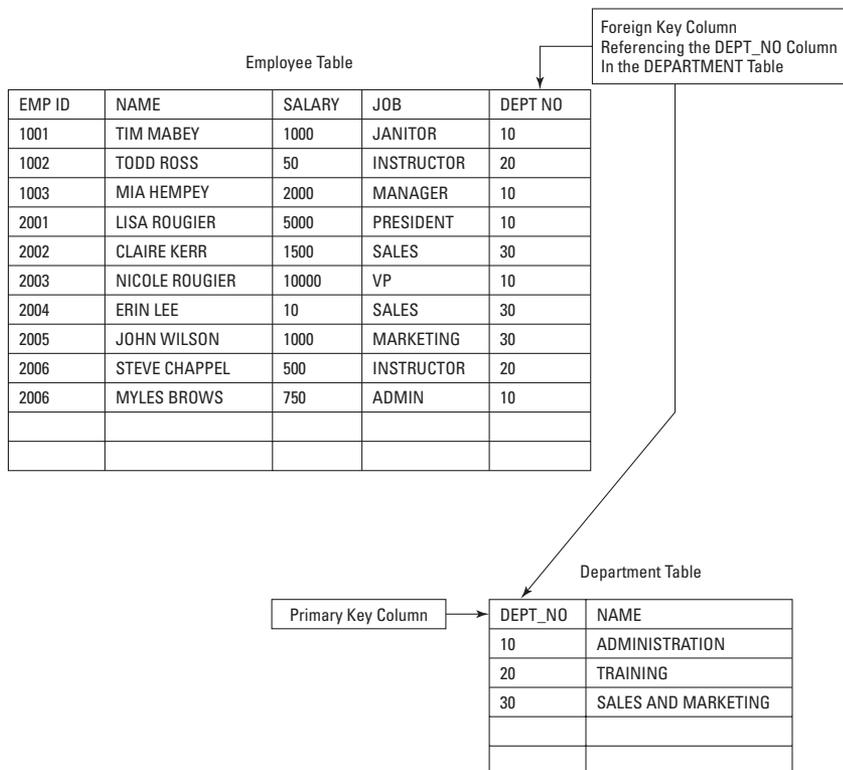


Figure 13-1: Using foreign key constraints

A special option that can be used with FOREIGN KEY constraints is ON DELETE CASCADE. Now when rows are deleted in the parent or referenced table, all rows in the child table with that key value will also be dropped. As you can imagine, this

is potentially dangerous and you must ensure that the two tables are mutually exclusive. Currently no ON UPDATE CASCADE or ON INSERT CASCADE constraints exist in Oracle. These types of constraints need to be enforced with triggers.



The FOREIGN KEY to PRIMARY KEY or UNIQUE key relationship is covered in more detail in the “Foreign Key Considerations” section.

CHECK

CHECK constraints are one of the best tools for implementing business rules in a database that are sometimes also enforced at the application level. A CHECK constraint allows a rule or condition to be placed on a column. A business rule might be something like “Valid department numbers are between 10 and 199,” and you can use a CHECK constraint to enforce this rule. Here is an example of how the check constraint is implemented.

```
CREATE TABLE DEPARTMENT (
  DEPTNO NUMBER(4) CONSTRAINT DEPT_DEPTNO_PK PRIMARY KEY,
  NAME    VARCHAR2(20),
  CONSTRAINT DEPT_DEPTNO_CK CHECK (DEPTNO BETWEEN 10 AND
199))
```

This code creates a table called DEPARTMENT with a PRIMARY KEY constraint on the DEPTNO column and a CHECK constraint on the DEPTNO column setting the range of valid DEPTNO values to be between 10 and 199. Two constraints are placed on the DEPTNO column and both will be enforced. Let’s insert some rows.

```
INSERT INTO DEPARTMENT VALUES (10,'ADMIN');
```

```
1 Row Created
```

```
INSERT INTO DEPARTMENT VALUES (20,'TRAINING');
```

```
1 Row Created
```

```
INSERT INTO DEPARTMENT VALUES (30,'SALES');
```

```
1 Row Created
```

Now if you try to insert department number 5, you receive the following error:

```
INSERT INTO DEPARTMENT VALUES (5,'MARKETING');
```

```
INSERT INTO DEPARTMENT VALUES (5,'MARKETING')
```

```
*
```

```
ERROR at line 1:
```

```
ORA-02290: check constraint (SYS.DEPT_DEPTNO_CK) violated
```

The results will be the same if you try to change the value through an update statement.

```
UPDATE DEPARTMENT
SET   DEPTNO = 200
WHERE DEPTNO = 10;
```

```
UPDATE DEPARTMENT
*
ERROR at line 1:
ORA-02290: check constraint (SYS.DEPT_DEPTNO_CK) violated
```

CHECK constraints are also a way of enforcing more complex business rules, such as conditional nulls. Take an example of a rental items table for a video store. The video store rents movies and video games. They track all movies and games rented. When renting a game, they need to know the type (such as Play Station or N64), but when renting a movie, this column is NULL. This is a conditional NULL. It is NOT NULL when games are rented. Here is an example of the CREATE TABLE command that would create this type of constraint.

```
CREATE TABLE RENTAL_ITEM
  (ID NUMBER(10) PRIMARY KEY,
   ITEM_NO NUMBER(10) REFERENCES ITEM(ID),
   RENTAL_TYPE VARCHAR2(2) NOT NULL,
   MEDIA VARCHAR2(10),
   GAME_TYPE VARCHAR2(10),
   CONSTRAINT RENTAL_ITEM_GAME_TYPE_CK CHECK
     ((RENTAL_TYPE='GA' AND GAME_TYPE IS NOT NULL)
      OR
      (RENTAL_TYPE<>'GA' AND GAME_TYPE IS NULL))
```

You can see that a check constraint has been added for the GAME_TYPE column. If the rental type equals game (RENTAL_TYPE='GA'), then the game type (GAME_TYPE) is NOT NULL. However, if the rental type is anything other than a game, the GAME_TYPE column must be NULL.

When you insert data, you can see that as long as the RENTAL_TYPE <> 'GA' the GAME_TYPE must be NULL, but as soon as the RENTAL_TYPE='GA', the GAME_TYPE column must have a value.

```
INSERT INTO RENTAL_ITEM VALUES (
  1,NULL,'MV','VHS',NULL);

1 row created

INSERT INTO RENTAL_ITEM VALUES (
  2,NULL,'GA','VHS','PL STATION')

1 row created

INSERT INTO RENTAL_ITEM VALUES (
  3,NULL,'GA','VHS',NULL)

INSERT INTO RENTAL_ITEM VALUES (
*
```

```
ERROR at line 1:
ORA-02290: check constraint (SYS.RENTAL_ITEM_GAME_TYPE_CK)
violated

INSERT INTO RENTAL_ITEM VALUES (
    4, NULL, 'MV', 'DVD', 'PL STATION')

INSERT INTO RENTAL_ITEM VALUES (
*
ERROR at line 1:
ORA-02290: check constraint (SYS.RENTAL_ITEM_GAME_TYPE_CK)
violated
```

A check constraint is a truly powerful tool for implementing business rules. It can save large amounts of money when business rules change. Business rules can be changed easily inside the database instead of in the application code.

There are some limitations to check constraints, however, the biggest one being that the constraint cannot reference other rows. It is applied to the current row being inserted or modified.



The constraint syntax is covered in the section “Implementing Constraints.”

Constraint states

One of the nice features about database constraints is that they can be disabled. Turning off a constraint or disabling it changes the constraint state. When disabled the constraint is no longer acting as a rule. Turning a constraint on or enabling it changes the constraint state as well. Enabling a constraint means that the rule is going to be enforced. They can be turned off and then turned back on again, without needing to re-create them. When processing a large batch job, it is often much quicker to disable constraint checking during the job and then enable it when done. The time it takes to complete the batch process or bulk load plus the time it takes to enable a constraint is almost always substantially faster than leaving the constraints enabled during the same process. This flexibility is a powerful tool for DBAs, often enabling them to process large jobs within an allotted window of time.

When constraints are enabled, Oracle must ensure that no rows violate that constraint. When enabling a constraint on a table with millions of rows, you can expect this to take some time. Oracle permits constraints to be enabled, which means all future DML statements will be checked against the constraint, however, existing data is not checked. The constraint will be enabled instantly, allowing a user to start processing against the table again. There are certain situations and scenarios where this is not advisable but it can be a big time saver for DBAs. Constraints can either be `DISABLED` or `ENABLED` and the existing data in the tables where the constraint is being `DISABLED` or `ENABLED` can either be `VALIDATED` or `NOT VALIDATED`. This probably sounds confusing, so let's break down all the combinations and see when one state is preferred over another.

DISABLED NOVALIDATE

When a constraint is `DISABLED NOVALIDATE`, you are turning off the constraint. The existing data is not validated and not protected from future changes. This is the default behavior when a constraint is disabled. The constraint definition remains in the data dictionary, but future DML statements can violate the constraint.

Assume there is a `UNIQUE` constraint on the `name` column in the `VENDOR` table, preventing two rows in the table from having the same name.

```
CREATE TABLE VENDOR (ID number(10)
                     CONSTRAINT VEND_ID_PK PRIMARY KEY,
                     NAME varchar2(50)
                     CONSTRAINT VEND_NAME_UN UNIQUE,
                     ADDRESS1 varchar2(30),
                     ADDRESS2 varchar2(30),
                     PHONE varchar2(15),
                     FAX   varchar2(15));
```

Initially there is a `PRIMARY KEY` constraint on the `ID` column and a `UNIQUE` constraint on the `NAME` column. These will be enforced when data is inserted.

```
INSERT INTO VENDOR VALUES (      1
,                               'PETES POTATOES'
,                               '123 Main St.'
,                               NULL
,                               '555-555-1234'
,                               '555-555-4321')
```

1 row created.

```
INSERT INTO VENDOR VALUES (      2
,                               'ABC PLUMBING'
,                               '123 West ST'
,                               NULL
,                               '555-555-5678'
,                               '555-555-8765')
```

1 row created.

```
INSERT INTO VENDOR VALUES (      3
,                               'ABC PLUMBING'
,                               '123 North ST'
,                               NULL
,                               '444-333-5678'
,                               '444-333-8765')
```

```
INSERT INTO VENDOR VALUES (      3
*
ERROR at line 1:
ORA-00001: unique constraint (SYS.VEND_NAME_UN) violated
```

This is what we expected. The name ABC PLUMBING violates the UNIQUE constraint on the NAME column. Now, let's DISABLE NOVALIDATE the UNIQUE constraint.

```
ALTER TABLE VENDOR DISABLE NOVALIDATE CONSTRAINT VEND_NAME_UN;
```

Table altered.

Now let's try to insert the row that failed previously.

```
INSERT INTO VENDOR VALUES (      3
,                                'ABC PLUMBING'
,                                '123 North ST'
,                                NULL
,                                '444-333-5678'
,                                '444-333-8765')
```

1 row created.

This is again what we expected to happen, but what about updating an existing row like PETES POTATOES and set the NAME column to ABC PLUMBING as well? We would expect this to work as well.

```
UPDATE VENDOR
SET     NAME = 'ABC PLUMBING'
WHERE  NAME = 'PETES POTATOES';
```

1 row updated.

And we can see, there was no problem with violating this rule as the constraint is disabled. One of the rows where the NAME column is ABC PLUMBING will need to be changed before this constraint can be enabled to prevent the following error:

```
SQL> ALTER TABLE VENDOR ENABLE CONSTRAINT VEND_NAME_UN;
ALTER TABLE VENDOR ENABLE CONSTRAINT VEND_NAME_UN
*
ERROR at line 1:
ORA-02299: cannot validate (SYS.VEND_NAME_UN) - duplicate keys
found
```

DISABLE VALIDATE

When a constraint state is DISABLE VALIDATE the table is effectively rendered READ ONLY. No DML operations are permitted on the table with a constraint state of DISABLE VALIDATE. Using the same table, VENDOR, with the two rows, one for PETES POTATOES and the other for ABC PLUMBING, you can see that when the constraint state is changed to DISABLE VALIDATE, only queries are allowed.

```
ALTER TABLE VENDOR DISABLE VALIDATE CONSTRAINT VEND_NAME_UN;
```

Table altered.

```
INSERT INTO VENDOR VALUES (      3
```


In this example, there are initially two rows in the VENDOR table. First we disable the UNIQUE constraint and then add a new row that would violate the constraint. We will then change the constraint to a state of ENABLE NOVALIDATE and see what happens..

```
SQL> ALTER TABLE VENDOR DISABLE CONSTRAINT VEND_NAME_UN;
```

Table altered.

```
SQL> INSERT INTO VENDOR VALUES (          3
2 , 'ABC PLUMBING'
3 , '123 North ST'
4 , NULL
5 , '444-333-5678'
6 , '444-333-8765')
7 /
```

1 row created.

```
SQL> SELECT NAME FROM VENDOR;
```

```
NAME
-----
PETES POTATOES
ABC PLUMBING
ABC PLUMBING
```

```
SQL> ALTER TABLE VENDOR ENABLE NOVALIDATE CONSTRAINT
VEND_NAME_UN;
```

You can see that there are two vendors with the name ABC PLUMBING and the constraint is enabled. All future DML statements must not violate the constraint. Here we are attempting to insert another vendor with the name PETES POTATOES, and you see that it is rejected.

```
SQL> INSERT INTO VENDOR VALUES (          4
2 , 'PETES POTATOES'
3 , '123 North ST'
4 , NULL
5 , '444-333-7777'
6 Input truncated to 42 characters
, '444-333-8888')
7 /
INSERT INTO VENDOR VALUES (          4
*
ERROR at line 1:
ORA-00001: unique constraint (SYS.VEND_NAME_UN) violated
```

ENABLE VALIDATE

This is the default behavior for constraints when they are enabled. All current and future rows will be checked to ensure that they do not violate the constraint. Note

that while the current data is being validated, the table is locked preventing DML against it. DBAs perform this task on ENABLE NOVALIDATE constraints during less busy hours when locking the table is tolerable to the users.

When are constraints checked?

Constraints can either set to be checked after each DML statement or when commits are issued. Constraints that are checked upon commit are called *deferrable*. Constraints checked after each DML statements are called *nondeferrable*. Nondeferrable constraints are the default for Oracle. Constraints defined as nondeferrable cannot ever be deferred.

In order to use deferrable constraints, the constraint must be created with one of two options. The INITIALLY IMMEDIATE option for the constraint means that the constraint checking will be done after every DML statement. However, the user can issue the ALTER SESSION SET CONSTRAINT[S] = DEFERRED command to set all constraints created INITIALLY IMMEDIATE as deferrable, meaning that the constraints will be checked upon commit instead of after each DML statement.

The other option for using deferrable constraints is to create the constraint using the INITIALLY DEFERRED option. This means that the constraint will not be checked until a commit is issued by the transaction. The ALTER SESSION SET CONSTRAINT[S] = DEFERRED command does not need to be issued.

If a constraint is INITIALLY DEFERRED, the user can change this behavior by issuing the ALTER SESSION SET CONSTRAINT[S] = IMMEDIATE. This will force constraint checking to be done after each DML statement for INITIALLY DEFERRED constraints.



Tip

Constraints cannot be modified to be DEFERRABLE. They must be created as such.

In the following example, there are two tables, ORDERS and ITEMS. The ITEMS table has a FOREIGN KEY constraint on the ORD_NO column referencing the ORD_NO column in the ORDERS table. The FOREIGN KEY constraint is set to INITIALLY IMMEDIATE to set constraint checking to be performed after each SQL STATEMENT.

```
CREATE TABLE ORDERS (  
    ORD_NO NUMBER(10) CONSTRAINT ORDERS_ORD_NO_PK PRIMARY KEY  
,  
    VENDOR_ID NUMBER(10) NOT NULL  
,  
    ORD_DATE DATE  
)  
  
CREATE TABLE ITEMS (  
    ITEM_ID NUMBER(10)  
,  
    ORD_NO NUMBER(10)  
    CONSTRAINT ITEMS_ORD_NO_FK REFERENCES ORDERS(ORD_NO)
```

```

,           DEFERRABLE INITIALLY IMMEDIATE
,           PROD_ID NUMBER(5) NOT NULL
,           PRICE NUMBER(8,2) NOT NULL
,           QTY NUMBER(5) DEFAULT 1 NOT NULL)

```

By setting the FOREIGN KEY constraint to DEFERRABLE INITIALLY IMMEDIATE, the user has the option of setting the constraint checking to be deferred. In this scenario, as data is entered, without changing the session, constraint checking is performed after each DML statement.

```

SQL> INSERT INTO ORDERS VALUES (1,1001,SYSDATE);

1 row created.

SQL> INSERT INTO ITEMS VALUES (1,1,567,10,5);

1 row created.

SQL> INSERT INTO ITEMS VALUES (2,1,765,2,20);

1 row created.

```

Now try to insert into the ITEMS table first and then the ORDERS table—a common thing to want to do. Assuming you have a Web site that is taking orders, users enter items they are purchasing in the ITEMS table and, when they are done, enter the ORDER information.

```

SQL> INSERT INTO ITEMS VALUES (1,2,999,1,500);
INSERT INTO ITEMS VALUES (1,2,999,1,500)
*
ERROR at line 1:
ORA-02291: integrity constraint (SYS.ITEMS_ORD_NO_FK) violated
- parent key not
Found

```

Again, this is what we expected, but notice what happens when we set the constraint checking to be deferred.

```

SQL> ALTER SESSION SET CONSTRAINTS = DEFERRED;

Session altered.

SQL> INSERT INTO ITEMS VALUES (1,2,999,1,500);

1 row created.

SQL> INSERT INTO ITEMS VALUES (2,2,444,40,2);

1 row created.

SQL> COMMIT;
COMMIT

```

```
*
ERROR at line 1:
ORA-02091: transaction rolled back
ORA-02291: integrity constraint (SYS.ITEMS_ORD_NO_FK) violated
- parent key not
Found
```

This time the rows were initially allowed, but were rejected when the user issued the COMMIT statement, because order 2 did not exist in the ORDER table.

```
SQL> INSERT INTO ITEMS VALUES (1,2,999,1,500);

1 row created.

SQL> INSERT INTO ITEMS VALUES (2,2,444,40,2);

1 row created.

SQL> INSERT INTO ORDERS VALUES (2,444,SYSDATE);

1 row created.

SQL> COMMIT;

Commit complete.
```

And you can see, this time when the user issued the COMMIT, the rows were accepted because order number 2 was inserted into the ORDERS table. When the constraints are checked, the FOREIGN KEY constraint in the ITEMS table is no longer violated.

The syntax for setting deferrable constraints at the session level is as follows:

```
ALTER SESSION SET CONSTRAINT[S] =
  {IMMEDIATE | DEFERRED | DEFAULT}

or

SET CONSTRAINT[S]
  {constraint [, constraint ] ... |ALL }
  {IMMEDIATE | DEFERRED}
```

Special considerations for constraints

There are several special considerations for constraints that DBAs should be aware of when creating and maintaining database constraints.

Using NON-UNIQUE indexes on PRIMARY KEY and UNIQUE constraints

When using deferrable constraints, Oracle has to consider the fact that duplicates can now exist in tables with PRIMARY KEY or UNIQUE constraints — at least until

the user issues a commit. Now, since Oracle typically maintains PRIMARY KEY and UNIQUE constraints with UNIQUE indexes, duplicates would violate the index. To fix this problem, Oracle must use a NON UNIQUE index for validating deferrable PRIMARY KEY and UNIQUE constraints. The biggest advantage with this option is when constraints are DISABLED, Oracle will not drop the index. This means that the index will be maintained by Oracle, can be used by queries, and re-enabling the constraint will be much quicker because Oracle does not need to create the index.

Oracle can use existing indexes to enforce PRIMARY KEY and UNIQUE constraints. Indexes occupy space and are costly to maintain. When you think about it, every time you insert a new row into a table, Oracle must update all the indexes against that table. This can be very expensive and slow down DML statements. Therefore, if the PRIMARY KEY for the ORDERS table is designated to be the ORD_NO column, Oracle creates a UNIQUE index for that column. But what if most queries use the ORD_NO and VENDOR column? To speed these queries up, you could create a composite index on the ORD_NO,VENDOR columns. By doing this, every query that just wants the order number and vendor number columns can use just the index. The downside is that you now have two indexes to maintain, which can be expensive and slow. The solution here would be to create an index on the ORD_NO,VENDOR column before creating the PRIMARY KEY constraint. This way when the constraint is created, Oracle sees that an index already exists and uses it instead of creating another one.


Tip

In order to use an existing index, the first column of the constraint must be the first column of the index. If the index in our example were on VENDOR,ORD_NO, it could not be used and Oracle would create another index just on ORD_NO.

In the following example, you can see that the table is initially created, then a NON UNIQUE index is created on the ID,NAME columns, and finally the PRIMARY KEY is added. When the DBA_INDEXES data dictionary view is queried, no additional indexes were required to support the PRIMARY KEY. Oracle used the existing NON-UNIQUE index that already existed. Pretty neat!

```
SQL> CREATE TABLE T1 (ID NUMBER(10)
2      ,                NAME VARCHAR2(40)
3      ,                T_DATE DATE)
4      /
```

Table created.

```
SQL> CREATE INDEX T1_ID_NAME_IND ON T1(ID,NAME);
```

Index created.

```
SQL> SELECT INDEX_NAME,INDEX_TYPE
        FROM   DBA_INDEXES
        WHERE  TABLE_NAME='T1';
```

INDEX_NAME	INDEX_TYPE
------------	------------

```
-----
T1_ID_NAME_IND                NORMAL
SQL> ALTER TABLE T1 ADD PRIMARY KEY (ID);
```

Table altered.

```
SQL> SELECT INDEX_NAME, INDEX_TYPE
       FROM   DBA_INDEXES
       WHERE  TABLE_NAME='T1';
```

```
INDEX_NAME                    INDEX_TYPE
-----
T1_ID_NAME_IND                NORMAL
```

Tips for FOREIGN KEY constraints

FOREIGN KEY constraints restrict the types of activities that can be performed on the table they are referencing. You can take steps to ensure that the FOREIGN KEY constraints are optimally configured.

Dropping parent or referenced table

If the parent table were to be dropped, this would violate the FOREIGN KEY constraint. In Figure 13-1 the EMPLOYEES table has a FOREIGN KEY referencing the parent table DEPARTMENTS. Dropping the DEPARTMENTS table would leave orphaned child records in the EMPLOYEES table.

```
SQL> DROP TABLE DEPARTMENTS;
DROP TABLE DEPARTMENTS
      *
ERROR at line 1:
ORA-02449: unique/primary keys in table referenced by foreign
keys
```

If maintenance operations are being performed on the DEPARTMENTS table, dropping it may be a temporary action. When it is re-created, the constraint will no longer be invalid. However, as you can see from the previous example, Oracle will not let you do this raising error ORA-02449. To get around this problem, you have two options.

Your first instinct might be to just disable the FOREIGN KEY constraint on the ITEMS table and then drop the ORDERS table. However, Oracle will not allow this. You will see the same error as the previous example, even though the key has been disabled. So, you either have to drop the constraint on the ITEMS table first and then drop the ORDERS, or — the easier choice — use the CASCADE CONSTRAINTS option of the drop table command, which drops all FOREIGN KEY constraints referencing the table being dropped.

```
DROP TABLE ORDERS CASCADE CONSTRAINTS;
```

This command drops the `ORDERS` table and the `FOREIGN KEY` constraint in the `ITEMS` table. When the `ORDERS` table is re-created, the `FOREIGN KEY` constraint in the `ITEMS` table will also need to be recreated — that is, it will not be automatically re-created by Oracle.

Dropping tablespaces with parent tables

If you are dropping a tablespace and there are `FOREIGN KEY` constraints referencing tables in the tablespace being dropped, a special option of the `DROP TABLESPACE` command is required. You must include the `CASCADE CONSTRAINTS` option as follows:

```
DROP TABLESPACE TEST123 INCLUDING CONTENTS  
CASCADE CONSTRAINTS
```



This is a dangerous and sometimes unrecoverable command. The `INCLUDING CONTENTS` clause tells Oracle to remove all tables in the `TABLESPACE` as well.

Excessive locking with DML statements

One of the biggest problems with foreign keys is locking. When DML operations are performed on the parent table, Oracle must first check that it does not violate any `FOREIGN KEY` constraints in a child table. To do this, it must scan the child table. If there is no index on the `FOREIGN KEY` column, then Oracle is forced to perform an expensive full table scan and it must also `LOCK` the child table to ensure no destructive actions can take place. If, however, the `FOREIGN KEY` columns are indexed, then Oracle can simply scan the index and not be required to lock the table. If the child table is large — thousands or perhaps millions of rows — and a full table scan is required, it will take some time to complete. Any users trying to access the child table must wait. Figure 13-2 provides an illustration.

In Figure 13-3, the `UPDATE` on the `ORDERS` table will be disallowed as there is a `FOREIGN KEY` on the `ITEMS` table. But, in order to reject this, Oracle must first check the `ITEMS` table. Without the presence of an index on the `ORD_NO` column in the `ITEMS` table, Oracle will lock the table and perform full table scan. This could take a long time and prevent users from adding new entries to the table.

When an index is present on the `FOREIGN KEY` column, Oracle scans the index instead of the table. It does not need to lock the table. This index look up is much quicker than the full table scan.

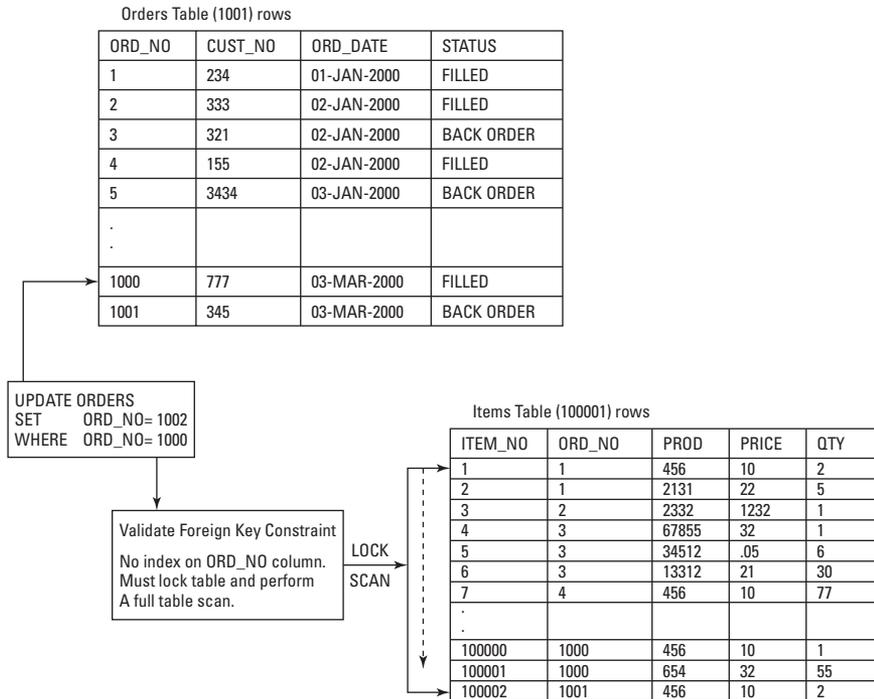


Figure 13-2: FOREIGN KEY locks on child tables



Tip

Always index columns with FOREIGN KEY constraints. Not only does it prevent excessive locking, the index also boosts the performance of join operations between the two tables. Since the FOREIGN KEY column is the link between two tables, it makes sense that this is how the tables will be joined in SQL statements. The presence of an index speeds up these join operations.

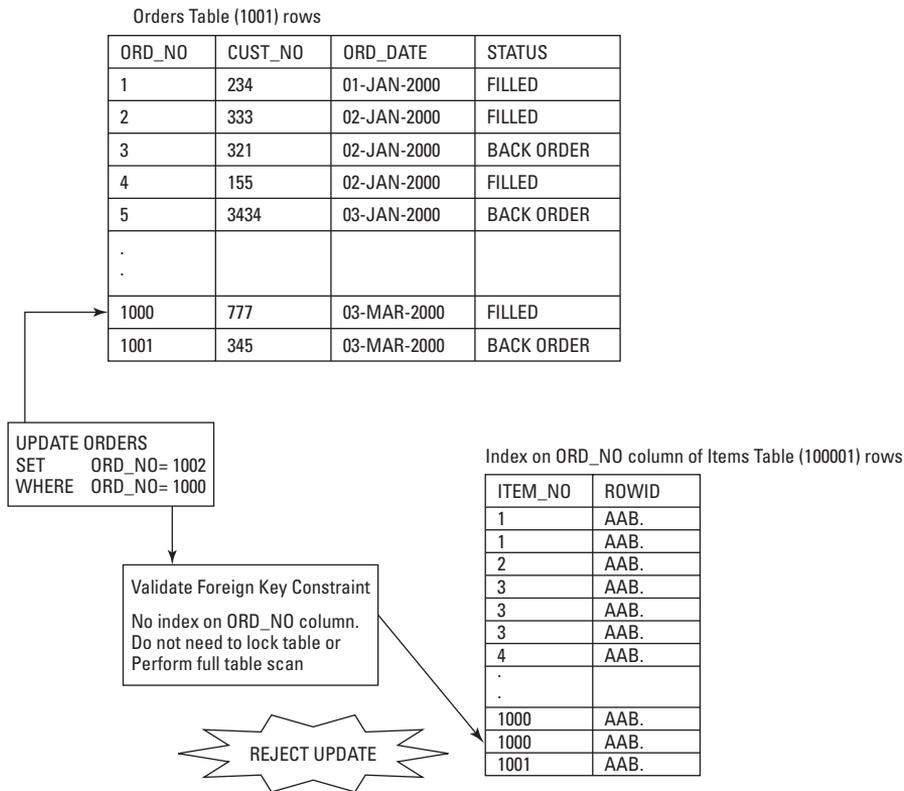


Figure 13-3: Preventing FOREIGN KEY locks on child tables

Implementing Constraints

Objective

Implement data integrity constraints

Constraints can either be implemented when tables are created or they can be added after the fact. When creating tables, constraints can either be defined as in-line (column-level) or out-of-line (table-level). The primary difference between in-line and out-of-line constraints, apart from some syntax, is that in-line constraints can only be applied against a single column. If you have a constraint over multiple columns it must be defined out-of-line or, if you are defining any constraint other than a NOT NULL, after the fact.

Here is an example of defining constraints during table creation. You can see that examples of both in-line and out-of-line constraints are used.

```

CREATE TABLE ITEMS (
    ITEM_ID NUMBER(10)
,   ORD_NO NUMBER(10) CONSTRAINT ITEMS_ORD_NO_FK REFERENCES ORDERS(ORD_NO)
    DEFERRABLE INITIALLY IMMEDIATE
,   PROD_ID NUMBER(5) NOT NULL
,   PRICE NUMBER(8,2) NOT NULL
,   QTY NUMBER(5) DEFAULT 1 NOT NULL,
    CONSTRAINT ITEMS_ITEM_ORD_NO_PK PRIMARY KEY (ITEM_ID,ORD_NO)
    USING INDEX TABLESPACE INDX)
TABLESPACE USERS

```

In this example, there are five constraints that are specified:

- ♦ FOREIGN KEY constraint on the ORD_NO column
- ♦ NOT NULL constraint on the PROD_ID column
- ♦ NOT NULL constraint on the PRICE
- ♦ NOT NULL constraint on the QTY column.
- ♦ A PRIMARY KEY constraint on the ITEM_ID,ORD_NO columns. This is an out-of-line constraint. It is defined at the end of the table definition.

Also in this example, we are specifying a location for the UNIQUE INDEX that will be created with the PRIMARY KEY. It will be stored in the INDX tablespace. The table will be stored in the USERS tablespace.

In-line (Column) constraints

Remember with in-line constraints, they are specified immediately after the column definition. Here is the syntax:

```

Column datatype [CONSTRAINT constraint_name]
{ [NOT] NULL
| PRIMARY KEY [USING INDEX index_clause]
| UNIQUE [USING INDEX index_clause]
| REFERENCES [schema.]table [(column)][ON DELETE CASCADE]
| CHECK (check row condition)
}
[NOT DEFERRABLE|DEFERRABLE [INITIALLY {IMMEDIATE|DEFERRED}]]
[DISABLE|ENABLE [VALIDATE|NOVALIDATE]]

```

The keywords in the syntax have the following meaning:

- ♦ CONSTRAINT lets you identify a name for the constraint. This is highly recommended because it makes future administration much simpler. If you do not specify a name for the constraint, Oracle will. The naming convention is SYS_C{SERIAL NUMBER}. So, when users violate constraints, they receive an error message like “ORA-##### violated constraint SYS_C3422323. Not very helpful.

- ♦ USING INDEX specifies that parameters are going to be used for the creation of an index. This is available for PRIMARY KEY and UNIQUE constraints.
- ♦ DEFERRABLE indicates when constraint checking will happen. (Used with either the INITIALLY IMMEDIATE or INITIALLY DEFERRED options.)
- ♦ NOT DEFERRABLE indicates that the constraint is not deferrable. When the SET CONSTRAINTS command is used, these constraints are checked after each DML statement, regardless of the settings specified with the SET CONSTRAINTS command.
- ♦ INITIALLY IMMEDIATE indicates that this constraint is set initially to check after each DML statement, however, if a SET CONSTRAINTS command is used to defer constraint checking, these constraints can be deferred until a COMMIT.
- ♦ INITIALLY DEFERRED indicates that this constraint is set to initially defer checking until commit.
- ♦ DISABLE by specifying this option, the constraint is defined in the data dictionary but not enabled. An ALTER TABLE command is required to ENABLE the constraint.
- ♦ REFERENCES is used for FOREIGN KEY constraints.
- ♦ ON DELETE CASCADE applies to FOREIGN KEY constraints. It implies that, when rows are deleted in the parent or referenced table, then rows with the same key value in this table should also be deleted.

Out-of-line (Table) constraints

The syntax for out-of-line constraints is similar. The only real differences are that multiple columns can be specified, NOT NULL constraints cannot be specified, and FOREIGN KEY constraint syntax is a little different. Apart from these three things they are identical. The definitions must follow the table definitions. Note that the comma “,” that appears at the end of the last line in the column definition separates the column definitions with out-of-line constraints. Here is the syntax:

```
[CONSTRAINT constraint_name]
{ PRIMARY KEY (column [, column ]... )
[USING INDEX index_clause]
| UNIQUE(column [, column ]... )
[USING INDEX index_clause]
| FOREIGN KEY (column [, column ]... )
REFERENCES [schema.]table [(column)] [ON DELETE CASCADE]
| CHECK (check row condition)
}
[NOT DEFERRABLE|DEFERRABLE [INITIALLY {IMMEDIATE|DEFERRED}]]
[DISABLE|ENABLE [VALIDATE|NOVALIDATE]]
```

Adding constraints to existing tables

If the table has already been created and you need to add a constraint, use the ALTER TABLE command. Here is the syntax:

```
ALTER TABLE table_name ADD (
  [CONSTRAINT constraint_name]
    { PRIMARY KEY (column [, column]... )
  [USING INDEX index_clause]
  | UNIQUE(column [, column]... )
  [USING INDEX index_clause]
  | FOREIGN KEY (column [, column]... )
  REFERENCES [schema.]table [(column)] [ON DELETE CASCADE]
  | CHECK (check row condition)
}
[NOT DEFERRABLE|DEFERRABLE [INITIALLY {IMMEDIATE|DEFERRED}]]
[DISABLE|ENABLE [VALIDATE|NOVALIDATE]]
```

Dropping constraints from existing tables

If the table has already been created and you need to drop (as opposed to disable) a constraint, use the ALTER TABLE DROP CONSTRAINT command. Note that when dropping a PRIMARY KEY or UNIQUE constraint, you do not need to know the constraint name. Otherwise you must first obtain the constraint name before it can be dropped. Here is the syntax:

```
DROP {PRIMARY KEY | UNIQUE (column)
      | CONSTRAINT constraint_name}
[CASCADE]
```

An example of dropping a primary key constraint would be:

```
ALTER TABLE EMPLOYEE DROP PRIMARY KEY
```

Here is an example of dropping a check constraint. The check constraint is on the DEPARTMENT table's DEPTNO column.

```
ALTER TABLE DEPARTMENT DROP CONSTRAINT DEPT_DEPTNO_CK;
```



The NOT NULL constraint cannot be added. If you want to make a column NOT NULL after the table has been created, use the ALTER TABLE *table_name* MODIFY *column_name* NOT NULL command.

Modifying constraints on existing tables

Constraints can also be modified. This is typically done to change a constraint state such as to validate a constraint that was enabled using the ENABLE NOVALIDATE option. You could modify the constraint to VALIDATE the existing data. If it is not the PRIMARY KEY you need to know the constraint name—another good reason for naming your constraints. You use the ALTER TABLE *table_name* MODIFY command. Here is the syntax.

```
MODIFY {PRIMARY KEY | UNIQUE (column)
        | CONSTRAINT constraint_name}
[NOT DEFERRABLE|DEFERRABLE [INITIALLY {IMMEDIATE|DEFERRED}]]
[DISABLE|ENABLE [VALIDATE|NOVALIDATE]]
```

Here is example of validating a primary key constraint:

```
ALTER TABLE MODIFY CONSTRAINT PRIMARY KEY VALIDATE;
```

General rules and conventions for implementing constraints

These are some rules and guidelines that you should follow with constraints.

- ♦ Always name your constraints. If constraints are not named, Oracle names them for you. This makes them more difficult to administer. Often times users are executing SQL statements through applications by way of SQL*Plus. Therefore, when they receive error messages, they do not always know what tables are being updated by the application. If the constraints are named using a predetermined naming convention, DBAs can generally determine what the problem is. In the following example, the same error is going to occur, a check constraint is going to be violated on the DEPARTMENT table. The first time, we will let Oracle name the constraint; the second time the constraint will be assigned a name when it is created. You will see that named constraints are much easier to troubleshoot.

```
CREATE TABLE DEPARTMENTS (
    ID NUMBER(5) PRIMARY KEY
    , NAME VARCHAR2(50) NOT NULL
    , CHECK (ID BETWEEN 10 and 50));
```

```
SQL> INSERT INTO DEPARTMENTS VALUES (
    2    60, 'SALES');
INSERT INTO DEPARTMENTS VALUES (
*
ERROR at line 1:
ORA-02290: check constraint (SYS.SYS_C00931) violated
```

Now, you see the error message that the user is going to report to the DBA. The user will generally not see the SQL statement that caused the exception. Therefore, the error message is all you have to go by. Not easy to resolve! You would need to query the DBA_CONSTRAINTS and DBA_CONS_COLUMNS data dictionary views to find out which table and columns this exception applies to.

Now in this example, the constraints are named and you can see the benefits when exceptions are raised.

```
CREATE TABLE DEPARTMENTS (
    ID NUMBER(5) CONSTRAINT DEPT_ID_PK PRIMARY KEY
    , NAME VARCHAR2(50) NOT NULL
    , CONSTRAINT DEPT_ID_RANGE_CHECK CHECK
    (ID BETWEEN 10 and 50));
```

```
SQL> INSERT INTO DEPARTMENTS VALUES (
    2    60, 'SALES');
INSERT INTO DEPARTMENTS VALUES (
*
ERROR at line 1:
```

```
ORA-02290: check constraint (SYS.DEPT_ID_RANGE_CHECK)
violated
```

Now this time when the user calls with the error report, you know that the table is the DEPT, the column is ID, and that a CHECK constraint was violated. Chances are that the second time the user receives the error, he or she will be able to fix the problem. A good and simple naming convention to follow is TABLENAME_COLUMN_TYPE where type is PK, UN, CK, NN, and FK.

- ♦ Always use the USING INDEX clause when creating PRIMARY KEY and UNIQUE constraints. This allows the indexes to be placed on separate tablespaces from the data.
- ♦ Always create indexes before constraints. If an index already exists for PRIMARY KEY and UNIQUE constraints, Oracle uses it. If it is created before the constraint, the DISABLE CONSTRAINT command does not drop the index. This means that it can still be used by other queries and when the ENABLE CONSTRAINT command is issued, there's no need to create an index, which saves lots of time.
- ♦ Consider using the DEFERRABLE option for FOREIGN KEY constraints. This will allow data to be loaded into these tables before loading the parent table. This is especially useful in a replicated environment.
- ♦ NOT NULL constraints cannot be added. You can only MODIFY columns.

```
ALTER TABLE DEPARTMENT MODIFY NAME NOT NULL;
```

Modifying Constraints

Objective

Maintain integrity constraints

There are some special considerations that need to be made when modifying a constraint. The most important deals with table locking. When the constraint state is changed to ENABLE, Oracle must lock the table. This prevents other users from making changes on this table. If the constraint being enabled is either a PRIMARY KEY or UNIQUE constraint, then Oracle also needs to create the index. If this is a large table, it may take some time. When the VALIDATE option is used for a constraint, Oracle has the ability to list all the violating rows. This can be useful when there are many rows in the table. Figure 13-4 illustrates the state of the data in tables when constraint states are changed.

Disabled novalidate

Constraints that are disabled novalidate are no longer checked. The table does not need to be locked so there is no wait time for users. The definition remains in the data dictionary so that constraints can be enabled at some future time. For PRIMARY

KEY and UNIQUE constraints, Oracle also drops the indexes used to maintain these constraints if they were created when the constraint was enabled. If the constraint was using a pre-existing index, then that index is not dropped.

It is common for DBAs to use this state when loading large amounts of data. Disabling the constraint means the load performs much quicker.

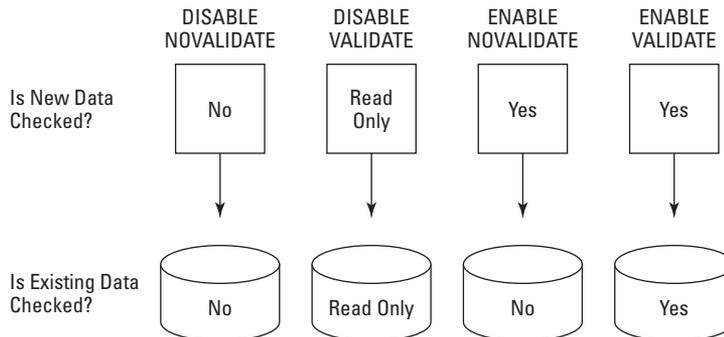


Figure 13-4: The state of the data when constraint states are changed

Disabled validate

The disabled validate state renders the table read only. No DML operations can be performed on the existing data and new data can be inserted. The constraint is disabled and there are no locks on the table, but only queries are permitted. For PRIMARY KEY and UNIQUE constraints Oracle also drops the indexes used to maintain these constraints if they were created when the constraint was enabled. If the constraint was using a pre-existing index, then that index is not dropped.

Enabled novalidate

Constraints in the enabled novalidate state are checked, but only on future DML statements. The existing data is not checked which means that the table does not need to be locked. This is a big bonus for DBAs who load large amounts of data or who need to process large batch jobs. When the load or job completes, they can turn the constraint back on and let the users start processing transactions against this table.

The only risk is that there may be data that violates the constraint in the table. This is often the case if there were problems with the data that was being loaded. If this is not a risk, it is safe to leave the constraint in this state. If there is a chance that the data is invalid, DBAs generally pick a non-busy time to try to figure out where the problems are.

**Caution**

If you are enabling a PRIMARY KEY or UNIQUE constraint, always have the indexes created before issuing this command. The benefit is that the table is not locked, however, if indexes need to be created, as is the case with PRIMARY KEY and UNIQUE constraints, Oracle locks the table anyway to perform this action. There would be no benefit.

Enabled validate

Enabled validate is the default state when creating or enabling constraints. All new data is checked against the constraint, and all existing data is checked. The table is locked and all changes are rejected until the validations are done. Furthermore, the constraint is not enabled until all existing data meets the constraint conditions.

If the constraint were in a state of DISABLE VALIDATE, constraints are enforced on the existing data and no new changes are permitted. It is READ ONLY. When using the ENABLE VALIDATE clause, the table does not need to be locked if the indexes for PRIMARY and UNIQUE constraints already exist.

When using this option, Oracle checks the existing data. If there is a violation, it does not enable the constraint. It reports an error message. Finding the offending rows can be done by using the EXCEPTIONS option of the ENABLE command. By using the EXCEPTIONS option, Oracle goes through the entire table and reports all errors as opposed to stopping when it finds the first. You can then fix the errors and reissue the ENABLE command. It should succeed the second time through. The only time it does not is if you missed fixing a row or if there were multiple errors on the same row. While Oracle will go through the entire table, Oracle stops checking the current row when it finds a violation. Therefore, if there are multiple violations on the same row, Oracle only reports the first one it finds.

Assume there is a table called DEPARTMENTS that has a PRIMARY KEY constraint on the DEPTNO column. The constraint was disabled to do a bulk load. The constraint now needs to be enabled, but you suspect there might be violations. The following steps guide you through the process of determining the violations.

STEP BY STEP: Enabling constraints using the exceptions table

1. Create the exceptions table if it does not already exist. Oracle has a script in the ORACLE_HOME/rdbms/admin directory called utlexcpt.sql. Run this script and if it completes, you now have a table called EXCEPTIONS. If it fails, it may be because the table already exists. If this is the case, then you do not need to create it, but you may want to inquire about the contents of the table. You may want to truncate the table before starting, just make sure no one else is in the process of using the table.

```
@%ORACLE_HOME%\RDBMS\ADMIN\utlexcpt.sql
```

2. Execute the ALTER TABLE with EXCEPTIONS clause command:

```
ALTER TABLE DEPARTMENTS ADD
  CONSTRAINT DEPT_DEPTNO_PK PRIMARY KEY (DEPTNO)
EXCEPTIONS INTO SYS.EXCEPTIONS;
```

```
ERROR at line 2:
ORA-02437: cannot validate (SYS.DEPT_DEPT_NO_PK) - primary
key violated
```

3. Identify the invalid data by querying the EXCEPTIONS table. The EXCEPTIONS table contains a column called HEAD_ROWID which is the ROWIDs from the DEPARTMENT table that have invalid data.

```
1 SELECT D.ROWID,D.DEPTNO FROM DEPARTMENTS D
2 WHERE ROWID in (SELECT ROW_ID
3                 FROM EXCEPTIONS
4                 WHERE TABLE_NAME='DEPARTMENTS')
5* ORDER BY D.DEPTNO
SQL> /
```

ROWID	DEPTNO
-----	-----
AAAAE9aAABAAAEwCAAE	5
AAAAE9aAABAAAEwCABk	5
AAAAE9aAABAAAEwCAAO	15
AAAAE9aAABAAAEwCABl	15
AAAAE9aAABAAAEwCAAY	25
AAAAE9aAABAAAEwCABm	25
AAAAE9aAABAAAEwCAAi	35
AAAAE9aAABAAAEwCABn	35
AAAAE9aAABAAAEwCAAs	45
AAAAE9aAABAAAEwCABo	45
AAAAE9aAABAAAEwCAA2	55

ROWID	DEPTNO
-----	-----
AAAAE9aAABAAAEwCABp	55
AAAAE9aAABAAAEwCABA	65
AAAAE9aAABAAAEwCABq	65
AAAAE9aAABAAAEwCABK	75
AAAAE9aAABAAAEwCABr	75
AAAAE9aAABAAAEwCABU	85
AAAAE9aAABAAAEwCABs	85
AAAAE9aAABAAAEwCABe	95
AAAAE9aAABAAAEwCABt	95

20 rows selected.

There are 20 rows that violate the constraint. It appears that all department numbers that end in a 5 have duplicates. You need to fix this before the constraint can be enabled.

4. Fix the violating rows. Now, this step is going to be different for almost all constraints. It is easy enough for PRIMARY KEY and UNIQUE constraints, because you can just delete the duplicate value. I am assuming that these rows were loaded twice:

```
DELETE FROM DEPARTMENTS OUTER
WHERE ROWID < (SELECT MAX(ROWID)
               FROM DEPARTMENTS INNER
               WHERE INNER.DEPTNO = OUTER.DEPTNO);
```

10 rows deleted.

5. Delete all rows in the EXCEPTIONS table for the DEPARTMENT TABLE and re-enable the constraint as follows:

```
DELETE FROM EXCEPTIONS WHERE TABLE_NAME = 'DEPARTMENT';

ALTER TABLE DEPARTMENTS ADD
  CONSTRAINT DEPT_DEPT_NO_PK PRIMARY KEY (DEPTNO)
EXCEPTIONS INTO EXCEPTIONS
```

Table altered.

Getting Constraint Information

Objective

Obtain constraint information from the data dictionary

The two main views for obtaining constraint information. These views can be used to determine what constraints have been created for a table, what columns the constraint applies to, what the CHECK constraint row condition is, the status of the constraint and when the constraints are checked, DEFERRED or NON-DEFERRED.

DBA_CONSTRAINTS

The DBA_CONSTRAINTS data dictionary view contains the following columns:

<i>COLUMN NAME</i>	<i>DESCRIPTION</i>
OWNER	Constraint owner
CONSTRAINT_NAME	Constraint name

Continued

COLUMN NAME	DESCRIPTION
CONSTRAINT_TYPE	Constraint type (P = PRIMARY KEY, U = UNIQUE, C = NOT NULL, C = CHECK, R = FOREIGN KEY) NOTE that NOT NULL and CHECK both use C.
TABLE_NAME	Table name constraint is applied against
SEARCH_CONDITION	Check constraint row condition
R_OWNER	Owner of table referenced by FOREIGN KEY
R_CONSTRAINT_NAME	PRIMARY KEY or UNIQUE constraint name that the FOREIGN KEY constraint references
DELETE_RULE	Is the ON DELETE CASCADE option enabled for FOREIGN KEY constraints
STATUS	Status of constraint (ENABLED or DISABLED)
DEFERRABLE	Is the constraint deferrable (DEFERRABLE or NOT DEFERRABLE)
DEFERRED	Is the constraint set to be deferrable INITIALLY IMMEDIATE or INITIALLY DEFERRED? The values are (DEFERRED or IMMEDIATE) and these only apply to deferrable constraints.
VALIDATED	Is the constraint validated (VALIDATED or NOT VALIDATED)
GENERATED	Was the name of the constraint generated by Oracle or was the constraint named when it was created? (GENERATED NAME or USER NAME)
BAD	Indicates that the constraint is to be rewritten. This might happen with CHECK constraints that in a previous release used a two-digit year.
RELY	Setting the RELY flag is something the DBA will do if they are confident that the constraints are valid but do not want to perform the validation. DBAs may do this to avoid the costs of validating them. If it is set, it tells the optimizer that the constraint is VALIDATED.
LAST_CHANGE	The date the constraint was last validated.

Here is a typical query against DBA_CONSTRAINTS:

```
SELECT TABLE_NAME, CONSTRAINT_NAME
       , CONSTRAINT_TYPE, DEFERRABLE, DEFERRED
       , VALIDATED, STATUS
FROM   DBA_CONSTRAINTS;
```

```
TABLE_NAME  CONSTRAINT_NAME  C DEFERRABLE      DEFERRED  VALIDATED  STATUS
-----
ORDERS      SYS_C00912      C NOT DEFERRABLE IMMEDIATE  VALIDATED  ENABLED
```

ORDERS	ORDERS_ORD_NO_PK	P	NOT DEFERRABLE	IMMEDIATE	VALIDATED	ENABLED
ITEMS	SYS_C00925	C	NOT DEFERRABLE	IMMEDIATE	VALIDATED	ENABLED
ITEMS	SYS_C00926	C	NOT DEFERRABLE	IMMEDIATE	VALIDATED	ENABLED
ITEMS	SYS_C00927	C	NOT DEFERRABLE	IMMEDIATE	VALIDATED	ENABLED
T1	T1_X	P	NOT DEFERRABLE	IMMEDIATE	NOT VALIDATED	DISABLED
ITEMS	ITEMS_ORD_NO_FK	R	DEFERRABLE	IMMEDIATE	VALIDATED	ENABLED
DEPARTMENTS	DEPT_DEPT_NO_PK	P	NOT DEFERRABLE	IMMEDIATE	VALIDATED	ENABLED

DBA_CONS_COLUMNS

The `DBA_CONS_COLUMNS` data dictionary view contains information about the columns that make up a constraint.

<i>COLUMN NAME</i>	<i>DESCRIPTION</i>
<code>OWNER</code>	Constraint owner
<code>CONSTRAINT_NAME</code>	Constraint name
<code>TABLE_NAME</code>	Table name constraint is applied against
<code>COLUMN_NAME</code>	Column Name referenced the constraint
<code>POSITION</code>	Position of column for composite constraints

This view is generally joined to `DBA_CONSTRAINTS` to retrieve constraint information.

To determine which columns in the `ITEMS` table have constraints and the type of constraint, run the following query.

```
SELECT CO.TABLE_NAME,CO.CONSTRAINT_NAME,CO.CONSTRAINT_TYPE,
       CO.STATUS,CC.COLUMN_NAME
FROM   DBA_CONS_COLUMNS CC, DBA_CONSTRAINTS CO
WHERE  CO.CONSTRAINT_NAME = CC.CONSTRAINT_NAME
AND    CO.TABLE_NAME = 'ITEMS'
ORDER BY CO.CONSTRAINT_NAME
```

TABLE_NAME	CONSTRAINT_NAME	C	STATUS	COLUMN_NAME
ITEMS	ITEMS_ORD_NO_FK	R	ENABLED	ORD_NO
ITEMS	ITEM_IT_ORD_PK	P	ENABLED	ITEM_ID
ITEMS	ITEM_IT_ORD_PK	P	ENABLED	ORD_NO
ITEMS	SYS_C00925	C	ENABLED	PROD_ID
ITEMS	SYS_C00926	C	ENABLED	PRICE
ITEMS	SYS_C00927	C	ENABLED	QTY

6 rows selected.

Other helpful queries

The `DBA_INDEXES` data dictionary view contains the index information. It can be used to see if `PRIMARY KEY` or `UNIQUE` constraints are being maintained by `UNIQUE` or `NON-UNIQUE` indexes. The `DBA_IND_COLUMNS` can be linked to `DBA_CONS_COLUMNS` to find all indexes on constraints. This might be useful if you want to ensure that all `FOREIGN KEY` constraints have indexes.

```
select constraint_name
,      table_name
,      column_name
from    dba_cons_columns
where   (table_name,column_name) not in
        (select table_name,column_name
         from    dba_ind_columns)
-- and table_name in ('HELP1','ITEMS','ORDERS','DEPARTMENTS')
and     constraint_name in
```

This query returns `FOREIGN KEY` constraints that do not have indexes. The `--` at the beginning of the line indicates a comment. You could specify individual tables if you wanted to narrow your search and remove the `--`.

To select `FOREIGN KEY` constraints and the constraint information for the tables they reference, use the following query. This query is using the `ITEMS` table, but you can substitute for any table name.

```
SELECT      CH.CONSTRAINT_NAME AS FOREIGN_KEY
,          PA.CONSTRAINT_NAME AS PARENT
,          PA.CONSTRAINT_TYPE
,          PA.TABLE_NAME
FROM        DBA_CONSTRAINTS CH, DBA_CONSTRAINTS PA
WHERE       CH.TABLE_NAME = 'ITEMS'
AND         CH.CONSTRAINT_TYPE = 'R'
AND         CH.R_CONSTRAINT_NAME = PA.CONSTRAINT_NAME

FOREIGN_KEY      PARENT          C TABLE_NAME
-----
ITEMS_ORD_NO_FK  ORDERS_ORD_NO_PK P ORDERS
```

Key Point Summary

Data integrity is a key element of any relational database and Oracle is no different. Oracle is designed for performing fast and efficient data integrity checks so Oracle recommends that they are used instead of application code or triggers. The database constraints do have some limitations such as `FOREIGN KEY` constraints on linked tables and in these situations using one of the other data integrity options is required. The final decision as to which method to use really does depend on the situation.

There are five Oracle database constraints:

- ♦ NOT NULL constraints prevent null values from being entered into the column. They can be specified when the table is created or by using the ALTER TABLE table_name MODIFY column_name NOT NULL. They cannot be added.
- ♦ UNIQUE constraints ensure uniqueness over a column or collection of columns for composite UNIQUE constraints. Oracle will always maintain UNIQUE constraints using an index. If an index does not exist and the constraint is NOT deferrable then Oracle will create a unique index. If the constraint is deferrable then Oracle must use a non-unique index. If an index already exists on the leading column of the constraint then Oracle will use that index instead of creating another one. It is always recommended that indexes be created before UNIQUE and PRIMARY KEY constraints. This prevents the indexes from being dropped when constraints are DISABLED.
- ♦ PRIMARY KEY constraints are simply a combination of NOT NULL and UNIQUE. You can only have one PRIMARY KEY constraint but as many combinations of UNIQUE and NOT NULL as required for a table.
- ♦ FOREIGN KEY constraints perform referential integrity checks on data in other tables or in the same table. Checks on the same table are called SELF REFERENCING FOREIGN KEYS. The syntax is no different, but these types of constraints should always be DEFERRED. When constraints are DEFERRED the constraints are not checked until a commit is issued.
- ♦ CHECK constraints allow you to place conditions on columns at the row level. This is a very powerful and flexible tool for implementing business rules inside of database as no application code is required.
- ♦ All constraints can be added when tables are being created. The NOT NULL constraint is the only one that cannot be added to an existing table with the ALTER TABLE table_name ADD constraint_type (column_name).
- ♦ The decision between using in-line or out-of-line constraints depends upon the constraint and when the constraints are being added. If constraints are being added at table creation time, then only constraints on multiple columns need to be specified as out-of-line. However, when adding constraints with the ALTER TABLE command, all constraints must be specified out-of-line. The only different syntax between the two options is how FOREIGN KEY constraints are enabled.
- ♦ Always name your constraints and adhere to a naming convention. This makes for simpler administration.
- ♦ Constraints can be in one of two states, ENABLED (default) or DISABLED.
- ♦ A constraint that is ENABLED checks all DML statements, ensuring no constraints are violated.
- ♦ A constraint that is DISABLED does not check DML statements.

- ♦ The time when constraints are checked can also be controlled. A deferrable constraint can be set to check constraints when the transaction is completed as opposed to when the statement is completed. This is especially useful for maintaining PRIMARY KEY or UNIQUE constraints with FOREIGN KEYS.
- ♦ Deferrable PRIMARY KEY and UNIQUE constraints have special requirements for indexing. The indexes used to maintain these constraints will be non-unique.
- ♦ Constraint information can be found in DBA_CONSTRAINTS and DBA_CONS_COLUMNS data dictionary views.



STUDY GUIDE

Now that you have learned about data integrity, you should test your understanding by reviewing the assessment questions and performing the exercises below.

Assessment Questions

1. How are UNIQUE constraints different from PRIMARY KEY constraints?
 - A. UNIQUE constraints allow duplicates.
 - B. PRIMARY KEY constraints allow duplicates.
 - C. PRIMARY KEY constraints are maintained using indexes.
 - D. UNIQUE constraints are maintained using indexes.
 - E. UNIQUE constraints do not restrict nulls.
2. What is NOT TRUE about FOREIGN KEY constraints?
 - A. FOREIGN KEY constraints are used for referential integrity.
 - B. FOREIGN KEY constraints must reference either PRIMARY KEY or UNIQUE constraints.
 - C. FOREIGN KEY constraints accept nulls by default.
 - D. FOREIGN KEY constraints are maintained by indexes.
 - E. FOREIGN KEY constraints CANNOT reference tables in other databases.
3. Under what circumstances can Oracle use existing indexes for PRIMARY KEY and UNIQUE constraints? (Pick two answers.)
 - A. When the constraints are deferrable.
 - B. When the index already exists and is valid.
 - C. When the constraints are initially created using the DISABLE option.
 - D. When a composite index exists where the leading column is the same as the leading column for the constraint.
 - E. When the USING INDEX option is specified with the constraint.

4. What is true about a constraint that is enabled with the NOVALIDATE option?
 - A. Oracle checks future DML statements but not the existing data.
 - B. No future DML is permitted until the constraint is enabled VALIDATE.
 - C. This is an illegal action.
 - D. The table is effectively made READ ONLY.
 - E. The existing table data is checked but not future DML statements.

5. What is true about a constraint that is enabled with the VALIDATE option?
 - A. Oracle checks future DML statements but not the existing data.
 - B. No future DML is permitted until the constraint is enabled VALIDATE.
 - C. The existing table data is checked as well as future DML statements.
 - D. The table is effectively made READ ONLY.
 - E. The existing table data is checked but not future DML statements.

6. Why would you change a constraint to DISABLE VALIDATE?
 - A. You want Oracle to check the existing data in the table for violations.
 - B. This is an invalid command.
 - C. You want to make the table READ ONLY.
 - D. You want to relocate the indexes for PRIMARY KEY and UNIQUE constraints to different tablespaces.
 - E. You want to allow INSERTS of new data but prevent DELETES and UPDATES of the existing data.

7. How do you enable a PRIMARY KEY constraint named EMP_ID_PK?
 - A. ALTER DATABASE ENABLE PRIMARY KEY EMP_ID_PK
 - B. ALTER TABLE *table_name* ENABLE PRIMARY KEY
 - C. ALTER TABLE *table_name* ADD PRIMARY KEY (*emp_id*)
 - D. ALTER TABLE *table_name* ENABLE CONSTRAINT EMP_ID_PK
 - E. Both B and D

8. How do you make the LNAME column of the EMPLOYEE table NOT NULL?
 - A. ALTER TABLE EMPLOYEE ADD NOT NULL (LNAME)
 - B. ALTER TABLE EMPLOYEE (LNAME NOT NULL)
 - C. ALTER TABLE EMPLOYEE MODIFY LNAME NOT NULL
 - D. ALTER TABLE EMPLOYEE ENABLE NOT NULL (LNAME)
 - E. You cannot change a column to NOT NULL after the table has been created.

9. What constraint would you use to restrict a range of values in a column?
- A. PRIMARY KEY
 - B. UNIQUE
 - C. CHECK
 - D. You must use a TRIGGER for this type of constraint.
 - E. You cannot limit a column to a range of values.
10. What is the EXCEPTIONS table used for?
- A. Keeping a list of exceptions for constraints that are ENABLE NOVALIDATE so that they can be fixed later
 - B. Recording trigger actions
 - C. Storing the invalidations for transactions that are using DEFERRABLE constraints
 - D. Storing information on rows that failed the constraint check during an ALTER TABLE *table_name* ENABLE constraint command
 - E. Storing the CHECK constraint rules.
11. What statement is true about PRIMARY KEY constraints that are set to be DEFERRABLE?
- A. Oracle must maintain these constraints using a UNIQUE index.
 - B. Oracle will not maintain these constraints using indexes.
 - C. PRIMARY KEY constraints cannot be deferred.
 - D. Oracle will maintain them using NON UNIQUE indexes.
 - E. They must be created using the INITIALLY DEFERRED option.
12. Which command is used to drop a constraint?
- A. ALTER DATABASE DROP CONSTRAINT *constraint_name*
 - B. DROP CONSTRAINT *constraint_name*
 - C. ALTER CONSTRAINT *constraint_name* DROP
 - D. ALTER TABLE *table_name* DROP CONSTRAINT *constraint_name*
 - E. Constraints cannot be dropped only disabled.

13. What statement is true about the ON DELETE CASCADE option of the FOREIGN KEY constraint?
- A. When rows in the referenced or parent table are dropped, corresponding rows in the table with the FOREIGN KEY will also be dropped.
 - B. Deletes are prevented in the parent table.
 - C. User will be prompted before rows in the child table are deleted.
 - D. This is only available with database triggers.
 - E. When the parent table is dropped so is the child table.
14. What data dictionary view contains information about whether or not a constraint is ENABLED or DISABLED?
- A. DBA_CONS_COLUMNS
 - B. DBA_CONSTRAINT_STATE
 - C. DBA_CONSTRAINTS
 - D. DBA_CONSTRAINTS_ENABLED
 - E. DBA_TABLES
15. When can in-line constraints not be used?
- A. When creating tables
 - B. When creating composite constraints
 - C. When adding constraints to an existing table
 - D. When creating CHECK constraints
 - E. Both B and C

Scenarios

1. You have just been hired as a DBA by We Want Your Business Ltd. One of your jobs as a DBA is to try and improve the performance of a bulk load that is done once a week. Sales data is loaded from another system and the process is taking several hours to complete. They are a 7/24 shop and have very little downtime. The data is coming from another system and there are generally no problems with it. It has always loaded without any problems in the past, it is just the amount of time that it is taking that is the biggest concern.

What recommendations would you make with regard to constraints to improve the performance and how would you implement this?

2. You have been hired as a consultant by JOBS JOBS JOBS Ltd to examine some severe performance and locking problems in the database. The database is very busy with many concurrent updates. Upon further investigation you discover that the problems appear to be happening when large tables are being modified. These tables are all referenced by FOREIGN KEY constraints.

What should you investigate further about these problems? And, if your theories are correct, how do you go about resolving the problem?

Lab Exercises

Lab 13-1 Creating and Maintaining Constraints

1. Connect to your instance using Server Manager line mode (svrmgrl) as a user with SYSDBA privileges.
2. Determine the constraint type, constraint name, column name, and status for the STUDENT.LOCATIONS table.
3. Add a CHECK constraint called LOCATIONS_COUNTRY_CK to the COUNTRY column of the LOCATIONS table so that the only valid values for the column are CANADA and USA. Be sure to prevent NULL values.
4. Verify that the constraint has been created.
5. Run the following update to test the constraint. CANADA is intentionally spelled incorrectly.

```
UPDATE STUDENT.LOCATIONS
SET    COUNTRY = 'CANNADA'
WHERE LOCATIONID = 100;
```

6. Disable the check constraint created in question 3.
7. Reissue the statement in question 5.
8. What happened and why?
9. Issue a Rollback command.
10. Drop the constraint created in question 3.

Lab 13-2 Deferrable Constraints

1. Connect to your instance using Server Manager line mode (svrmgrl) as a user with SYSDBA privileges.
2. Determine the constraint type, constraint name, column name, and status for the STUDENT.LOCATIONS table.
3. Update the SCHEDULEDCLASSES table and change the value of the LOCATIONID column 999.

4. What happened and why?
5. Modify the FK_SCHEDULEDCLASSES_LOCATIONID and enable deferred constraint checking. Use the INITIALLY IMMEDIATE option of the MODIFY COMMAND.
6. Rerun the update statement attempted in question 3.
7. What happened and why?
8. Drop the constraint and recreate it this time making it DEFERRABLE.
9. Issue the ALTER SESSION SET CONSTRAINTS = DEFERRED command to defer constraint checking until commits are issued.
10. Rerun the update statement attempted in question 3.
11. What happens this time?
12. Issue a commit.
13. What happens and why?

Answers to Chapter Questions

Chapter Pre-Test

1. The following is the list of the five Oracle database constraints:
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY (Combination of a NOT NULL and UNIQUE Constraint)
 - FOREIGN KEY
 - CHECK
2. A PRIMARY KEY constraint is simply a combination of a UNIQUE and NOT NULL constraint. PRIMARY KEYS ensure that every row in a table can be uniquely identified. It maintains PRIMARY KEYS by using an index. This index ensures that the constraint can be efficiently validated. You can only have one PRIMARY KEY constraint per table but you can have as many UNIQUE and NOT NULL constraints as needed. PRIMARY KEY constraints are not mandatory but strongly recommended to ensure data integrity.
3. The only difference between a PRIMARY KEY and a UNIQUE constraint is that PRIMARY KEYS do not allow NULLs. Since the UNIQUE constraint is maintained with an index, and NULLs are not stored in the index, they do not validate the UNIQUE constraint. When a NOT NULL constraint is added to a UNIQUE constraint they validate data the exact same way as a PRIMARY KEY constraint.

4. DEFERRABLE constraints are constraints that are validated upon commit as opposed to after every DML statement. By default constraints are checked after every DML statement. This means that you could not issue a DML statement that violates a constraint. When the constraints are deferred, Oracle will not validate the constraint until the user commits. This means that a user could violate a constraint with a DML statement. The violation will be reported to the user when a commit is issued. If the user resolves the violation with another DML statement before the commit occurs, and the constraint is no longer violated, no error message is returned.
5. The main benefit of disabling a constraint is performance. Whenever a constraint is enabled, Oracle must check every DML statement to ensure that the constraints are not violated. If you are performing a bulk load or update of thousands or even millions of rows of data, the costs of maintaining these constraints after each statement is high. If the constraint is disabled before the batch job, the job will complete much quicker. When the job completes, just enable the constraint. When enabling constraints you have several options but if you stick with the defaults and enable the constraint for all rows in the table performing this task is much less time consuming than performing the same check after each insert. Doing it once for all rows is generally faster than doing it after every row of a batch job.
6. The two main data dictionary views for retrieving constraint information are DBA_CONSTRAINTS and DBA_CONS_COLUMNS. DBA_CONSTRAINTS contains the table name, constraint name, constraint type, and the status of the constraint while DBA_CONS_COLUMNS contains the constraint name and the columns that the constraint is associated with.
7. The main difference between an in-line versus an out-of-line constraint is simply where in the table definition the constraint is declared. If it is declared with the column, then it is in-line. Certain types of constraints, like PRIMARY KEY constraints on multiple columns, cannot be declared in-line. These are considered out-of-line constraints. Any constraint can be declared out-of-line but only constraints on individual columns can be declared in-line.
8. A database trigger is an event that is triggered or fired when certain events happen. They are placed on Tables and are hidden from the user performing the action. There are 12 types of database triggers. Triggers can be created on INSERT, UPDATE, and DELETE statements. The triggers can be set to fire BEFORE or AFTER every STATEMENT and they can also be set to fire ONLY ONCE per statement or set to fire ONCE for every row that is affected. An example is an UPDATE trigger that is set to fire whenever an update is made on the SALARY column of a table. Every time a user issues an update of the column the trigger performs whatever action it was programmed to do, like recording the change in an audit table. Triggers do not require any action to be taken by the user as they fire automatically. This makes triggers a good way to audit changes made in the database.

9. The exceptions table can be used to record the ROWIDs of the rows that violate the constraint being enabled. This allows the DBA to enable a constraint and have all violating rows reported in the EXCEPTIONS table. By processing the entire table, the DBA only needs to enable the constraint once that will report all the violating rows, fix all the rows with violations, and then issue the enable statement again. By proceeding in this sequence, the DBA does not need to reissue the statement every time an exception is found. This can save a tremendous amount of time.

Assessment Questions

1. **E.** PRIMARY KEY constraints are a combination of UNIQUE and NOT NULL constraints. UNIQUE constraints allow nulls because the uniqueness is maintained by an index and Oracle does not store NULL values in the index. NOT NULL constraints can be added to UNIQUE constraints. Refer to the section on “Integrity Constraints in Oracle” for more information.
2. **D.** While it is always recommended to create indexes on columns with a FOREIGN KEY constraint, it is not required. They are not maintained by indexes. Answer B is true as FOREIGN KEYS must reference a column with a UNIQUE or PRIMARY KEY constraint. Refer to the section on “Integrity Constraints in Oracle” for more information.
3. **B and D.** Oracle will use existing indexes if one already exists. It will not create two indexes for the same column. Also, if there is a composite index on the table where the leading column is the first column of the constraint, they do not need to be UNIQUE indexes. Refer to the section on “Special Considerations for Constraints.”
4. **A.** ENABLE NOVALIDATE checks future DML but not the existing data. This means that there can be rows that violate the constraint inside the table. Refer to the section on “Constraint States” for more information.
5. **C.** This is the default when enabling constraints. Both future and existing data is checked. Refer to the section on “Constraint States” for more information.
6. **C.** DISABLE VALIDATE turns off the constraint but still enforces it. However, it is not just the constraint that is enforced, all DML statements against the table, even ones that would not violate the constraint are disallowed. This makes the table effectively READ ONLY. Refer to the section on “Constraint States” for more information.
7. **E.** The syntax for both **B** and **D** are valid. Refer to the section on “Modifying Constraints” for more information.
8. **C.** You must use the MODIFY option of the ALTER TABLE command. More information can be found in the section on “Modifying Constraints.”
9. **C.** CHECK constraints are very capable of enforcing this rule. Refer to the section on “Types of Constraints.”

10. **D.** The exceptions table is used for storing rows that violate the constraint that is being enabled. Oracle will process all the rows in the table, report all the errors in the exceptions table, but not `ENABLE` the constraint. Refer to the section on “Modifying Constraints” for more information.
11. **D.** Any time a `PRIMARY KEY` or `UNIQUE` constraint are deferrable, Oracle must maintain them using `NON-UNIQUE` indexes. Since constraint checking is deferred until commit. This would not be allowed if `UNIQUE` indexes were being used because the insert would fail when adding the entry to the index. Refer to section “Special Considerations for Constraints” for more information.
12. **D.** Refer to section “Modifying Constraints” for more information.
13. **A.** The `ON DELETE CASCADE` option is used to delete rows from a child table when the corresponding rows in the parent table are dropped. This is done automatically and does not need to be coded as two transactions. Refer to the “Types of Constraints” section for more information.
14. **C.** `DBA_CONSTRAINTS` contains all constraint information except which column or columns the constraint is applied against. This information exists in the `DBA_CONS_COLUMNS` view. More information exists in the “Getting Information on Constraints” section.
15. **E.** Both **B** and **C** are true. In-line constraints cannot be used for composite columns or when adding constraints to existing tables. In-line constraints are always specified when the column is being defined, therefore, it would be impossible to specify composite constraints in-line.

Scenarios

1. Based on the fact that the data has never had problems with constraint violations in the past, the best way to improve performance would be to `DISABLE` all the constraints on the tables being loaded. This will dramatically improve the performance of the load. Since the data is always good, when you `ENABLE` the constraints after the load has completed the `ENABLE NOVALIDATE` option should be used. This validates future changes but not the existing data. This should be fine as there have not been any problems in the past. You could then, during one of the slow periods, `VALIDATE` the data in the tables by using the `ENABLE VALIDATE` command to ensure that the load was indeed successful.
2. Since the problems appear to be happening when the large tables are being updated and the large tables are being referenced with `FOREIGN KEYS`, you should investigate the presence of indexes on the `FOREIGN KEY` columns. Without indexes on the `FOREIGN KEY` columns any modifications made by Oracle must lock the entire table to perform a full table scan. This increase in locking will force other transactions that are making modifications to the table to wait for the lock to be released. The presence of indexes on the `FOREIGN KEY` columns means that Oracle is not forced to lock the entire table to perform the scan. It can simply scan the index. This eliminates the locks being placed on the table and reduces the number of transactions that are going to be waiting for locks to be freed.

Lab Exercises

Lab 13-1

2.

```

1 SELECT CO.CONSTRAINT_NAME,CO.CONSTRAINT_TYPE,
      CO.STATUS,CO.TABLE_NAME,CC.COLUMN_NAME
2 FROM   DBA_CONSTRAINTS CO, DBA_CONS_COLUMNS CC
3 WHERE  CC.TABLE_NAME = CO.TABLE_NAME
4 AND    CC.CONSTRAINT_NAME = CO.CONSTRAINT_NAME
5 AND    CO.TABLE_NAME = 'LOCATIONS'
6* AND   CO.OWNER = 'STUDENT'
SQL> /

```

CONSTRAINT_NAME	C STATUS	TABLE_NAME	COLUMN_NAME
SYS_C00958	C ENABLED	LOCATIONS	LOCATIONID
SYS_C00959	C ENABLED	LOCATIONS	LOCATIONNAME
PK_LOCATIONID	P ENABLED	LOCATIONS	LOCATIONID

3.

```

ALTER TABLE STUDENT.LOCATIONS
ADD CONSTRAINT LOCATIONS_COUNTRY_CK CHECK
((UPPER(COUNTRY)='CANADA' or UPPER(COUNTRY)='USA')
 and COUNTRY is NOT NULL)

```

4.

```

1 SELECT CO.CONSTRAINT_NAME,CO.CONSTRAINT_TYPE,
      CO.STATUS,CO.TABLE_NAME,CC.COLUMN_NAME
2 FROM   DBA_CONSTRAINTS CO, DBA_CONS_COLUMNS CC
3 WHERE  CC.TABLE_NAME = CO.TABLE_NAME
4 AND    CC.CONSTRAINT_NAME = CO.CONSTRAINT_NAME
5 AND    CO.TABLE_NAME = 'LOCATIONS'
6* AND   CO.OWNER = 'STUDENT'
SQL> /

```

CONSTRAINT_NAME	C STATUS	TABLE_NAME	COLUMN_NAME
SYS_C00958	C ENABLED	LOCATIONS	LOCATIONID
SYS_C00959	C ENABLED	LOCATIONS	LOCATIONNAME
PK_LOCATIONID	P ENABLED	LOCATIONS	LOCATIONID
LOCATIONS_COUNTRY_CK	C ENABLED	LOCATIONS	COUNTRY

5.

```
UPDATE STUDENT.LOCATIONS
SET   COUNTRY = 'CANNADA'
WHERE LOCATIONID = 100;
```

```
UPDATE STUDENT.LOCATIONS
*
```

```
ERROR at line 1:
ORA-02290: check constraint (STUDENT.LOCATIONS_COUNTRY_CK)
violated
```

6.

```
ALTER TABLE STUDENT.LOCATIONS DISABLE CONSTRAINT
LOCATIONS_COUNTRY_CK;
```

7.

```
UPDATE STUDENT.LOCATIONS
SET   COUNTRY = 'CANNADA'
WHERE LOCATIONID = 100;
```

8. The transaction succeeded because when a constraint is disabled, it is not checked.

9.

```
ROLLBACK;
```

10.

```
ALTER TABLE STUDENT.LOCATIONS DROP CONSTRAINT
LOCATIONS_COUNTRY_CK;
```

Lab 13-2

2.

```
SELECT CO.CONSTRAINT_NAME,CO.CONSTRAINT_TYPE,
       CO.STATUS,CO.TABLE_NAME,CC.CO
2 FROM   DBA_CONSTRAINTS CO, DBA_CONS_COLUMNS CC
3 WHERE  CC.TABLE_NAME = CO.TABLE_NAME
4 AND    CC.CONSTRAINT_NAME = CO.CONSTRAINT_NAME
5 AND    CO.TABLE_NAME = 'SCHEDULEDCLASSES'
6* AND   CO.OWNER = 'STUDENT'
```

CONSTRAINT_NAME	C STATUS	TABLE_NAME	COLUMN_NAME
SYS_C00965	C ENABLED	SCHEDULEDCLASSES	CLASSID
SYS_C00966	C ENABLED	SCHEDULEDCLASSES	COURSENUMBER

SYS_C00967	C	ENABLED	SCHEDULEDCLASSES	LOCATIONID
SYS_C00968	C	ENABLED	SCHEDULEDCLASSES	CLASSROOMNUMBER
SYS_C00969	C	ENABLED	SCHEDULEDCLASSES	INSTRUCTORID
SYS_C00970	C	ENABLED	SCHEDULEDCLASSES	STARTDATE
SYS_C00971	C	ENABLED	SCHEDULEDCLASSES	DAYSURATION
SYS_C00972	C	ENABLED	SCHEDULEDCLASSES	STATUS
PK_CLASSID	P	ENABLED	SCHEDULEDCLASSES	CLASSID
FK_SCHEDCLASS_COURSENUM	R	ENABLED	SCHEDULEDCLASSES	COURSENUMBER
FK_SCHEDCLASSES_LOCATIONID	R	ENABLED	SCHEDULEDCLASSES	LOCATIONID
CONSTRAINT_NAME	C	STATUS	TABLE_NAME	COLUMN_NAME
FK_SCHEDCLASSES_INSTID	R	ENABLED	SCHEDULEDCLASSES	INSTRUCTORID

3.

```
1 UPDATE STUDENT.SCHEDULEDCLASSES
2 SET   LOCATIONID = 999
3* WHERE CLASSID = 50
```

```
UPDATE STUDENT.SCHEDULEDCLASSES
*
```

```
ERROR at line 1:
ORA-02291: integrity constraint
(STUDENT.FK_SCHEDCLASSES_LOCATIONID) violated - parent key
not found
```

4. The update failed because the FOREIGN KEY constraint was violated. 999 is not a valid value in the LOCATIONS table.

5.

```
ALTER TABLE STUDENT.SCHEDULEDCLASSES MODIFY CONSTRAINT
FK_SCHEDCLASSES_LOCATIONID INITIALLY IMMEDIATE;
```

6.

```
1 UPDATE STUDENT.SCHEDULEDCLASSES
2 SET   LOCATIONID = 999
3* WHERE CLASSID = 50
```

```
UPDATE STUDENT.SCHEDULEDCLASSES
*
```

```
ERROR at line 1:
ORA-02291: integrity constraint
(STUDENT.FK_SCHEDCLASSES_LOCATIONID) violated - parent key
not found
```

7. The Constraint still fails because constraints must be created DEFERRABLE in order to use DEFERRED constraint checking. Also, the constraint state was set to INITIALLY IMMEDIATE, which means that the constraints are not initially deferred. You should use the INITIALLY DEFERRED option to defer constraints until commit by default.

8.

```
ALTER TABLE STUDENT.SCHEDULEDCLASSES DROP CONSTRAINT
FK_SCHEDCLASSES_LOCATIONID;
```

```
ALTER TABLE STUDENT.SCHEDULEDCLASSES ADD
CONSTRAINT FK_SCHEDCLASSES_LOCATIONID
FOREIGN KEY (LOCATIONID)
REFERENCES LOCATIONS(LOCATIONID)
INITIALLY DEFERRED;
```

9.

```
1 UPDATE STUDENT.SCHEDULEDCLASSES
2 SET LOCATIONID = 999
3* WHERE CLASSID = 50
```

10. The update succeeds this time because the constraints are deferred. So even though the constraint is violated, Oracle does not check it until a commit is issued.

11. COMMIT;

```
commit
*
ERROR at line 1:
ORA-02091: transaction rolled back
ORA-02291: integrity constraint
(STUDENT.FK_SCHEDCLASSES_LOCATIONID)
violated -
parent key not found
```

12. The commit fails because the constraint is now checked and it is violated. The statement is rolled back.

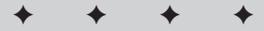
IV

Managing Oracle Data

During your career as a database administrator there will be times when you need to load large amounts of data into your database, move data from one database to another, or simply reorganize data in an existing database to improve performance and provide for more efficient storage. The two chapters in this part of the book deal with all these issues.

In Chapter 14 you will learn how to use direct load inserts and SQL*Loader to load data into an Oracle database. The syntax for SQL*Loader will be described in detail with examples of its use.

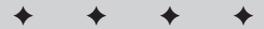
Chapter 15 will introduce you to the reasons why you may want to reorganize data in your database and several methods that can be used to do so. The CREATE TABLE AS syntax will be covered first, followed by the Export and Import utilities. The syntax of each utility and the impact of database character sets and language settings will be outlined as well. Finally, you will learn how to transport entire tablespaces from one database to another and simply attach a whole set of data to another database.



In This Part

Chapter 14
Loading Data

Chapter 15
Reorganizing Data



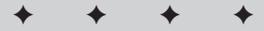
Loading Data

EXAM OBJECTIVES

- ◆ Loading Data
 - Load data using direct-load INSERT
 - Load data into Oracle tables using SQL*Loader:
Conventional Path
 - Load data into Oracle tables using SQL*Loader: Direct Path

14

CHAPTER



CHAPTER PRE-TEST

1. What is the difference between a regular INSERT and a direct-load INSERT?
2. What are some of the advantages of using a direct-path load instead of a conventional load?
3. How can you limit the number of records to be inserted into a table when performing a conventional load? A direct-path load?
4. If you attempt to start two SQL*Loader sessions that need to load data in the same table, what type of load should you perform so that the process completes as quickly as possible?
5. After performing a direct-path load and specifying the UNRECOVERABLE parameter in the control file, what should you do to safeguard your data?
6. When invoking SQL*Loader, what parameter value must you specify?
7. What types of loads ensure that INSERT triggers fire?
8. In what situation would a conventional load be better than a direct-path load?
9. Where and how is the data saved when you add data to a table using a direct-load INSERT?
10. If you created a table on a cluster, what type of INSERT operation can you perform to load a large number of rows into the table from another table in the same database?

As a database administrator you will, at some point in your career, be asked to take data from an external source and load it into an Oracle database, or take data from one or more tables in your current database and INSERT it into another table in the database. There could be many reasons for this, including reorganizing data (covered in the previous chapter), or a migration from another platform to Oracle, or taking data from an OLTP system and moving to a data warehouse, or acquisition of third-party data to be used in analysis, and many others. Whatever the reason, an important ability for a DBA is to be able to take data from outside of Oracle and load it into the database using the SQL*Loader utility provided by Oracle, or take data in existing tables and load it into other tables. The “Oracle8i: Architecture and Administration” exam tests your ability to perform these tasks and understand the process.

Overview of Loading Data into Oracle Databases

Taking data from external sources and loading it into an Oracle database can be accomplished using the SQL*Loader utility or other third-party tools. Adding data from existing tables to another table in the database is simply the process of issuing an INSERT INTO ... SELECT statement. Oracle provides variations of both operations that are more or less efficient.

For an INSERT operation two methods exist: a conventional INSERT that works much the same way as adding a single row into the table many times over, or a direct-load INSERT whose outcome is the same as a regular INSERT but that bypasses the database buffer cache and writes data directly into the datafiles, thereby speeding up performance.

Similarly, using the SQL*Loader utility to add data to the database from external files, you can perform a conventional load, similar to a regular INSERT for each row being loaded, or a direct load that writes the data to be migrated to the Oracle database directly to the datafiles. No matter which method of SQL*Loader you use, it is important to remember that the utility is designed to migrate data from external ASCII datafiles into the Oracle database.

If you need to migrate data from one Oracle database to another, the Import and Export utilities covered in the previous chapter can be used to do this. SQL*Loader cannot be used with export files but only with ASCII datafiles whose format is known and described in the SQL*Loader control files. Furthermore, direct-load INSERTs only work when adding data to a table from another table in the same database. If you need to move data between databases, or re-organize existing data, Import and Export, as well as the methods described in the previous chapter should be employed.

Loading Data Using Direct-Load INSERTs

Objective

Load data using direct-load INSERT

When adding data to a table, you can INSERT one row at a time using the following syntax:

```
INSERT INTO tablename VALUES (expr, expr, ...);
```

This syntax enables you to add one row to the table whose column values are specified in order in parentheses by each of the expressions separated by commas. However, if you want to add a number of rows to the same table from another table, you can also issue the following command:

```
INSERT INTO tablename  
SELECT ...
```

In this syntax, the SELECT specifies any valid subquery that returns data to be added to the table. The subquery must provide values for all columns in the table, or a column list must be specified in the INSERT clause. For example, if you wanted to re-populate the Temp_Instructors table with an updated list of instructors from the Instructors table, you can issue the following command:

```
SQL> TRUNCATE TABLE temp_instructors;
```

```
Table truncated.
```

```
SQL> INSERT INTO temp_instructors  
2 SELECT * FROM Instructors;
```

```
8 rows created.
```

```
SQL> COMMIT;
```

```
Commit complete.
```

```
SQL>
```

The INSERT statement adds new rows to the Temp_Instructors table and updates any indexes that were created on the table. Naturally, any constraints that you may have created on the Temp_Instructors table will also be enforced, so if you want to reload existing data, you also need to TRUNCATE the table first. If the table does not exist, you can both create the table and load the data using the CREATE TABLE ... AS SELECT ... syntax outlined in Chapter 11, as in this example:

```
SQL> CREATE TABLE temp_instructors AS
2  SELECT * FROM Instructors;

Table created.

SQL> SELECT COUNT(*) FROM temp_instructors;

  COUNT(*)
-----
         8

SQL>
```

As you can see, the same number of rows were in the Temp_Instructors table as after the INSERT statement in the previous example.

Up to this point, you have not performed a direct-load INSERT. In fact, you have performed conventional INSERT statements that use the database buffer cache to load the block into memory and populate it with data. In performing these INSERT statements the Server process gets the freelist and determines the next block to which data will be added, brings that block into the database buffer cache, and adds the rows. For eight rows, this may not be such a big deal; for eight million this requires a long time and a lot of work on the part of the Server process to complete. It is the time issue and the work required by the Server process that is addressed by a direct-load INSERT.

A direct-load INSERT differs from a conventional INSERT in that an optimizer hint must be specified in the INSERT portion of the command to indicate that data will be appended to the segment on disk and will not follow the conventional path. When data is appended to the end of the segment, Oracle does not use the database buffer cache and bring each block into memory as the INSERT operation takes place, but rather writes the rows to be added to the table directly to the datafiles. These rows are added above the high-water mark of the table segment and appended to the table when the operation completes.



In the Real World

Oracle provides you with the capability to change the way commands are executed and the execution plans that are used by means of optimizer hints. Using optimizer hints is quite common for developers as this enables them to program applications that have good performance, or at least help to tweak the performance of applications. DBAs can also make use of many optimizer hints in optimizing database performance and increasing the speed of execution of certain commands. Loading data into a table using direct-load INSERTs is just one example of this. For a full discussion on optimizer hints, please refer to the *Oracle8i Designing and Tuning for Performance* manual in the Oracle documentation set.



Cross-Reference

For information on the high-water mark and its meaning regarding allocation of space to tables, please refer to Chapter 11.

To specify that the action to be performed is a direct-load INSERT, you need to add the APPEND hint to the INSERT statement. For example, if you want to reload the list of instructors from the Instructors table into Temp_Instructors using a direct-load INSERT, you issue the following command:

```
SQL> INSERT /*+APPEND */ INTO temp_instructors
      2 SELECT * FROM Instructors;
```

8 rows created.

```
SQL> COMMIT;
```

Commit complete.

```
SQL>
```



Caution

When including an optimizer hint in your SQL statement, it is very important that the hint be prefixed with the three-character combination of /*+. If the + symbol is not together with the /* characters, Oracle treats the text after the /* and before the / as a comment and ignores it in the execution. The /*+ three-character combination tells Oracle that what follows is one or more optimizer hints that should be parsed and applied.

NOLOGGING option

When performing direct-load INSERTs, Oracle, by default, logs the fact that data was added to the table in the redo log files, thus providing you with full recoverability in case the datafile becomes damaged prior to a backup having taken place. This, however, causes the operation to take longer than if the changes were not logged and can mean a significant difference in the execution time for large volumes of data. If you routinely backup the datafiles of the tables on which the data was loaded right after the load operation, you can forgo the redo activity by specifying the NOLOGGING option in the INSERT command. For example, to not generate redo log activity when the Temp_Instructors has data loaded, you would change the INSERT command in the previous example to the following:

```
SQL> INSERT /*+APPEND */ INTO temp_instructors NOLOGGING
      2 SELECT * FROM Instructors;
```

8 rows created.

```
SQL> COMMIT;
```

Commit complete.

```
SQL>
```

Oracle chooses to use either the NOLOGGING or LOGGING option of the INSERT command, depending on the table's default setting for this option. If the table into which data is being inserted was created with the LOGGING option, which is the

default, the direct-load INSERT logs all changes to the redo log files. However, if you created the table with the NOLOGGING attribute specified, then Oracle does not generate redo log entries for each row added, unless you specify LOGGING in the command itself. In other words, whether LOGGING or NOLOGGING is the default for a direct-load INSERT depends on the table setting. If no setting was specified when the table was created, the Oracle default is LOGGING.

Even though you can specify that you do not want the rows added to the table to be logged to the redo log files, Oracle always logs certain activity to ensure the structural integrity of the table. This includes the allocation of new extents to the table to support the additional rows; that is, if the existing extents on the table do not have sufficient space to store the data, Oracle will allocate additional extents to hold the data and, when the direct-load INSERT completes, add these extents to the table segment.

You should specify NOLOGGING or LOGGING explicitly when performing a direct-load INSERT. This is to ensure that Oracle behaves as you intend it to and generates redo or not, as you expect. It is possible to change the LOGGING or NOLOGGING attribute for a table using an ALTER TABLE command, which means that unless the option was explicitly specified for the direct-load INSERT, it is possible for the same command to behave differently at two invocations.

Parallel direct-load INSERTS

If you want to speed up a direct load INSERT even further, and are loading a large amount of data, you can invoke additional processes to perform the action. This works particularly well when data is being added to a partitioned table since Oracle allocates parallel query processes to perform both the INSERT and the SELECT speeding up both sides of the operation.

In order to perform a parallel direct-load INSERT, you need to ensure that you have enabled query rewrite and parallel DML for the session by issuing the following command:

```
SQL> ALTER SESSION SET QUERY_REWRITE_ENABLED=TRUE;
```

```
Session altered.
```

```
SQL> ALTER SESSION ENABLE PARALLEL DML;
```

```
Session altered.
```

```
SQL>
```

Enabling parallel DML must be executed outside of a transaction; otherwise, you will receive the following error:

```
SQL> ALTER SESSION ENABLE PARALLEL DML;
```

```
ERROR:
```

```
ORA-12841: Cannot alter the session parallel DML state within a transaction
```

Enabling parallel DML does not necessarily mean that DML operations will automatically be parallelized. In order to ensure that additional processes are launched to perform the direct-load INSERT, you need to ensure that either the table was created with a PARALLEL clause indicating the default degree of parallelism, or that a PARALLEL optimizer hint is included in the INSERT statement, as follows:

```
SQL> CREATE TABLE temp_students AS
  2  SELECT * FROM Students WHERE 0=1;

Table created.

SQL> INSERT /*+PARALLEL(temp_students,2) */
  2  INTO temp_students NOLOGGING
  3  SELECT * FROM Students;

11 rows created.

SQL> COMMIT;

Commit complete.

SQL>
```

It is possible to specify a parallel hint for the INSERT statement as well as for the SELECT statement in the subquery. Oracle will choose only one of the statement's parallel hints to determine the overall degree of parallelism for the entire statement. The rules for doing so are applied in the following priority:

1. If a PARALLEL hint is specified in the INSERT statement, the degree of parallelism specified there is used for the entire parallel direct-load INSERT. If not, then...
2. If a default degree of parallelism is specified for the table being inserted into, then the table's default degree of parallelism is used for the entire parallel direct-load INSERT. If not, then...
3. The maximum query directive is used, which means that the table with the highest degree of parallelism has its degree of parallelism (either hint or default degree for the table) applied.

For example, if you issued the following statement, the degree of parallelism for the entire statement would be 2, since this is what was specified in the INSERT statement:

```
SQL> INSERT /*+PARALLEL(temp_students,2) */
  2  INTO temp_students NOLOGGING
  3  SELECT /*+PARALLEL (Students,3) */ * FROM Students;
```

As a general rule of thumb, you should explicitly specify the degree of parallelism for the INSERT operation in an optimizer hint in the statement itself. Doing this ensures that the degree of parallelism is what you expect and want, assuming all other session parameters have been properly applied.

In the case where the proper session parameters have not been modified Oracle will *not* generate an error when a parallel hint is included in the direct-load INSERT. In fact, it will simply process the statement, usually serially unless the APPEND hint was also specified in the INSERT statement, and return no error. Your only hint that something has gone wrong will be when the statement takes much longer than expected. For this reason, always include both the PARALLEL and APPEND hints in the statement, while also using the appropriate ALTER SESSION parameters, as follows:

```
SQL> ALTER SESSION SET QUERY_REWRITE_ENABLED=TRUE;

Session altered.

SQL> ALTER SESSION ENABLE PARALLEL DML;

Session altered.

SQL> INSERT /*+PARALLEL(temp_students,2) APPEND */
  2 INTO temp_students NOLOGGING
  3 SELECT * FROM Students;

11 rows created.

SQL> COMMIT;

Commit complete.

SQL>
```

You have another option to help ensure that DML statements, including INSERT statements, are parallelized. You can also force Oracle to perform parallel DML for your session. To do so, issue the following command:

```
SQL> ALTER SESSION FORCE PARALLEL DML;

Session altered.

SQL>
```

Restrictions on parallel direct load INSERTs

One of the interesting side effects of performing parallel DML is that you can issue parallel DML statements in a single transaction for as many tables as you like. However, once a table has been modified in parallel within a transaction, no queries or further DML are allowed on the table. This is because parallel DML uses many processes making changes to the table at the same time and further changes may not be able to locate the rollback segment information for any rows already changed. Coordinating this activity can become burdensome when large amounts of data are changed, so Oracle prohibits it. Attempting to further modify or query a table in the same transaction where you performed a parallel direct-load INSERT on the table results in errors as shown here:

```

SQL> INSERT /*+PARALLEL(temp_students,4) */ INTO temp_students
2 SELECT * FROM Students;

11 rows created.

SQL> UPDATE temp_students
2 SET Country='United States of America'
3 WHERE Country='USA';
UPDATE temp_students
*
ERROR at line 1:
ORA-12838: cannot read/modify an object after modifying it in parallel

SQL> SELECT COUNT(*) FROM temp_students;
SELECT COUNT(*) FROM temp_students
*
ERROR at line 1:
ORA-12838: cannot read/modify an object after modifying it in parallel

SQL>

```

Other restrictions on parallel DML in general that also apply to parallel direct-load INSERTs include:

- ♦ If the Oracle initialization parameter `ROW_LOCKING = INTENT` is configured, the INSERT statements are not parallelized, regardless of the hints specified in the statement or the default degree of parallelism for any of the affected tables.
- ♦ Parallel direct-load INSERTs, and other parallel DML statements, will not fire any triggers on the table being modified. You should ensure that all trigger conditions are met prior to issuing the INSERT statement.
- ♦ Parallel direct-load INSERTs will not be supported in table replication. Any data added in parallel will not be replicated because replication uses triggers to move the data across and parallel direct-load INSERTs fire no triggers.
- ♦ Parallel direct-load INSERTs do not support any referential integrity constraints. This means that if a FOREIGN KEY is defined on a table in which you are inserting data, the FOREIGN KEY constraint will not be verified.
- ♦ Parallel DML cannot occur on tables with object columns or LOB columns, or on index-organized tables (IOTs). If a parallel direct-load INSERT is attempted on a table with a LOB column or on an IOT, the operation will be performed serially.
- ♦ You cannot perform a parallel direct-load INSERT on a table created on a cluster.

If your statement violates any of these rules for parallel DML, as mentioned previously, Oracle will not return an error but simply performs the operation serially, either as a direct-load INSERT or as a conventional INSERT operation.

Pros and cons of direct-load INSERTs

Direct-load INSERTs, whether parallelized or not, provide significant performance benefits to the DBA. Here are some of the reasons why you should use direct-load INSERTs when loading data from one or more tables or views in your database to another table in the same database:

- ♦ When a direct-load INSERT is taking place, it does not restrict other users from performing queries or other DML on existing table data. As far as Oracle is concerned, until the direct-load INSERT completes and the additional extents are added to the table above the high-water mark, the data is not part of the table and other users should not be restricted from modifying existing data.
- ♦ When parallelized, direct-load INSERTs can take advantage of additional processing power on the computer and perform quicker.
- ♦ When the NOLOGGING option is specified, the amount of activity for adding the data to the target table is further reduced, thereby reducing the workload on the database and server.
- ♦ Direct-load INSERTs update indexes at the end of the process, causing less activity on the index segments and completing the index updates more quickly than if the data were added serially.
- ♦ It's faster. This is about as simply as it can be stated.

While there are good reasons for using direct-load INSERTs, there are some drawbacks:

- ♦ If a direct-load INSERT was taking place when the instance failed, it must be restarted. This may sound like a drawback, but remember that any transaction that is not committed when the instance crashes also has to be rolled back. Direct-load INSERTs actually require less work on instance recovery than conventional INSERTs, so maybe this point should be in the benefits list.
- ♦ If you have a lot of empty space below the high-water mark for the table, this space will not be used by the direct-load INSERT and will slow down full table scans on the table. Since a direct-load INSERT appends data above the high-water mark and then resets the high-water mark to the last block with appended data, this can cause performance problems because a full table scan scans all blocks of a segment from the header block to the high-water mark. If many blocks below the high-water mark are empty, this wastes time.
- ♦ Parallel direct-load INSERTs do not enforce integrity constraints. Direct-load INSERTs that are not parallelized enforce integrity constraints.
- ♦ Parallel direct-load INSERTs use additional processes on the server, which could increase server load. You should not use parallel direct-load INSERTs on servers with a single processor or on multi-processor systems that are busy servicing other applications in addition to your Oracle database.

In deciding whether to use direct-load INSERTs, you should know that in most cases the benefits outweigh the drawbacks. Besides, are you going to be loading large amounts of data into tables on the database during periods of high database activity, or when the database is fairly quiescent? If the former is true, you may want to rethink your answer.

Loading Data Using SQL*Loader

While direct-load INSERTs are great for getting data from one or more tables in a database into a table in the same database, the SQL*Loader utility is used to load data from external sources into an Oracle database. SQL*Loader is a command-line utility that takes a number of parameters and uses at least one file to control its behavior and contain the data. The capabilities of SQL*Loader have been evolving over time and the version provided with Oracle8i has new and better functionality than its predecessors. Furthermore, the syntax that can be specified in the control file (more on this file later) to control how data is loaded into Oracle is quite sophisticated.

Key features of SQL*Loader include:

- ♦ The ability to load data from one or more external input files.
- ♦ The ability to load data into one or more tables from an input file.
- ♦ You can load data into the database using either conventional or direct-path loads. Conventional loads operate much the same way as INSERT statements and use the database buffer cache to store the data before the blocks are saved in the datafile. Direct-path loads append data directly to the datafile above the high-water mark for the segment.
- ♦ The ability to combine data from several records in the input file into a single row in the database.
- ♦ You can load data from several different sources including disk, tape, or named pipe. It is possible to load data from any combination of these sources in a single load operation.
- ♦ The input files supported by SQL*Loader can include ASCII, binary, date, packed decimal, or zoned decimal data.
- ♦ The input fields supported can be of a fixed or variable length.
- ♦ Variable length fields can have a prefix set of characters indicating the length of the remaining data thereby ensuring only the relevant information is added, and also allowing characters such as carriage returns to be treated as data.
- ♦ You have the capability to specify a string as a field or record terminator. This is useful today as comma-separated files are rare and many organizations have field or record separators that are several bytes long.
- ♦ If a record already exists in the database, you can replace its column values with data found in the input file. If the data does not exist, you can choose to append it to the table. This check can be performed during the load operation.

- ♦ You can “massage” the data prior to storing in the database with SQL functions and other conditional logic.
- ♦ You can use conditional logic to change column values based upon the data in other fields in the input file.

While other third-party tools exist to load data into Oracle from external sources, the flexibility and features provided by SQL*Loader often make it the choice for Oracle DBAs. With the ability to invoke SQL*Loader using Oracle Enterprise Manager and configure its control file using a graphical user interface, it becomes even more useful.

Files used by SQL*Loader

When invoking SQL*Loader either using the command-line interface or through Enterprise Manager, you can specify one or more files for it to use. Some, like the SQL*Loader control file (not to be confused with the database control file), are required while most others are either optional or have defaults. Table 14-1 lists the files that SQL*Loader can use.

Table 14-1
Files Used by SQL*Loader

<i>File</i>	<i>Description</i>
Control File	The control file provides instructions and information on the data to be loaded, where the data can be found (in the control file or in a datafile), the format of the data, and more. The control file is required for SQL*Loader operation and its name must be specified when the utility is invoked.
Datafiles	Datafiles contain the data that will be loaded into the Oracle database in the format specified in the control file. More than one datafile can be used for a single load. A datafile is not required for a load if the data to be loaded is contained in the control file.
Parameter File	SQL*Loader allows you to place the command-line parameters and their values in a text file with a single parameter per line. When invoking SQL*Loader, you can indicate that you have the parameters in a parameter file with the PARFILE= command-line option. If you do not use a parameter file, you must specify all command-line options when you invoke SQL*Loader.

If a parameter is specified both in the parameter file and on the command-line when SQL*Loader is invoked, the command-line version of the parameter has precedence over any value specified in the parameter file.

Continued

Table 14-1 (continued)

<i>File</i>	<i>Description</i>
Log File	<p>A log file is automatically created when a load is started and contains a record of the actions that took place during the load. It has a specific structure (outlined later in this chapter).</p> <p>The default filename for the log file is the same name as the control file with a “.log” extension.</p>
Bad File	<p>SQL*Loader can create a bad file for those rows that are rejected during the load operation. Rejected records are those in the datafile that were not loaded into the database because they did not meet the criteria specified in the control file or were rejected by the Oracle server because they violated a primary key or other constraint.</p> <p>The default filename for the bad file is the same name as the control file with a .bad extension.</p>
Discard File	<p>SQL*Loader can create a discard file to store those records that were found in the datafiles that did not meet the selection criteria specified in the control file.</p>

A number of the files that SQL*Loader uses require further explanation and a little more detail. These include the control file and the log file.

The control file

A SQL*Loader load cannot be invoked without a control file. The control file, which is not the same as the database control file specified by the CONTROL_FILES Oracle initialization parameter, tells SQL*Loader what to do and how the data to be loaded is formatted, where it’s located, and what the load criteria are.



The “Oracle8i: Architecture and Administration” exam does not directly test your knowledge of the control file syntax of SQL*Loader so there is no need to attempt to memorize that information. While the test requires that you be familiar with the basic syntax for configuring delimiters and specifying the location of the datafile, the questions you are more likely to be asked deal with the files used by SQL*Loader, the modes of operation, and how to perform efficient loads. This notwithstanding, you should still review the information in the *Oracle8i Utilities* manual concerning SQL*Loader.

The control file contains what are generically called *load instructions*. These load instructions are specified using a syntax specific to SQL*Loader. The instructions and other information that can be specified in the control file include:

- ♦ Names of the datafiles to be used by the load. The names are specified using the INFILE clause in the control file.

- ♦ The makeup of a single logical record in the datafiles. Control file syntax can use such clauses as `CONCATENATE` or `CONTINUEIF` to perform condition logic and massage data during the load operation.
- ♦ The names of fields in the datafiles including their starting point and length, as well as their datatypes. Mapping of fields in the datafile to table columns are also specified here.
- ♦ The names of the tables into which the data will be loaded. The `INTO TABLE` clause of the control file syntax enables you to specify this information.
- ♦ Instructions on how the data will be loaded into the tables. Because `SQL*Loader` provides great flexibility here, you are allowed to specify whether the data should be appended to the table, replace existing rows in the table with new data during the load, or loaded into an empty table with no previous data. As mentioned earlier, data can be loaded into multiple tables at the same time and these options can be specified for each table individually.
- ♦ If you want to skip records for any of the tables into which data is being loaded, you can use the `CONTINUE_LOAD` command to start the load from a particular point and indicate this in the control file.
- ♦ A `WHEN` clause allows you to determine whether or not a record should be loaded into a table. This clause allows you to have a datafile with more records than you will load and gives you the ability to load only the ones meeting your criteria.
- ♦ Instructions on generating column values in the target table for columns for which there is no data in the datafiles, or for which the data does not make sense and you want to ensure that a value is provided during the load. You are able to make use of `RECNUM` and `SYSDATE` clauses, as well as SQL functions to accomplish this.
- ♦ You can massage data in columns that are loaded by trimming or padding data as needed, changing `NULLs` to zeros or empty strings, or vice versa.
- ♦ The control file can include comments to make it easier to understand what will take place. All comments need to be prefixed with a `--` (double dash) string.
- ♦ Additional load parameters can be specified with the `OPTIONS` clause to ensure that the load satisfies the conditions of its operations. Specifying this clause is the same as specifying the parameters on the command line, and includes options such as whether a load is to run in parallel or serially, or the number of rows to load before issuing a commit.
- ♦ If you are performing a direct-path load, you can specify additional options, including whether to enable constraints that were disabled at the start of the load when the load completes (the `REENABLE` option), whether the data is pre-sorted so index creation is quicker (`SORTED_INDEXES`), or whether to suppress generation of redo records (`UNRECOVERABLE`) for the tables into which the data is being loaded if the table currently has the `LOGGING` attribute set. Many other options are also available.

The control file can contain data within itself, in which case the “INFILE **” instruction is in the control file, or you may specify the name of the datafile in the control file. If the “INFILE **” instruction is used, all data at the end of the control file will be loaded. An example of a control file with the data in the same file is as follows:

```
LOAD DATA
INFILE *
INTO TABLE student.Instructors
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '''
(InstructorId, LastName, FirstName, InstructorType, EMail, Comments)
BEGINDATA
500,Bersinic,Damir, ORACLE,"damir@bradsys.org","Very versatile."
501,Giles,Stephen,ORACLE,"sgiles@bradsys.org","Good rapport with students"
502,Ross,Todd,ORACLE,"tross@bradsys.org","Teaches Unix classes as well"
503,Sabinin,Yury,ORACLE,"ysabinin@bradsys.org","Works with DB2"
504,Brown,Myles,Prog,"mbrown@bradsys.org","Great programming instructor"
```

An example of a more complex control file that uses an external datafile is as follows:

```
LOAD DATA
INFILE 'studentdata.dat'
APPEND
INTO TABLE STUDENT.Students
  WHEN (57)='.'
  TRAILING NULLCOLS
(StudentNumber   POSITION(1:4)   INTEGER EXTERNAL(4)
 LastName        POSITION(7:24)  CHAR TERMINATED BY WHITESPACE
 EMail           POSITION(28:48) CHAR TERMINATED BY WHITESPACE
                 NULLIF EMail=BLANKS,
 FirstName       POSITION(50:68) CHAR TERMINATED BY WHITESPACE
 HomePhone       POSITION(70:80) CHAR TERMINATED BY WHITESPACE,
 Comments        POSITION(84:200) CHAR TERMINATED BY WHITESPACE
```



The For a complete listing of the control file syntax and available commands, please refer to the *Oracle8i Utilities* manual in the Oracle8i documentation set.

The log file

As the load progresses, SQL*Loader always places information about the progress of the load, as well as information about files used and the outcome, in the log file. The log file is mandatory and is always created in the current directory from where SQL*Loader is invoked. The log file, by default, has the same name as the control file but with a .log extension.

The SQL*Loader log file has a defined structure. It is made up of the following parts, in order:

- ♦ **Log File Header.** The header contains information on when the load took place and the version of the software that was used to perform the operation.
- ♦ **Global Information.** This section of the log file includes the names of all the input files used in the load operation as well as any command-line arguments or parameter file values used to perform the load.
- ♦ **Table Information.** Here you will find which table's data was loaded into, the load conditions, and the method used (conventional or direct-path load).
- ♦ **Field and Column Information.** Mappings of fields in the datafile or control file to columns in the table in the database are listed in this part of the log file.
- ♦ **Records Processed.** This part of the log file shows the number of records processed during the load, including the number of rows rejected and/or discarded because of conditions in the control file, or Oracle server errors.
- ♦ **Table Load Information.** For each table into which data was loaded, the log file provides information on the number of records loaded as well as the number of records rejected due to data errors or constraint violations. It also provides the number of records discarded for each table, as well as the number of records whose fields were all NULL and the load was not attempted.
- ♦ **Summary Statistics.** The last section of the log file provides statistical information on the operation, including elapsed time, memory used, CPU time used, and the start and end time of the load.

An example of the output that can be found in the log file is shown here:

```
SQL*Loader: Release 8.1.7.0.0 - Production on Tue Jun 12 00:16:16 2001
```

```
(c) Copyright 2000 Oracle Corporation. All rights reserved.
```

```
Control File:   instructors.ctl  
Datafile:      instructors.ctl  
Bad File:      instructors.bad  
Discard File:  none specified
```

```
(Allow all discards)
```

```
Number to load: ALL  
Number to skip: 0  
Errors allowed: 50  
Bind array:     64 rows, maximum of 65536 bytes  
Continuation:   none specified
```

Path used: Conventional

Table STUDENT.INSTRUCTORS, loaded from every logical record.
Insert option in effect for this table: INSERT

Column Name	Position	Len	Term	Encl	Datatype
INSTRUCTORID	FIRST	*	,	0(")	CHARACTER
LASTNAME	NEXT	*	,	0(")	CHARACTER
FIRSTNAME	NEXT	*	,	0(")	CHARACTER
INSTRUCTORTYPE	NEXT	*	,	0(")	CHARACTER
EMAIL	NEXT	*	,	0(")	CHARACTER
COMMENTS	NEXT	*	,	0(")	CHARACTER

Table STUDENT.TEMPINSTRUCTORS:
5 Rows successfully loaded.
0 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

Space allocated for bind array: 65016 bytes(42 rows)
Space allocated for memory besides bind array: 0 bytes

Total logical records skipped: 0
Total logical records read: 5
Total logical records rejected: 0
Total logical records discarded: 0

Run began on Tue Jun 12 00:16:16 2001
Run ended on Tue Jun 12 00:16:16 2001

Elapsed time was: 00:00:00.12
CPU time was: 00:00:00.05

Invoking SQL*Loader from the command line

Objective

Load data into Oracle tables using SQL*Loader: Conventional Path

Load data into Oracle tables using SQL*Loader: Direct Path

As mentioned previously, SQL*Loader is normally invoked from the command line and passed one or more parameters. The syntax for invoking SQL*Loader from the command line is the same on all platforms where Oracle is available, as follows:

```
sqlldr [parameter=value [,parameter=value] [, ...]]
```

The list of valid parameters and their default values is outlined in Table 14-2.

Table 14-2
SQL*Loader Command Line Parameters

<i>Parameter</i>	<i>Description</i>
USERID	<p>The Oracle username and password to use for connecting to the target database. This parameter can be specified using the normal username/password@instance syntax. If the instance is omitted, the ORACLE_SID parameter must be specified to connect to an instance on the local machine. If the password parameter is not specified, you will be prompted to provide it when SQL*Loader starts.</p> <p>This parameter has no default.</p>
CONTROL	<p>The name of the control file to be used for the load. This parameter is required.</p>
LOG	<p>The name of the log file to be created. The default is the same filename as the control file with a .log extension. The file is placed in the current directory if a pathname is not included in the filename.</p>
BAD	<p>The name of the file that will be used to store records rejected by the server or SQL*Loader. The records in the “bad” file have the exact same structure as those in the datafile.</p> <p>The default name for the “bad” file is the same as the control file name with a .bad extension.</p>
DATA	<p>The names of the datafiles that will be used for the load. There is no default for this parameter.</p>
DISCARD	<p>The name of the file that will be used to store records that were discarded by SQL*Loader and not loaded in the database. The records in the discard file will have the same structure as those in the datafile.</p> <p>Specifying a discard file can be useful in determining what the records that did not make it into the database look like, thereby allowing you to modify the control file in case they should not have been discarded.</p>
DISCARDMAX	<p>Specifies the maximum number of records that can be discarded by SQL*Loader before the load is terminated. This is essentially a safety feature to ensure that, in case a wrong datafile was specified, the load does not continue for a long time and populate a table with data not meant for it.</p> <p>The default for DISCARDMAX is unlimited – in other words no maximum is configured allowing SQL*Loader to process all records in the datafiles.</p>

Continued

Table 14-2 (continued)

<i>Parameter</i>	<i>Description</i>
SKIP	<p>The number of records in the datafiles to skip before actually starting the load into the database. This parameter is useful when a load has been interrupted and you do not want to start from the beginning, but only at the point where the load was interrupted. By reviewing the log file and determining how many records were committed to the database, you can easily determine the value to use.</p> <p>The default for SKIP is 0, which means that all records will be loaded into the database.</p>
LOAD	<p>Determines the number of records to load. The value specified for the LOAD command will be applied after the number of records specified by SKIP have been skipped. In other words, after SKIP number of records have been processed, the counter for LOAD will start and the number of records to be loaded will stop when the value specified by LOAD is exceeded.</p> <p>The default for LOAD is all records in the datafiles.</p>
ERRORS	<p>This parameter specifies the maximum number of bad records that are allowed to be processed before the load terminates with an error condition. This is useful to limit a load taking CPU and other resources processing data that never reaches the database.</p> <p>The default for ERRORS is unlimited – in other words, process all records.</p>
ROWS	<p>For a conventional load, specifies the number of rows to be loaded into the database before issuing a COMMIT to save the changes in the database. For a direct-path load, this parameter indicates the number of rows that will be saved to the datafile in a single datasave operation.</p> <p>The default for ROWS for a conventional load is all records to be loaded, i.e. perform only a single COMMIT at the end of the entire load operation. This can be problematic if the rollback segment that is being used by the load does not have sufficient space to hold the before image of all rows being loaded. Setting ROWS to a value other than the default also allows you to re-start failed loads using the SKIP parameter to start at the point after the last COMMIT.</p>
BINDSIZE	<p>Indicates the amount of memory to be used by SQL*Loader to build an array of rows to be loaded into the database for conventional loads. The value specified for BINDSIZE must be at least large enough to hold a single row.</p> <p>If the ROWS parameter is also specified, SQL*Loader will build an array up to the value specified by the ROWS parameter, subject to the maximum memory allowed as indicated by BINDSIZE. In other words, BINDSIZE limits the amount of data transferred to the database in one operation, while ROWS limits the number of rows that will be committed to the database at one time.</p> <p>The default for BINDSIZE is operating system dependent.</p>

<i>Parameter</i>	<i>Description</i>
DIRECT	This parameter, whose possible values are TRUE or FALSE, determines whether SQL*Loader will perform a conventional or direct-path load. The default value is FALSE, which means that a conventional load will be performed.
PARFILE	Specifies the name of a text file that contains valid SQL*Loader parameter/value pairs, one per line. If a value for a parameter is specified both on the command line and in the parameter file, the command line value will override any settings in the PARFILE.
PARALLEL	<p>Indicates that the load will take place in parallel. Parallel loads can only take place using the direct-path method, which means that the DIRECT=TRUE parameter must also be specified. The default is to perform a serial load.</p> <p>Specifying PARALLEL=TRUE on the command line allows more than one instance of SQL*Loader to be active at the same time against the same table while performing a direct-path load. Attempting to load data into the same table without specifying the PARALLEL parameter will generate an error.</p>
FILE	<p>For a parallel direct-path load, specifies the name of the file in which the temporary segment will be created prior to being moved to the datafile where the table exists. The datafile specified by the FILE parameter must belong to the same tablespace as the table or partition into which the data is being loaded.</p> <p>The FILE parameter and parallel direct-path loads are useful for loading data into partitioned tables as more than one process can work on the table or partition at the same time. The end result of the operation is faster processing of the load.</p>

If you invoke SQL*Loader without specifying any parameters, you will bring it up in interactive mode and be prompted for the minimum number of parameters, such as the username and password for the Oracle database into which data will be loaded, as well as the name of the control file. All other parameters should be included in the control file; otherwise, SQL*Loader will use default values for any parameter not specified.

Invoking SQL*Loader using Oracle Enterprise Manager

SQL*Loader can also be invoked by using Oracle Enterprise Manager. In order to do so, follow these steps:

Invoking SQL*Loader Without Parameter Names

It is possible to invoke SQL*Loader without specifying parameter names but only their values, as in this example:

```
C:\CERTDB> sqlldr student/oracle instructorsctl instructors.log
```

In this case, SQL*Loader always assumes that the first value passed is the equivalent of the USERID parameter, the second is the equivalent of the CONTROL parameter, and the third value passed is the same as the LOG parameter. Other values passed must be in the order presented in Table 14-2, otherwise SQL*Loader may generate errors.

While this syntax works, and you can even pass the first few values without parameter names and the rest with the parameter=value syntax, this is generally not recommended. Using this syntax may cause confusion when reviewing the code or, if the order of parameters changes with future releases of SQL*Loader, cause scripts to fail. It is generally recommended that all parameters be specified on the command line with the parameter=value syntax, or be included in the control file, or in a parameter file that is indicated by the PARFILE parameter when SQL*Loader is invoked.

STEP BY STEP: Loading Data Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database you are going to perform the load into.
2. Right-click on the name of the database that you will load the data into and choose Data Management ⇨ Load from the menu, as shown in Figure 14-1. This invokes the Load Wizard, as shown in Figure 14-2.



If you got an error at this point, it probably indicates that you have not set preferred credentials for the node. Use the System menu, Preferences option to set preferred credentials for the node on which you are performing the load (that is, where the database resides).

If the Data Management option does not appear when you right-click on the instance name in the Enterprise Manager console, you have invoked DBA Studio instead of Enterprise Manager Console. In order to perform a load, you must invoke Enterprise Manager and connect to a Management Server.

3. Click Next to bypass the introductory screen of the Load Wizard. Enter the name of the control file to be used for the load when prompted, as shown in Figure 14-3.

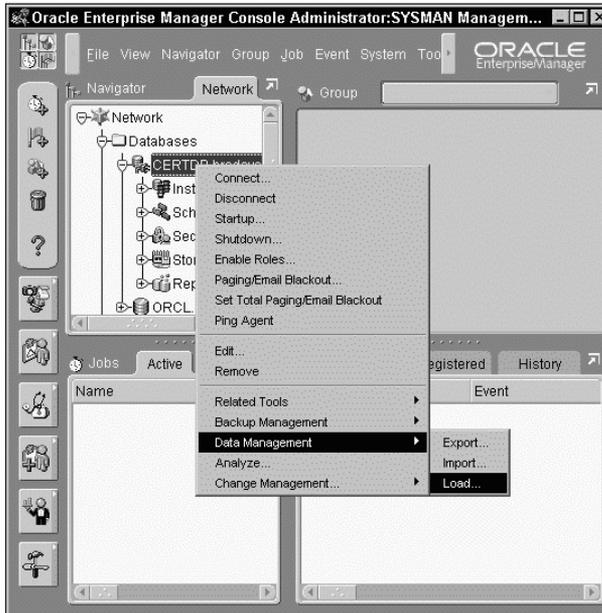


Figure 14-1: Use Data Management → Load to start the Load Wizard.



Figure 14-2: The Load Wizard starts and presents an introduction screen.

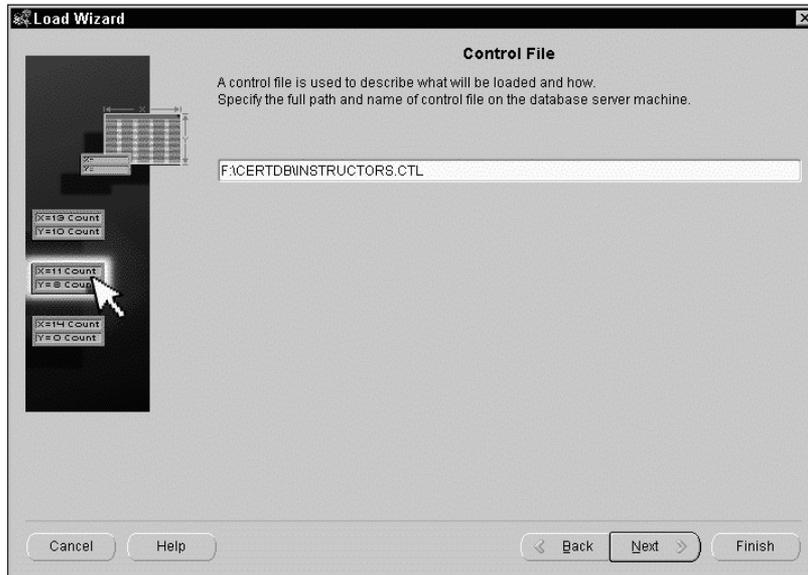


Figure 14-3: A control file must be specified in the Load Wizard.

4. As you can see in Figure 14-4, after clicking Next, you are prompted to let the Load Wizard know whether the datafile to be used for the load is specified in the control file (or the data is in the control file), or you can indicate the name of the datafile on the machine where the database is located. If the file is not on the same machine as where the database is located, you will need to ensure that a network path exists to the location where the file is physically located.
5. Click Next and you will be prompted to specify the type of load to be performed: conventional, direct-path, or parallel direct-path, as shown in Figure 14-5. Choosing Advanced on this screen also allows you to configure elements of the load. This screen may not be available for all load types and will not be available if the data being loaded is completely contained in the control file.
6. The Schedule screen is shown when you click Next in the Wizard, as presented in Figure 14-6. You can schedule to run the load immediately, or have it take place once, on a regular interval, weekly, or monthly. Determine which scheduling options suit your requirements best and click Next.
7. The Job Information screen, shown in Figure 14-6, provides information of the job to be scheduled by the Load Wizard. You have the option to submit the job (that is, have it run as scheduled), add it to the library of jobs in Oracle Enterprise Manager but not submit it, or both. Choose the option that best suits your requirements and click Finish.

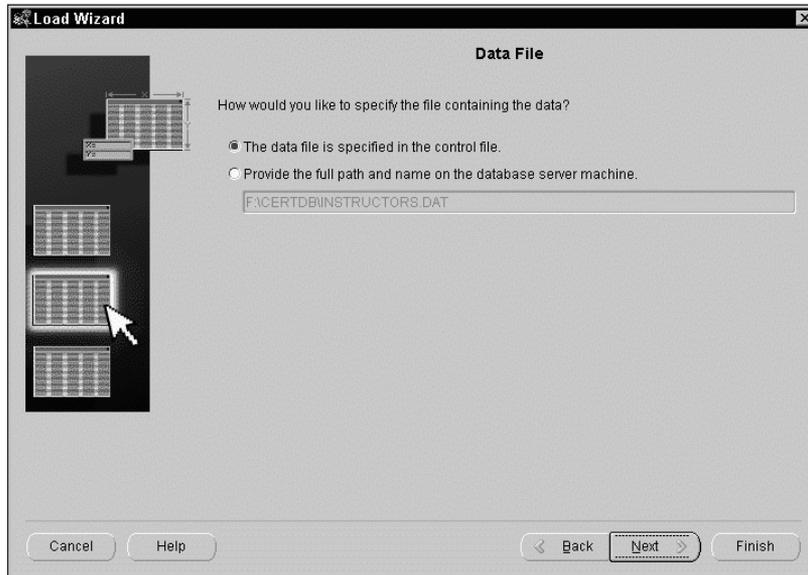


Figure 14-4: The Load Wizard will need to know where the datafile containing records to be loaded is located.

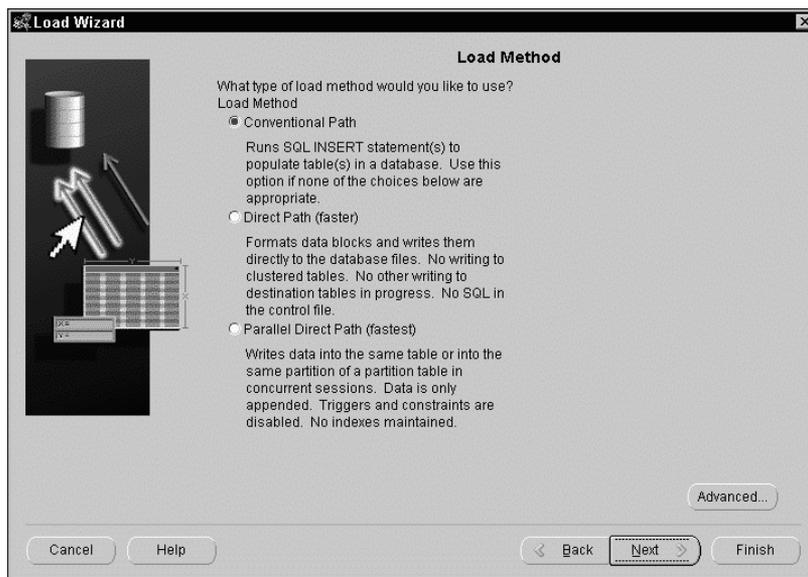


Figure 14-5: The Load Wizard prompts you to indicate the type of load being performed.

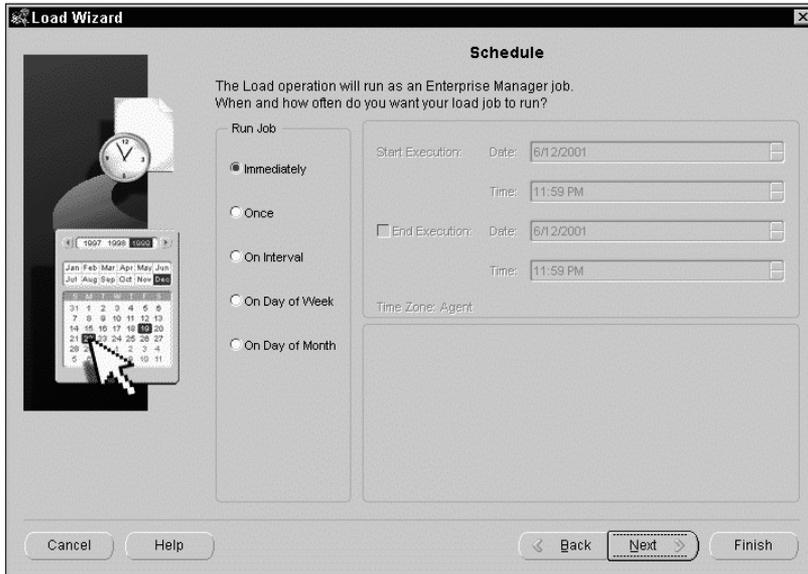


Figure 14-6: The Load Wizard Schedule screen allows you to schedule the load to run once or on a regular interval.

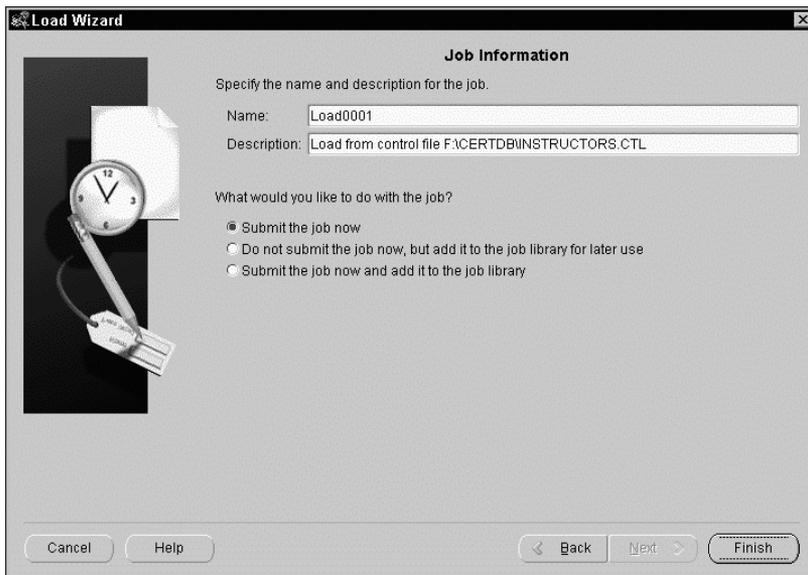


Figure 14-7: The Load Wizard Job Information screen provides a summary of the job to be scheduled and allows you to submit the job, save it to the job library, or both.

8. The final screen includes a summary of information on all of the Load Wizard actions. Click OK to confirm the changes, or Cancel to abandon your work.



Figure 14-8: The Load Wizard Summary screen allows you to confirm your choices or cancel the operation.

Comparing conventional and direct-path loads

As you have already found out, SQL*Loader enables you to make use of conventional, direct-path, and parallel direct-path loads. The apparent choice might be to always use direct-path loads because they do not affect the objects in the database buffer cache. Before making this, or any other snap judgment, you should understand what happens during a direct-path or conventional load. Table 14-3 presents a summary of the key features of each method.

Table 14-3
Comparing Conventional and Direct-Path Loads

<i>Feature</i>	<i>Conventional Load</i>	<i>Direct-Path Load</i>
Data is saved to the database how?	Conventional path loads issue COMMIT statements after the number of rows indicated by the ROWS parameter have been written to the database.	Uses data saved to write data to the database and append data to the table above the high-water mark.

Continued

Table 14-3 (continued)

<i>Feature</i>	<i>Conventional Load</i>	<i>Direct-Path Load</i>
Database buffer cache usage?	Uses database buffer cache in the same way that an INSERT statement operates.	Does not use the database buffer cache, and therefore does not affect other objects in the cache.
Redo log generation?	Always generates redo log entries in much the same way an INSERT statement would.	Does not generate redo log entries if the database is in NOARCHIVELOG mode. If the database is in ARCHIVELOG mode and the NOLOGGING attribute has been set for the table, or the UNRECOVERABLE parameter has been specified in the control file, no redo is generated.
Constraint enforcement?	Enforces all constraints in much the way as an INSERT statement would.	Enforces only PRIMARY KEY, UNIQUE and NOT NULL constraints. All CHECK and FOREIGN KEY constraints on the target table are disabled. It is possible for constraints to remain disabled if the load encounters problems or PRIMARY KEY or UNIQUE constraints are violated during or at the end of the load.
INSERT trigger firing?	INSERT triggers fire normally for each row loaded.	INSERT triggers are not fired. Direct-path loads disable INSERT triggers at the start of the load and re-enable them after the load completes.
Clustered table support?	Loads into tables created on the cluster are fully supported.	Direct-path loads cannot load data into tables created on a cluster.
Other DML allowed on target table?	Users may make changes to all data not being loaded or modified by the load. Rows in the process of being loaded cannot be modified until a COMMIT is issued by SQL*Loader.	The segment into which the load is occurring is locked for exclusive use by SQL*Loader. No other changes may be made on the segment until the load completes or aborts.
Can be performed in parallel?	No, conventional loads can only take place serially.	Yes, direct-path loads can be parallelized.

As is evident from the information presented in Table 14-3, conventional loads, while making use of the database buffer cache and generating full redo, allow other DML to take place on the target segments. Conventional loads also ensure that all constraints are enforced and INSERT triggers fire. The main benefits of direct-path loads are speed and the fact that they can be parallelized.

Parallel direct-path loads

A variation of a direct-path load is a parallel direct-path load. Using this method, you can configure several SQL*Loader direct-path load sessions to load data into the same table at the same time. The main benefit of this process is speed.



Tip

Invoking multiple SQL*Loader sessions to load data into the same table when performing a conventional load is also supported. This is because Oracle does not treat SQL*Loader conventional loads any differently from a regular user session. Just as more than one user can perform INSERT operations on the same table at the same time, multiple SQL*Loader sessions can also operate on the same table without any special configuration parameters, as long as they are performing a conventional load. However, if you intend to do this you should ensure that multiple freelists exist for the table into which you will be loading data; otherwise, the load may take longer to complete.

The reason that a parallel direct-path load is different is because a direct-path load locks the segment when it starts, and in order to allow more than one direct-path load to take place, you need to specify the PARALLEL keyword on the SQL*Loader command line.

Even though multiple conventional loads on the same table are supported, they are typically not performed because of the excessive overhead and slow performance relative to a direct-path or parallel direct-path load.

In a parallel direct-path load, each SQL*Loader session allocates space for a temporary segment in a datafile belonging to the tablespace where the table being loaded is created. The FILE parameter of SQL*Loader enables you to specify which file a particular SQL*Loader session should use; if none is specified, a datafile belonging to the tablespace is chosen at random by SQL*Loader. You can also specify storage parameters for the temporary segments that are being created; otherwise, the storage parameters of the table being loaded will be used for each temporary segment.

Each SQL*Loader session continues to allocate space in the temporary segment it created as the load progresses. When the load is complete, the last extent of each SQL*Loader session's temporary segment is trimmed to delete any space not used by data. Oracle then combines all of the temporary segments to form a single segment, which is attached to the table segment above the high-water mark and now becomes part of the table.

While parallel direct-path loads can speed up loading data into a single table, they do place some restrictions on the process. These are similar to the restrictions placed on direct-path loads shown in Table 14-3, including:

- ♦ FOREIGN KEY and CHECK constraints are not enforced and need to be disabled manually before the load and enabled afterwards.
- ♦ INSERT triggers do not fire.
- ♦ Indexes are not updated and need to be dropped before the load starts and re-created after the load completes.
- ♦ Rows cannot be modified by a parallel direct-path load but only appended to a table. If the load replaces existing rows as well as adds new ones, you need to manually truncate the table before the load begins.

Parallel direct-path loads are primarily designed to perform updates of large fact tables in a data-warehousing environment. They either replace all the data in the table with new rows, or append a large amount of data to an existing fact table. Because of the volume of data being loaded, indexes are normally dropped and re-created to make the process more efficient. The data loaded usually meets constraint conditions, or, in case it does not, procedures are in place to deal with it.

Guidelines for Using SQL*Loader

When performing loads using SQL*Loader, you can implement a number of good practices that can make your job easier and the process smoother. Guidelines that you should follow when using SQL*Loader include the following:

- ♦ Instead of specifying all of the command-line parameters for the load when invoking SQL*Loader, create an ASCII text file with the parameters and values needed for each load and pass the name of the parameter file to SQL*Loader with the PARFILE parameter. This way, if you need to make a change to the parameters for a load, all you need to do is modify the parameter file and SHELL scripts or CMD/BAT files that invoke the load do not need to be modified. Furthermore, if you need to configure another load with similar parameters and values, you need only make a copy of the parameter file and make changes to the affected parameters, instead of creating the command-line syntax from scratch.
- ♦ Unless the amount of data to be loaded is small, always have a separate control file and datafile. Even though examples used previously in this chapter showed how to include the data in the control file, this is not normally done in real-world applications of SQL*Loader. Separating the datafile from the control file enables you to re-use the same control file for different datafiles if the data is being loaded into the same table.
- ♦ Prior to performing a large conventional load, you should manually create extents for the table to which data will be loaded. This prevents the dynamic allocation of extents during the load process, which will further slow it down. If you manually pre-allocate the extents before the load, this also allows you to place data on a different datafile belonging to the same tablespaces, which may improve performance of queries on the table after the load.

- ♦ When performing a direct-path load, if possible, pre-sort the records in the datafiles according to the index key on the table being loaded. The best key to use for this is the PRIMARY KEY for the table. If you do this, then you can also tell SQL*Loader that the datafiles contain pre-sorted data, which minimizes the use of sort space when SQL*Loader needs to create index entries at the end of the direct-path load.
- ♦ If you are performing a parallel direct-path load, and the tablespace where the table being loaded is created is composed of several datafiles (preferably on different disks), use the FILE parameter of SQL*Loader to specify a different datafile to place each SQL*Loader's temporary segment. This will allow potential striping of the data to be loaded across multiple physical disks and even further improve performance.
- ♦ When loading large amounts of data using a conventional load, specify a value for the ROWS parameter so that COMMITs take place at regular intervals, thereby minimizing the use of rollback segment space. This will make it less likely that a load terminates because of lack of rollback segment space for its before image information.
- ♦ After performing a direct-path load (parallel or not), make a backup of the datafiles that were affected. This is especially true if the UNRECOVERABLE parameter was specified in the control file. Failing to perform a backup after the load may result in loss of data and the requirement to repeat the load.
- ♦ To update statistics for the affected tables, you should issue the ANALYZE command.

Troubleshooting SQL*Loader

While SQL*Loader generally does what you tell it to, and, assuming you have given it good direction, it performs its task without errors; however, you may encounter situations where problems occur.

Perhaps one of the more common problems is that a load terminates abnormally because it has exceeded the MAXDISCARD or ERRORS value, or run out of rollback segment space, or any other reason. In this case, it may be likely that some or all of the rows that were loaded prior to the abnormal termination may be already in the database. In this case, starting the load from the beginning may produce duplicate rows or cause the load to fail a second time. In the case of terminated loads, you should perform these steps to continue from where the load left off:

- ♦ Review the log file created by the load to determine the number of rows that were successfully loaded into the affected tables.
- ♦ If loading data into a single table, re-start the load and include the SKIP parameter, passing that parameter a value representing the value you found in the log file for the number of rows successfully loaded on the previous attempt.
- ♦ If loading data into multiple tables, modify the control file and use the CONTINUE_LOAD option specifying the number of records to skip for each table.

- ♦ Whether or not you decide to continue a failed load, always check the status of all indexes on the affected tables. Chances are that SQL*Loader left the indexes in an UNUSABLE state because of the abnormal termination. Drop and re-create the indexes to make them usable once again.

Table 14-4 lists some reasons that a load may terminate and other conditions that can cause problems either during or after a load, as well as how to correct the problem.

Table 14-4
SQL*Loader Problems and Solutions

<i>Problem</i>	<i>Solution</i>
Insufficient space for the data being loaded	If there is not sufficient space to load the data or lack of space in the rollback segment, a load may terminate. Determine if the cause of the problem is due to lack of physical space on the hard drive, no more logical space in the datafile, or MAXEXTENTS being reached for the segment. Once you have determined the cause of the problem, rectify it and re-try the load.
Instance failure during the load	If the instance fails while the load is taking place, restart the instance and determine how much of the data was loaded into the database. You should also determine what caused the instance to fail and rectify that problem first. After doing so, re-start the load from the point at which it terminated when the instance failed.
Indexes UNUSABLE after performing a direct-path load with the SORTED INDEXES clause in the control file	The most likely reason that indexes remain unusable after performing this type of load is that the data was actually not in the proper order when it was loaded. Even a single row out of order causes SQL*Loader to leave the indexes in an unusable state. Drop and re-create the indexes to make them usable again.
Duplicate values for PRIMARY KEY or UNIQUE constraints, or unique indexes, when performing a direct-path load	<p>Because a direct-path load verifies these constraints during the load process, it may leave the indexes in an UNUSABLE state if violations are found and the data has been saved. This may be particularly true when performing a parallel direct-path load. To correct the problem, create an exceptions table and place the violating rows in the exceptions table. Correct the affected rows and enable the constraints.</p> <p>In the case of a unique index, you may need to create a UNIQUE constraint on the affected column(s) and then place the exceptions into the exceptions table to locate the rows causing the problem. You will then need to modify or remove the offending rows and then re-create the index.</p>

Problem	Solution
A single row does not fit inside the limit specified by the BINDSIZE parameter	In order for a load to succeed, BINDSIZE must specify sufficient memory to hold a single row. If you are loading LOBs into the database, this may require a rather large amount of RAM per row. Ensure that BINDSIZE is large enough to hold all of the data that makes up a single row. If this is not the case, increase BINDSIZE until it is.
ERRORS value exceeded	<p>If the value of the ERRORS parameter is exceeded during the load, SQL*Loader will terminate the load. Determine why the ERRORS parameter is being exceeded, correct the problem, and continue the load.</p> <p>The most common cause for ERRORS being exceeded is that an incorrect datafile was specified. In this case, you may also need to correct the data in the table to which rows may have been inserted and then re-try the load.</p>
DISCARDMAX value exceeded	If the number of discarded records exceeded the value specified by the DISCARDMAX parameter, the load will abort. The most common reason for this is the same as for ERRORS being exceeded – a wrong datafile was specified. Use the same methods as outlined for ERRORS being exceeded to correct the problem.

Key Point Summary

In preparing for the “Oracle®i DBA: Architecture and Administration” exam, please keep these points regarding loading data into Oracle in mind:

- ♦ You can load data from external sources into Oracle using the SQL*Loader utility, as well as third-party tools.
- ♦ Loading data from one table in the database to another table in the database can be accomplished using the INSERT INTO ... SELECT syntax.
- ♦ Using the APPEND hint in the INSERT INTO ... SELECT syntax performs a direct-load INSERT, which is typically much quicker than a conventional INSERT operation.
- ♦ Direct-load INSERTs can be parallelized if the PARALLEL hint is included in the statement, or the table being loaded into has a default degree of parallelism specified.
- ♦ Only one PARALLEL clause will be used when performing a direct-load INSERT.
- ♦ SQL*Loader supports both conventional and direct-path loads.

- ♦ Conventional loads work in much the same way as a series of INSERT statements and enforce all constraints, generate redo for each row loaded, fire INSERT triggers, and keep all indexes up-to-date.
- ♦ Direct-path loads are much quicker and may be configured to generate minimal redo, but they only enforce PRIMARY KEY, UNIQUE, and NOT NULL constraints. Direct-path loads do not fire INSERT triggers or enforce CHECK or FOREIGN KEY constraints.
- ♦ You need to verify the status of indexes after a direct-path load to ensure that they have not been marked UNUSABLE due to an error during the load. You should also check all constraints on a table to ensure that they are enabled after the direct-path load completes as they may be left disabled if rows that were loaded violate constraint conditions.
- ♦ Parallel direct-path loads, though potentially the fastest type of load available, require you to manually drop and re-create indexes before and after the load, respectively.
- ♦ Use the log file for the load to determine the reason for its abnormal termination, as well as the starting point for the load continuation when using the SKIP parameter or the CONTINUE_LOAD option in the control file.
- ♦ When invoking SQL *Loader, you must specify at least the name of the control file.



STUDY GUIDE

To better prepare yourself for the questions you may be asked about loading data when taking the “Oracle8i: Architecture and Administration” exam, answer the assessment questions and perform the labs in this part of the chapter.

Assessment Questions

1. In order to perform a direct-load INSERT, which of the following commands must be issued prior to performing the INSERT? (Choose the best answer.)
 - A. ALTER SESSION DISABLE PARALLEL QUERY
 - B. ALTER SYSTEM SET QUERY_REWRITE_ENABLED=TRUE
 - C. ALTER DATABASE ENABLE PARALLEL QUERY
 - D. ALTER SESSION ENABLE PARALLEL DML
 - E. ALTER SESSION DISABLE PARALLEL DML
 - F. None of the above.
2. Which of the following are enforced during a parallel direct-path load? (Choose all correct answers.)
 - A. INSERT trigger
 - B. PRIMARY KEY
 - C. FOREIGN KEY
 - D. CHECK constraint
 - E. NOT NULL constraint
3. If you issue the following command:

```
sqlldr system/manager@certdb students
```

What will be the name of the “bad” file created by the load? (Choose the best answer.)

- A. BAD.LOG
- B. SQLLDR.BAD
- C. STUDENTS.BAD
- D. STUDENTS.LOG
- E. No “bad” file will be created

4. Which of the following types of SQL*Loader loads can be used to add data to a table created on a cluster? (Choose all correct answers.)
- A. Direct-Path Load
 - B. Conventional Load
 - C. Parallel Direct-Path Load
 - D. Parallel Conventional Loads
 - E. You cannot use SQL*Loader to load data into a clustered table.
5. When invoking SQL*Loader, where can you specify the location of the datafile that will be used as the source of the records to be loaded? (Choose all correct answers.)
- A. On the command line
 - B. In the datafile
 - C. In the control file
 - D. In the parameter file
 - E. In the log file
6. If a SQL*Loader direct-path load fails, which of the following is most likely true? (Choose three correct answers.)
- A. INSERT triggers are disabled on the table.
 - B. INSERT triggers are enabled on the table.
 - C. PRIMARY KEY constraints are disabled on the table.
 - D. PRIMARY KEY constraints are enabled on the table.
 - E. FOREIGN KEY constraints are disabled on the table.
 - F. FOREIGN KEY constraints are enabled on the table.
7. When performing a direct-path load, which of the following operations are allowed on the table being loaded? (Choose all correct answers.)
- A. SELECT
 - B. DELETE
 - C. GRANT
 - D. TRUNCATE
 - E. UPDATE
 - F. None of the above

8. You need to load data into a table that is partitioned on a date column. You have 12 load files, each representing a month's worth of data. All of the partitions had large amounts of data recently deleted, but still contain about 60 percent of the original rows. Your load will double the amount of data in the tables. Which type of SQL*Loader load method should you perform? (Choose the best answer.)
- A. Conventional load
 - B. Direct-path load
 - C. Parallel conventional load
 - D. Parallel direct-path load
9. Which of the following is not a file that is required or created by SQL*Loader? (Choose the best answer.)
- A. Log file
 - B. Control file
 - C. Bad file
 - D. Parameter file
 - E. Discard file
10. What degree of parallelism will be used by the following statement? (Choose the best answer.)
- ```
INSERT /*+PARALLEL(ArcStudents,2) APPEND */ INTO ArcStudents
 SELECT /*+PARALLEL(Students,4) */ FROM Students
 WHERE StudentNumber < 1000
```
- A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 0

## Scenarios

1. You have been asked to devise a strategy for populating the fact table of the data warehouse for a large retailer. You have been provided with the following information:
- The fact table in the database is partitioned according to product line.
  - The datafiles are generated from the order entry system and will need to be transformed during the load operation.

- The datafiles use data extracted according to product line and date of order.
- The load must complete in 4 hours or less on a weekend.
- Each datafile will contain approximately 1 million rows per day of activity
- Full backups are scheduled to take place on Monday between 1:00a.m. and 5:00a.m.
- Transaction logs are backed up daily at midnight.

Based upon the information presented, devise a load strategy that would best suit the requirements of this organization.

2. You need to load large amounts of data on a clustered table. Some of the data already exists in the database while some of it needs to be extracted from a DB2 database on a mainframe and then added to the table. Information from the mainframe may duplicate information already found in the other tables in the database. When loaded into the clustered table, these conflicts need to be resolved.

What load strategy would you devise for the clustered table to ensure that only the sanitized data is added?

## Lab Exercises

### Lab 14-1 Loading Data Using SQL\*Loader

1. Start SQL\*Plus and connect to the CERTDB database as user student with a password of oracle.
2. Create a table with the exact same structure as the Instructors table and call it *temp\_instructors*. Make sure that the temp\_instructors table you just created has no rows and then exit SQL\*Plus
3. If you have not already done so, copy the Labs folder in the Scripts folder of the companion CD-ROM to the hard drive. Make sure that you have removed the Read-Only attribute from all files in the Labs folder.
4. Modify the **Instructors.ctl** file in the Labs folder so that it loads data into the temp\_instructors table in the student schema.
5. Invoke SQL\*Loader and load the data pointed to by the Instructors.ctl control file into the temp\_instructors table. You can use Oracle Enterprise Manager to perform this load, or invoke SQL\*Loader from the command line with or without a PARFILE.
6. Review the log of the load to see if any errors occurred. If so, correct the problem, truncate the table, and try again.

## Lab 14-2 Loading Data Using Direct-Load INSERT

1. Start SQL\*Plus and connect to the CERTDB database as user Student with a password of **oracle**.
2. Formulate a SQL statement that would perform a direct-load INSERT to add all the rows of the temp\_instructors table to the Instructors table in the Student schema.
3. Verify that all of the rows have been added to the Instructors table and commit your changes.

# Answers to Chapter Questions

## Chapter Pre-Test

1. The difference between a regular INSERT and a direct-load INSERT is that a regular INSERT makes use of the database buffer cache, always generates rollback and redo, and determines which block to add the rows to by grabbing the freelist for the segment and finding the next block.  
  
A direct-load INSERT writes the rows directly to the datafile and appends the data to the segment above the high-water mark. A direct-load INSERT may not generate a redo, which means that you should backup the tablespace where the table or partition resides after the operation is performed.
2. Some of the advantages of using a direct-path load instead of a conventional load are that a direct-path load is significantly quicker than a conventional load. This is because a direct-path load does not enforce FOREIGN KEY or CHECK constraints, does not fire INSERT triggers, and does not make use of the database buffer cache. Furthermore, direct-path loads generate only minimal redo incurring less IO on the system.
3. You can limit the number of records to be inserted into a table when performing a conventional or direct-path load by specifying a value for the ROWS parameter? In both types of loads this directs SQL\*Loader to commit the changes after the specified number of rows (for conventional load) or close after the specified number of rows (for a direct-path load) have been added to the table.
4. If you attempt to start two SQL\*Loader sessions that need to load data in the same table, you should perform a parallel direct-path load in order for the process to complete as quickly as possible.
5. After performing a direct-path load and specifying the UNRECOVERABLE parameter in the control file, you should backup the tablespaces on which the tables that had data loaded into them exist. If you do not perform a backup, the loaded data may be lost or the load may need to be repeated should a disk failure take place or the files become corrupt.

6. When invoking SQL\*Loader you must specify the name of the control file. All other parameters can be in the control file or you will be prompted for any required values.
7. Conventional loads ensure that INSERT triggers fire. Direct-path loads disable INSERT triggers at the beginning of the load and enable them after the load completes.
8. If you are only loading a small amount of data, or you have a lot of empty blocks below the high-water mark for the segment, you would be better off performing a conventional load rather than a direct-path load. In the second scenario, this ensures that all of the empty blocks will once again be populated with data and not cause full table scans to read a large number of empty blocks, thus performing worse than normal.
9. Using a direct-load INSERT to add data to a table from another table in the same database saves it in new extents built for the table and attached above the high-water mark for the table.
10. If you created a table on a cluster, you would need to perform a conventional INSERT operation to load a large number of rows into the table from another table in the same database. This is because direct-load INSERTs cannot load data into clustered tables.

## Assessment Questions

1. **F.** In order to perform a direct-load INSERT, you do not need to issue any special commands. You simply need to include the APPEND hint in the INSERT INTO statement. If you were performing a parallel direct-load INSERT, you would need to enable parallel DML for the session using the ALTER SESSION ENABLE PARALLEL DML command.
2. **B, E.** Only PRIMARY KEY and NOT NULL constraints are enforced during a parallel direct-path load. FOREIGN KEY, CHECK constraints, and INSERT triggers are disabled before the load.
3. **C.** If you issue the command:

```
sqlldr system/manager@certdb students
```

The name of the bad file will be the same as the control file, but with a .bad extension, or students.bad This is because the command-line parameters specified infer that the first value is for the USERID parameter while the second is for the CONTROL parameter. As no other values were specified, they are either in the control file or will assume the default. In this case, the default for the bad file will be the control file name “students” with the .bad extension, or students.bad.

4. **B, D.** Tables created on a cluster only support conventional loads, whether done serially or in parallel.

5. **A, C, D.** You can specify the location of the datafile that will be used as the source of the records to be loaded on the command line with a `DATA` parameter, in the control file using the `INFILE` parameter, or in a parameter file with the `DATA` parameter.
6. **A, D, E.** If a SQL\*Loader direct-path load fails, it is likely that `INSERT` triggers and `FOREIGN KEY` constraints on the table will remain disabled, since they were disabled when the load began, along with `CHECK` constraints. `PRIMARY KEY` constraints should remain enabled, except in the unlikely situation that rows were loaded which violated the constraint, in which case they may also be disabled.
7. **F.** When performing a direct-path load the table being loaded into is locked for exclusive use by SQL\*Loader. For this reason, no operations are allowed on the table, including queries (`SELECT`), DML statements (`INSERT`, `UPDATE`, `DELETE`), DCL statements (`GRANT`, `REVOKE`), or DDL statements (`ALTER`, `DROP`, `TRUNCATE`).
8. **C or A.** The best load type to use in this case is a conventional load into each partition, which can also be considered a parallel conventional load, but is not really as each SQL\*Loader session will touch only a single segment—the partition that corresponds to the datafile data being loaded. You would not use a direct-path load, parallel or not, because there is too much empty space below the high-water mark and doing so would leave this intact, making full table scans inefficient.
9. **D.** The parameter file is not required or created by SQL\*Loader. SQL\*Loader requires a control file, and creates the log file, bad file, and discard file when the load starts.
10. **B.** When issuing the statement, the parallel hint specified for the table into which the data is being loaded in a parallel direct-load `INSERT` takes precedence and is used. The value specified for the `ArcStudents` tables was 2 and will be the maximum degree of parallelism for the entire operation.

## Scenarios

1. The best strategy to employ in this scenario is a parallel direct-path load. The fact that the table is partitioned according to product line and the datafiles will also contain information that is extracted according the product line, you can take each product line's datafile and use it as the source for each of the parallel direct-path loads. Setting the `UNRECOVERABLE` flag in the control file for each load ensures that minimal redo is generated and thus allows each load to complete in the four-hour window. If the load is performed on a Sunday evening after 7p.m., it can take part in the regular weekend backup strategy that performs a full database backup on Monday at 1:00a.m. (although this time may need to be changed as the data grows).

2. Since the load in this case involves data on a clustered table, the only load that can be used is a conventional load directly to the table. However, because some of the data comes from a DB2 database and the rest from the same database as the clustered table, you should create a “scratch” table and load the data from the DB2 database into the “scratch” table using a direct-path load. You can then perform a direct-load INSERT to load the data from the other tables in the database into the “scratch” table as well. When all of the data is in the “scratch” table, perform any operations to clean up the data so that any conflicts between the DB2 data and the current database data are resolved. Once this is complete, you can perform a conventional INSERT INTO ... SELECT statement to add the data to the clustered table.

## Lab Exercises

The solutions to the labs presented here assume that you are performing the lab on a WindowsNT or Windows 2000 computer. If you are using a Linux or UNIX-based computer, use the equivalent commands for your operating system to perform the OS-related actions (for example, editing the control file).

### Lab 14-1

1. Start SQL\*Plus and connect to the CERTDB database as user student with a password of **oracle**.
2. Create a table with the exact same structure as the Instructors table and call it *temp\_instructors*. Make sure that the temp\_instructors table you just created has no rows and then exit SQL\*Plus.

```
SQL> CREATE TABLE temp_instructors AS
 2 SELECT * FROM Instructors WHERE 0=1;
```

Table created.

SQL>

3. If you have not already done so, copy the Labs folder in the Scripts folder of the companion CD-ROM to the hard drive. Make sure that you have removed the Read-Only attribute from all files in the Labs folder.

```
C:\> H:
H:\> CD Scripts
H:\Scripts>xcopy *.* C:\CERTDB /S/V
H:add_data.sql
H:CERTDB.doc
H:certdbobj.sql
H:createuser.sql
H:enrollmentHistory.sql
H:insert_data.sql
```

```
H:nonequijoin.sql
H:selfjoin.sql
H:Labs\Instructors.ctl
...

H:\Scripts>
C:\>cd certdb
C:\CERTDB>attrib -r *.* /s
C:\CERTDB>
```

4. Modify the **Instructors.ctl** file in the Labs folder so that it loads data into the **temp\_instructors** table in the student schema.

```
C:\CERTDB>cd Labs
C:\CERTDB\Labs>notepad instructors.ctl

-- Copyright (c) 2001 by Hungry Minds Inc.
--
-- Instructors.ctl
--
-- SQL*Loader Control and Data Combined Example
--
-- Created: June 1, 2001 by Damir Bersinic
-- Modified:
--
LOAD DATA
INFILE *
INTO TABLE student.Temp_Instructors
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
... (remainder of file truncated to save space)
```

5. Invoke SQL\*Loader and load the data pointed to by the **Instructors.ctl** control file into the **temp\_instructors** table. You can use Oracle Enterprise Manager to perform this load, or invoke SQL\*Loader from the command line with or without a PARFILE.

```
C:\CERTDB\Labs>SET ORACLE_SID=CERTDB
C:\CERTDB\Labs>sqlldr USERID=student/oracle
CONTROL=Instructors.ctl
```

```
SQL*Loader: Release 8.1.7.0.0 - Production on Wed Jun 13
15:18:19 2001
```

```
(c) Copyright 2000 Oracle Corporation. All rights reserved.
```

```
Commit point reached - logical record count 4
Commit point reached - logical record count 5
```

```
C:\CERTDB\Labs>
```

6. Review the log of the load to see if any errors occurred. If so, correct the problem, truncate the table, and try again.

To review the log file on a Windows-based computer, use the EDIT.EXE text editor or the TYPE command at the command prompt level. Using Notepad presents the data with line feeds showing up as printable characters in Notepad until the file has been edited with EDIT.EXE.

```
C:\CERTDB\Labs>edit Instructors.log
```

```
SQL*Loader: Release 8.1.7.0.0 - Production on Wed Jun 13
15:18:19 2001
```

```
(c) Copyright 2000 Oracle Corporation. All rights reserved.
```

```
Control File: instructors.ctl
Datafile: instructors.ctl
Bad File: instructors.bad
Discard File: none specified
```

```
(Allow all discards)
```

```
Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array: 64 rows, maximum of 65536 bytes
Continuation: none specified
Path used: Conventional
```

```
Table STUDENT.TEMP_INSTRUCTORS, loaded from every logical
record.
```

```
Insert option in effect for this table: INSERT
```

| Column Name    | Position | Len | Term | Encl | Datatype  |
|----------------|----------|-----|------|------|-----------|
| INSTRUCTORID   | FIRST    | *   | ,    | 0(") | CHARACTER |
| LASTNAME       | NEXT     | *   | ,    | 0(") | CHARACTER |
| FIRSTNAME      | NEXT     | *   | ,    | 0(") | CHARACTER |
| INSTRUCTORTYPE | NEXT     | *   | ,    | 0(") | CHARACTER |
| EMAIL          | NEXT     | *   | ,    | 0(") | CHARACTER |
| COMMENTS       | NEXT     | *   | ,    | 0(") | CHARACTER |

```
Table STUDENT.TEMP_INSTRUCTORS:
```

```
5 Rows successfully loaded.
0 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.
```

```
Space allocated for bind array: 65016
bytes(42 rows)
Space allocated for memory besides bind array: 0 bytes
```

```

Total logical records skipped: 0
Total logical records read: 5
Total logical records rejected: 0
Total logical records discarded: 0

Run began on Wed Jun 13 15:18:19 2001
Run ended on Wed Jun 13 15:18:21 2001

Elapsed time was: 00:00:01.49
CPU time was: 00:00:00.06

```

## Lab 14-2

1. Start SQL\*Plus and connect to the CERTDB database as user student with a password of oracle.
2. Formulate a SQL statement that would perform a direct-load INSERT to add all the rows of the temp\_instructors table to the Instructors table in the Student schema.

```

SQL> INSERT /*+APPEND */ INTO Instructors
 2 SELECT * FROM temp_instructors;

```

5 rows created.

SQL>

3. Verify that all of the rows have been added to the Instructors table and commit your changes.

```

SQL> SELECT InstructorID, LastName, FirstName FROM
Instructors;
SELECT InstructorID, LastName, FirstName FROM Instructors
*
ERROR at line 1:
ORA-12838: cannot read/modify an object after modifying it in
parallel

```

```

SQL> COMMIT;

```

Commit complete.

```

SQL> SELECT InstructorID, LastName, FirstName FROM
Instructors;

```

| INSTRUCTORID | LASTNAME | FIRSTNAME |
|--------------|----------|-----------|
| 300          | Harrison | Michael   |
| 310          | Keele    | Susan     |
| 100          | Ungar    | David     |
| 110          | Jamieson | Kyle      |
| 200          | Cross    | Lisa      |

|     |          |         |
|-----|----------|---------|
| 210 | Williams | Geoff   |
| 410 | Chiu     | Lana    |
| 450 | LaPoint  | Adele   |
| 500 | Bersinic | Damir   |
| 501 | Giles    | Stephen |
| 502 | Ross     | Todd    |

| INSTRUCTORID | LASTNAME | FIRSTNAME |
|--------------|----------|-----------|
|--------------|----------|-----------|

---

|     |         |       |
|-----|---------|-------|
| 503 | Sabinin | Yury  |
| 504 | Brown   | Myles |

13 rows selected.

SQL>

# Reorganizing Data

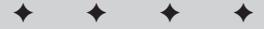
---

## EXAM OBJECTIVES

- ◆ Reorganizing Data
  - Reorganize data using the Export and Import utilities
  - Move data using transportable tablespaces

# 15

CHAPTER



## CHAPTER PRE-TEST

1. What are the four modes of the Export utility?
2. All objects in the database are exported using the database mode except for objects owned by which user and why?
3. What is the difference between a Conventional Path versus a Direct Path export?
4. What is the biggest limitation of an Interactive Export mode?
5. What options of the import are used to change the ownership of objects from one user(s) to another user(s)?
6. How can you ensure that imports do not run out of rollback space?
7. What are transportable tablespaces used for?
8. What are some of the limitations of transportable tablespaces?

**T**he main focus of this chapter is on the use of the Import and Export utilities as well as transportable tablespaces. The first section of this chapter provides a general overview on reorganizing data that covers the CREATE TABLE as SELECT (CTAS) command, the ALTER TABLE *table\_name* MOVE command, and the Import and Export utilities. The second part of the chapter focuses specifically on the Export utility. There are three different export modes as well as two different export paths that need to be considered. The Export utility can be run interactively, using a command-line interpreter or going through Enterprise Manager, and it is important to understand the strengths and weaknesses of all three options. When you're using the command-line interpreter or Enterprise Manager, Oracle provides a rich set of options to add flexibility to the utility. The second section of this chapter covers the Import utility. In addition to covering the three different methods for invoking the utility, this chapter also discusses the rich parameters of the utility, which add flexibility and power. Finally, this chapter discusses a new feature for Oracle8i, transportable tablespaces. Transportable tablespaces provide a powerful tool for moving data between databases extremely quickly and efficiently. The utility does, however, have some limitations that need to be examined.

The Import and Export utilities used to be the primary means for many DBAs of reorganizing and rebuilding data. Many DBAs also used the utilities as part of a backup and recovery strategy. However, with the introduction of the ALTER TABLE *table\_name* MOVE command, DBAs can perform most of the data reorganization tasks using this command, which is much quicker, safer, and easier to use. Although this task no longer requires the Import and Export utilities, a great many still do. These are tools and knowledge that all DBAs must have. Using these utilities is still the only way to change ownership of objects in a database, perform logical backups of objects in the database, and perform logical recoveries of objects, and in Oracle8i, using these tools is required for using transportable tablespaces. Transportable tablespaces allow DBAs to move data from one database to another by simply exporting the metadata and copying datafiles for a tablespace at the OS level. This dramatically simplifies the task of populating Data Warehouses and saves a tremendous amount of time. Although Oracle is improving data reorganization in the database, the Import and Export utilities are still required knowledge for all DBAs.

## Overview of Reorganizing Data

Almost all DBAs are going to encounter a time in their careers when data will need to be moved. The main reason is to fix problems caused by fragmentation, but some of the other common reasons are the addition of new hardware, users who own objects in the database that need to be removed, migrating to new releases of the database or operating system, moving data between OLTP and DSS databases, and creating test environments. The Import and Export utilities used to be the best choice for these types of activities. In Oracle8i, a new feature has been added

to the ALTER TABLE command that permits tables to be moved to new tablespaces or simply to be rebuilt on the same tablespace, fixing the problems caused by fragmentation. Although this new command has taken some of the focus away from the Import and Export utility, these utilities are still an integral part of all reorganizing data in all databases.

The task of moving data between databases used to be very time consuming. The data would first need to be exported from the production system and then imported in the DSS or test database. Depending on the volume of data, this process could take anywhere from a couple of minutes to a couple of days. In Oracle8i, transportable tablespaces have been added to simplify and speed up this task.

The following is a list of reasons and scenarios for moving data:

- ♦ Reorganizing table(s) and table data when one of the following situations arise:
  - Tables in one tablespace may need to move to another to reduce disk IO on devices to facilitate tablespace backups or when using transportable tablespaces.
  - Tables that contain migrated rows need to be rebuilt.
  - A table may contain many blocks with a large amount of free space on blocks. This can happen if a table is subject to many UPDATE and DELETE statements, or after data has been moved to another table for archiving.
  - A table may contain many empty blocks as a result of a bulk DELETE statement. This artificially sets the high-water mark to an inflated level, which is bad for full table scan performance.
- ♦ Moving data and objects owned by one user to another when a user is removed from the database. Oracle will not allow users to be deleted from the system if they own objects, so the objects must either change ownership or be dropped when the user is dropped. Although dropping objects owned by a user is not always possible in production environments, changing ownership generally is.
- ♦ Moving data and objects between databases. The move can be a copy to a test system or it can be from an OLTP to a DSS or OLTP to a Staging Database and then to a DSS database.
- ♦ Migrating a database to a different OS such as Solaris to HP-UX or Migrating a databases to a newer release of an Oracle database. The new database can be on a different machine or use a different character set or even a different database block size. Note that going from newer releases to older ones is not always possible.

- ♦ Creating a baseline database for repeating tests in a test or QA database. When QA engineers are performing tests and find bugs or problems in program logic, these test runs commonly span several trials before all bugs and problems are corrected. Having the ability to quickly recreate an environment to test a new fix is a valuable feature of the Import and Export utilities.
- ♦ Creating sets of default database values. For example, the payroll tax tables for each state will be different. An export file can be created with all the defaults for each state. When the application is being deployed in California, the California defaults are loaded; when it is New York, the New York defaults are the ones loaded.
- ♦ Performing logical backups of database objects. A logical backup can be done for any object in a database; however, it is most beneficial for static objects. If a static object is accidentally dropped or data is deleted, it can be quickly recreated from an export file. The export file cannot be used with database recovery, so using exports for objects that change is not a good idea because changes cannot be recovered from export file backups. DBAs will also perform logical backups of objects before performing maintenance operations on objects. This way, if something goes wrong during the maintenance operation, the objects can be quickly recovered from the export file.

## Moving Data

This section looks at the three main methods for moving data, including CREATE TABLE <tablename> as SELECT, ALTER TABLE <tablename> MOVE, and Import/Export.

### The create table as select (CTAS) method

This command creates a new table with the columns and data that are selected in the subquery. The command can be used to select all columns and all rows from another table. If the table being selected is in the same schema as the new table being created, it must have a different name because no two objects in the same schema can share the same name. The table being created can have its own storage parameters and can be located in a different tablespace, or another schema, or another database. After you have created the new table, you would need to drop the original table before you can change the new table's name to the original table. The following example copies the LOCATIONS table to a new table called LOCATIONS\_TEMP. The command syntax looks something like this:

```
CREATE TABLE LOCATIONS_TEMP
 STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 0)
 TABLESPACE TOOLS
AS
 SELECT * FROM LOCATIONS;
```

To verify that the new table has been created with the correct number of rows, you can enter the following commands:

```
SQL> DESC LOCATIONS_TEMP
Name Null? Type

LOCATIONID NOT NULL NUMBER(38)
LOCATIONNAME NOT NULL VARCHAR2(50)
ADDRESS1 VARCHAR2(50)
ADDRESS2 VARCHAR2(50)
CITY VARCHAR2(30)
STATE CHAR(2)
COUNTRY VARCHAR2(30)
POSTALCODE CHAR(10)
TELEPHONE CHAR(15)
FAX CHAR(15)
CONTACT VARCHAR2(50)
DESCRIPTION VARCHAR2(2000)
```

```
SQL> SELECT COUNT(*) FROM LOCATIONS;
```

```
 COUNT(*)

 3
```

```
SQL> SELECT COUNT(*) FROM LOCATIONS_TEMP;
```

```
 COUNT(*)

 3
```

So far, so good! The command created a new table and brought over all the data. To complete the task, the `LOCATIONS` table would need to be dropped and the table `LOCATIONS_TEMP` would need to be changed to `LOCATIONS`:

```
SQL> DROP TABLE LOCATIONS CASCADE CONSTRAINTS;
```

```
SQL> ALTER TABLE LOCATIONS_TEMP RENAME TO LOCATIONS;
```

The major downside to this method for moving data is that only `NOT NULL` constraints are carried forward to the new table. All other existing constraints would need to be added afterward, and all referencing `FOREIGN KEY` constraints would need to be `ENABLED`. Also, any program units such as procedures, packages, functions, and triggers would need to be recompiled or recreated. As you can imagine, errors can be easily made using this method.

This method is a great way for quickly moving data from one table to another for performing certain maintenance tasks or for creating subsets of data for reporting purposes.

You also can use this method to quickly create a table with the same structure as an existing one but without the data in the source. To do this, you simply issue the same statement as before but include a WHERE condition that returns no rows in the SELECT statement (WHERE 1=0, for example), as follows:

```
CREATE TABLE LOCATIONS_TEMP
 STORAGE (INITIAL 100k NEXT 100k PCTINCREASE 0)
 TABLESPACE TOOLS
 AS
 SELECT * FROM LOCATIONS WHERE 1=0;
```

In this case, issuing any command to retrieve data from the LOCATIONS\_TEMP table would result in zero rows returned, as follows:

```
SQL> SELECT COUNT(*) FROM LOCATIONS_TEMP;

COUNT(*)

0
```

## The alter table method

New for Oracle8i is the MOVE option added to the ALTER TABLE command. This option recreates the table being altered, with the same name, preserving all constraints, indexes, and program units. The MOVE option allows for the specification of a new tablespace and storage options. If you don't specify a tablespace, the table is rebuilt in the same tablespace. If you don't specify any storage options, the existing storage options, not the system defaults, are used.



If tables are being rebuilt in the same tablespace, two times the disk space is required because the table is created first before the original is dropped. In moving a 100MB table, at least 200MB of space needs to be available in the tablespace.

The ALTER TABLE command dramatically simplifies the reorganization of tables. The only drawback is that all indexes on the table will need to be rebuilt. Rebuilding is required because the rows are assigned new ROWIDs. The syntax is as follows:

```
ALTER TABLE LOCATIONS MOVE TABLESPACE TOOLS;

SQL> SELECT * FROM LOCATIONS WHERE LOCATIONID=10;
SELECT * FROM LOCATIONS WHERE LOCATIONID=10
*
ERROR at line 1:
ORA-01502: index 'STUDENT.PK_LOCATIONID' or partition of such
index is in
unusable state
```

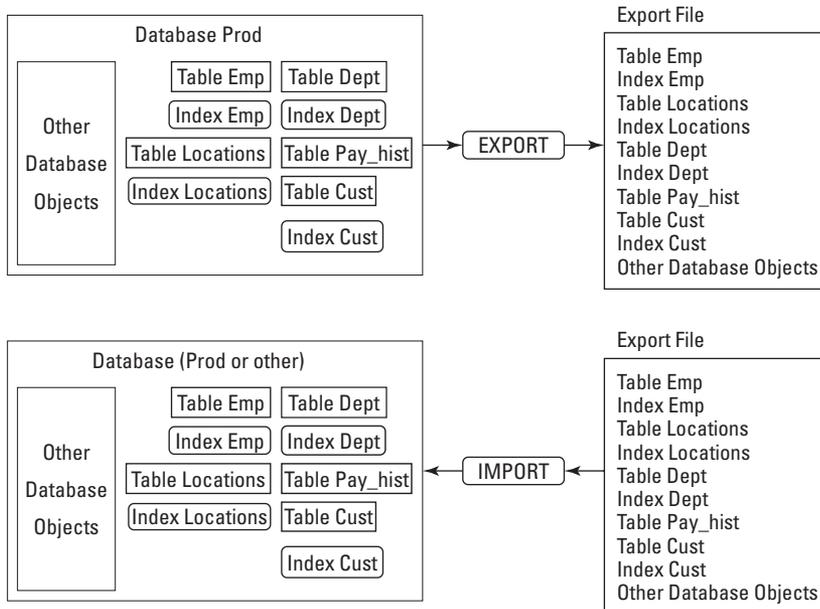
So, all that is required is an index rebuild before the table is usable.

```
SQL> ALTER INDEX PK_LOCATIONID REBUILD;
```

You can't use this command to change ownership of objects or to transfer data between databases.

## The Import and Export utilities

The Import and Export utilities allow DBAs to move data between Oracle databases or within the same database. The Export utility creates an OS binary file to disk or tape with a consistent view of the data and the data definitions of the object(s) being exported. The Import utility must read a binary file created by the Export utility and can create and load the objects stored in the export file into a database. The Import utility cannot be used to read files in any other format. Figure 15-1 illustrates the creation of the binary export file using the Export utility and the processing of the binary file with the Import utility.



**Figure 15-1:** Import and Export utility overview

## Export modes

The Export utility supports four different export modes. Deciding on which method to use depends upon the data that needs to be exported. The four modes are Table, User, Tablespace and Database.

## Table mode

Using the Table mode, users can export any table in their schema or any table they own. If a privileged user is performing the export, that user can export any object. A privileged user is one with the EXP\_FULL\_DATABASE role enabled.



Chapter 19 covers the EXP\_FULL\_DATABASE role in more detail.

For each table in the table list that the user is exporting, the following information can be exported and imported:

- ♦ Object type definitions used by table
- ♦ Table definitions
- ♦ Table data
- ♦ Table data by partition for partitioned tables (default is all partitions)
- ♦ Nested table data
- ♦ Owner's table grants (that is, permissions granted to others by the owner of the table)
- ♦ Owner's table indexes (that is, indexes created on the table by the owner of the table)
- ♦ Table constraints (primary, unique, check)
- ♦ Definition of analyze method to be used on the import
- ♦ Column and table comments
- ♦ Auditing information
- ♦ Security policies for the table
- ♦ Table referential constraints
- ♦ Owner's table triggers (that is, triggers created on the table by the owner of the table)

In addition, privileged users can export and import:

- ♦ Triggers on the table owned by other users
- ♦ Indexes on the table owned by other users

This option provides a great way to quickly move a table or to back up a table before performing maintenance operations.

## User mode

When the User mode is specified, the data that is exported depends upon the user performing the export. If the user is a privileged user, objects owned by any user can be exported. The following are the exceptions:

- ♦ Index and triggers owned by the user that reference another user's objects are not exported.
- ♦ Triggers and indexes on the user's tables not owned by the user being exported.

When a nonprivileged user is performing the export, only objects owned by that user can be exported. Triggers and indexes created on that user's tables by other users are not exported.

Following is the list of data that can be exported and imported under the User mode:

- ♦ Object type definitions used by table
- ♦ Table definitions
- ♦ Table data
- ♦ Nested table data
- ♦ Owner's table grants
- ♦ Owner's table indexes
- ♦ Table constraints (primary, unique, check)
- ♦ Analyze table method
- ♦ Column and table comments
- ♦ Auditing information
- ♦ Security policies for table
- ♦ Table referential constraints
- ♦ Private synonyms
- ♦ User views
- ♦ User stored procedures, packages, and functions
- ♦ Referential integrity constraints
- ♦ Operators
- ♦ Triggers
- ♦ Indextypes
- ♦ Snapshots and materialized views
- ♦ Snapshot logs
- ♦ Job queues
- ♦ Refresh groups
- ♦ Dimensions
- ♦ Procedural objects

The User mode is a great way to back up objects owned by a particular user or to transfer ownership of database objects between users.

## Tablespace mode

This mode is used to move tablespaces from one database to another as part of the transportable tablespace feature of Oracle8i. The user must be a privileged user to perform this type of export. Exports using this mode just export the metadata for the tablespace and not the actual data. The data will be transferred over in the physical datafiles when they get transferred between the two databases.

Following is a list of information that can be exported using the Tablespace mode.

- ♦ Cluster definitions
- ♦ Object type definitions used by the table
- ♦ Table definition (table rows are not included)
- ♦ Table grants
- ♦ Table indexes
- ♦ Table constraints (primary, unique, check)
- ♦ Column and table comments
- ♦ Referential integrity constraints
- ♦ Bitmap indexes (*Note:* not functional or domain indexes)
- ♦ Triggers

## Database mode

The Database mode will export everything in the database except for objects owned by the user SYS. Only privileged users can perform full database exports. Full database exports offer a good way to create test databases and to migrate databases between versions of Oracle or different operating systems.



Because it is impossible, during a full database export, to export objects owned by the user SYS, SYS should not own any objects except for the data dictionary. Always create a new user for creating objects so that, when the full database export is performed, all objects except the ones not needed, those of the data dictionary, will be exported and then imported. If objects other than the data dictionary objects are owned by the user SYS, Oracle recommends that a table-level export be performed on those tables to change the ownership of the objects to some user other than SYS.

The Database mode is also a good way to get a logical backup of the database, create test databases, and migrate databases.

Following is a list of data that you can export and import using the Database mode.

- ♦ Tablespace definitions
- ♦ User definitions
- ♦ Roles
- ♦ System privilege grants
- ♦ Role grants
- ♦ Default roles
- ♦ Tablespace quotas
- ♦ Resource costs
- ♦ Rollback segment definitions
- ♦ Database links
- ♦ Sequence numbers
- ♦ All directory aliases
- ♦ Application contexts
- ♦ All foreign function libraries
- ♦ All object types
- ♦ All cluster definitions
- ♦ Default and system auditing
- ♦ Table definitions
- ♦ Table data
- ♦ Nested table data
- ♦ Table grants
- ♦ Table indexes
- ♦ Table constraints (primary, unique, check)
- ♦ Analyze table
- ♦ Column and table comments
- ♦ Auditing information
- ♦ All referential integrity constraints
- ♦ All synonyms
- ♦ All views
- ♦ All stored procedures, packages, and functions
- ♦ Operators

- ♦ Indextypes
- ♦ All triggers
- ♦ Analyze cluster
- ♦ All snapshots and materialized views
- ♦ All snapshot logs
- ♦ All job queues
- ♦ All refresh groups and children
- ♦ Dimensions
- ♦ Password history
- ♦ System auditing

## Moving data scenarios

There is not always one right way to move data. Often, you can use any of the three options; the best option really does depend upon the situation. Table 15-1 outlines the different reasons for moving data as well as which options are available.

| <b>Table 15-1<br/>Moving Data Options</b> |             |                             |                                           |                                                                          |
|-------------------------------------------|-------------|-----------------------------|-------------------------------------------|--------------------------------------------------------------------------|
| <b>Scenario</b>                           | <b>CTAS</b> | <b>ALTER TABLE<br/>MOVE</b> | <b>Export</b>                             | <b>Recommended Option</b>                                                |
| Reorganize table, index, or both          | (           | (                           | Table or User mode                        | ALTER TABLE MOVE                                                         |
| Change ownership of object                | (           |                             | Table or User mode                        | Table Mode export                                                        |
| Move data between different databases     | (           |                             | Table, User, Tablespace, or Database mode | One of the export options, depending on which data needs to be moved     |
| Migrate to new release of Oracle          |             |                             | Database mode                             | Database mode                                                            |
| Relocate table to new tablespace          | (           | (                           | Table mode                                | ALTER TABLE MOVE                                                         |
| Perform logical backup                    |             |                             | Table, User, or Database mode             | One of the export options, depending on which data needs to be backed up |

*Continued*

Table 15-1 (continued)

| <i>Scenario</i>                                                | <i>CTAS</i> | <i>ALTER TABLE<br/>MOVE</i> | <i>Export</i>                              | <i>Recommended Option</i>                                                         |
|----------------------------------------------------------------|-------------|-----------------------------|--------------------------------------------|-----------------------------------------------------------------------------------|
| Repeat test runs                                               |             |                             | Database or<br>Tablespace<br>mode          | Tablespace mode will be<br>the fastest and easiest                                |
| Create default<br>database values                              |             |                             | Database, User,<br>or Table mode           | One of the export<br>options, depending on<br>which data needs to be<br>backed up |
| Create sample<br>data for testing<br>using subset of<br>values |             |                             | Table or User<br>mode with<br>Query Option | CTAS command                                                                      |

## Export types

Two basic export types are available in Oracle — Conventional Path and Direct Path.

### Conventional Path

The Conventional Path is the default export method for Oracle. The Conventional Path writes to export files by reading data into the database the same way that regular SQL statements are processed. A SQL SELECT statement is used to retrieve the data into the cache. The data is transferred to the evaluation buffer, evaluated, and then transferred to the client process, which writes the data to the export file. The data on the blocks will be reorganized before it is passed back to the client. When using this method, no special consideration needs to be given to the character set of the client or database because the Import process does the translation.

### Direct Path

The Direct Path method is much faster because it can bypass or skip two steps required in the Conventional Path export. The Direct Path export skips the SQL COMMAND processing layer, reading blocks directly from files into the database buffer cache. Blocks are then transferred directly to the export process, skipping the evaluation buffer. Because the blocks are in a known format, Oracle can write them directly to disk. Because no conversion of the block data is performed, the character sets of the database and the client process must be the same.



Chapter 20 discusses database and client-side character sets.

Direct Path exports will complete much quicker than Conventional Path exports. The Import utility is indifferent to the export method used to generate the export file.

## Exporting Data Using the Export Utility

Three ways exist to invoke the Export utility: Command Line mode, Interactive mode, and through Enterprise Manager. The Command Line mode is the most powerful and most common method used. It offers the full range of export parameters and has the ability to be invoked through scripts for regularly scheduling exports. Although Interactive mode is conceptually very easy to use, it does not support all available export parameters. Table 15-2 lists export parameters as well as the ones available to the Interactive mode (IM).

The export utility normally exports schemas in alphabetical order, and within the schemas, the tables in alphabetical order, with the index definitions following each table. This effectively ignores many times the relationships, especially in complex databases, with complicated dependencies. Also some of the grants can be lost during user or table exports. Because of that, DBA's use a number of methods, when importing, in order to restore fully all these dependencies.

**Table 15-2**  
**Export Utility Parameters**

| <i>Parameter</i> | <i>Default</i> | <i>IM</i> | <i>Meaning</i>                                                                                                                                                                                                                                                                                        |
|------------------|----------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| USERID           |                | (         | Oracle username and password for connecting to the database. If not supplied, Oracle will prompt for the password. If performing FULL database exports, must be a privileged user. To connect as a SYSDBA user, the connect string must be quoted.<br><br>Example:<br>USERID="sys/password as sysdba" |
| BUFFER           | OS DEP         | (         | Specifies the number of rows to be processed in an array fetch by the export. If zero is specified, then only one row is processed for each array fetch.<br><br><i>Note:</i> The parameter is used only for Conventional Path exports and not Direct Path exports.                                    |

*Continued*

Table 15-2 (continued)

| <i>Parameter</i> | <i>Default</i> | <i>IM</i> | <i>Meaning</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------|----------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COMPRESS         | Y              | (         | <p>Merges all extents for a table definition into one. If a table is made up of 50 1MB extents, it will be compressed into one 50MB extent. If set to N, then the extents are not compressed.</p> <p><i>Note:</i> Even though the consolidation occurs during the import, the parameter can be specified only during the export. If the parameter is set to NO, the size of the extents is determined from the table definitions.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CONSISTENT       | N              |           | <p>If Y, then the export runs as a READ ONLY transaction. The export will not export rows that have changed since the export started. If the export starts at 9:00 a.m. and a row that the export has not yet read is changed at 9:01, when it reads the row, it must read an image of the row as of 9:00 a.m.; therefore, it reads the before image from the rollback segment. If the rollback info does not exist, the export returns an error.</p> <p>If N, then the export simply reads the block as is. This can lead to inconsistent data because some blocks that changed after the export started will be recorded in the export file. From the previous example, the change at 9:01 would be in the export file. This can be problematic.</p> <p>Avoid setting the value to Y. Try getting the tables or database in a quiescent state before starting the export. Doing so will eliminate the “snapshot too old” errors that occur when rollback segments cannot be accessed.</p> |
| CONSTRAINTS      | Y              |           | Specifies that table constraints will or will not be exported.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| DIRECT           | N              |           | A value of Y specifies Direct Path exports. Direct Path is available only through the Command Line mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| <b>Parameter</b> | <b>Default</b>  | <b>IM</b> | <b>Meaning</b>                                                                                                                                                                                                                                                                                                                                                                            |
|------------------|-----------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FEEDBACK         | 0               |           | A nonzero value indicates that a progress meter should be used. A value of 10 means that a dot will appear after every 10 rows have been exported. This means that DBAs can check the progress of an export easily.                                                                                                                                                                       |
| FILE             | Expdat.dmp      | (         | Specifies the binary filename for the export file. If the FILESIZE parameter is used, when this limit is reached, the export stops writing to the current file and starts writing to another file. If not enough filenames are specified, the export will prompt the user for a new filename.                                                                                             |
| FILESIZE         | 0               |           | Specifies the maximum size for an export file. When this threshold is reached, the export stops writing to the current file and starts writing to the next file in the list of files from the FILE parameter. The value can be specified in bytes (B), kilobytes (K), megabytes (M) or gigabytes (G). This option allows the user to overcome maximum file size limits imposed by the OS. |
| FULL             | N               |           | Perform full database export for Database mode exports. Cannot be used in conjunction with Table or User parameters. The USERID performing the export must have the EXP_FULL_DATABASE role enabled.                                                                                                                                                                                       |
| GRANTS           | Y               | (         | Export Object grants.                                                                                                                                                                                                                                                                                                                                                                     |
| HELP             | N               |           | A value of Y displays a help window listing all available parameters. When this option is specified, all other parameters are disabled.                                                                                                                                                                                                                                                   |
| INCTYPE          | none            |           | Specifies type of incremental export in which the list of possible values consists of COMPLETE, CUMULATIVE, and INCREMENTAL.                                                                                                                                                                                                                                                              |
| INDEXES          | Y               |           | Specifies that indexes should or should not be exported.                                                                                                                                                                                                                                                                                                                                  |
| LOG              | none            |           | Specifies a log file name of the export messages.                                                                                                                                                                                                                                                                                                                                         |
| OWNER            | Value of USERID | (         | A list of users for User Mode exports. Cannot be used in conjunction with Table Mode, Tablespace Mode or Database Mode exports.                                                                                                                                                                                                                                                           |

Continued

Table 15-2 (continued)

| <i>Parameter</i> | <i>Default</i>  | <i>IM</i> | <i>Meaning</i>                                                                                                                                                                                                                                                                       |
|------------------|-----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PARFILE          |                 |           | File for storing export parameters that can be used at runtime. Export parameters must be specified one per line in the file and the file must be an ASCII text file. This simplifies the export, making it easier to change and to script.                                          |
| QUERY            |                 |           | Allows a WHERE clause to be included in table-level exports only. The QUERY is applied to all tables specified by the parameter TABLE and must be applicable to all tables specified.<br><br>Example:<br><code>QUERY="where city='New York'"</code>                                  |
| RECORD           | Y               |           | If Y, a record of the incremental export is recorded in the data dictionary.                                                                                                                                                                                                         |
| RECORDLENGTH     | OS<br>DEPENDANT |           | Specifies the length in bytes for the file record. This is required if the export file is being transferred to a different operating system that has a different default value.                                                                                                      |
| ROWS             | Y               | (         | Specifies whether the data is to be exported with the tables.                                                                                                                                                                                                                        |
| STATISTICS       | ESTIMATE        |           | Specifies the type of statistics to be taken during the import. Valid values are COMPUTE, ESTIMATE, and NONE.                                                                                                                                                                        |
| TABLES           |                 | (         | Indicates that the export is a Table mode export. Cannot be used in conjunction with Tablespace Mode, User Mode, or DATABASE MODE exports. If a list of tables is being specified, enclose the list in parentheses.<br><br>Example:<br><code>TABLES=(ORACLE.EMP, ORACLE.DEPT)</code> |
| TABLESPACES      |                 |           | List of tablespaces to be exported for Tablespace mode exports. Can be used only if the TRANSPORT_TABLESPACE is set to Y. Cannot be used in conjunction with Table Mode, User Mode, or Database Mode exports.                                                                        |

| <i>Parameter</i>         | <i>Default</i> | <i>IM</i> | <i>Meaning</i>                                              |
|--------------------------|----------------|-----------|-------------------------------------------------------------|
| TRANSPORT_<br>TABLESPACE | N              |           | Indicates that a Tablespace Mode export is to be used.      |
| VOLSIZE                  |                |           | Specifies the maximum export file size on each tape volume. |

## Interactive mode

The Interactive mode is maintained by Oracle primarily for backward compatibility. Oracle is no longer enhancing this method, which means that all the new features of the Export utility are not incorporated. They are available only through the Command Line mode.

The Interactive mode is very easy to use but has its limits. It is invoked by simply issuing the “EXP” or “EXP USERID” command from an operating system prompt. The user simply answers a series of questions when prompted and Oracle does the rest. Each question has a default value that the user can accept by simply pressing Enter when prompted. This mode supports only Conventional Path exports. The parameters that can be changed during an Interactive Mode export are as follows:

- ♦ BUFFER
- ♦ FILE
- ♦ MODE (TABLE or USER)
- ♦ GRANTS
- ♦ ROWS
- ♦ COMPRESS

Here is an example of an Interactive Mode export. The @@< indicates when the utility prompts the user for an answer.

```
C:\>EXP student@certdb @@< 1
```

```
Export: Release 8.1.6.0.0 - Production on Sun Mar 11 11:07:28 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Password: @@< 2
```

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
```

```

Enter array fetch buffer size: 4096 > @@< 3

Export file: EXPDAT.DMP > sample.dmp @@< 4

(2)U(sers), or (3)T(ables): (2)U > 2 @@< 5

Export grants (yes/no): yes > no @@< 6

Export table data (yes/no): yes > @@< 7

Compress extents (yes/no): yes > no @@< 8

Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
Note: grants on tables/views/sequences/roles will not be exported
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user STUDENT
. exporting object type definitions for user STUDENT
About to export STUDENT's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export STUDENT's tables via Conventional Path ...
. . exporting table AUD_SAL 1 rows exported
. . exporting table BATCHJOBS 3 rows exported
. . exporting table CLASSENROLLMENT 7 rows exported
. . exporting table COURSEAUDIT 0 rows exported
. . exporting table COURSES 9 rows exported
. . exporting table EMPLOYEES 1 rows exported
. . exporting table INSTRUCTORS 8 rows exported
. . exporting table LOCATIONS 3 rows exported
. . exporting table SCHEDULEDCLASSES 3 rows exported
. . exporting table STUDENTS 11 rows exported
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting snapshots
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully without warnings.

C:\>DIR SAMPLE.DMP

```

```
Volume in drive C has no label.
Volume Serial Number is EC56-2FC2
```

```
Directory of C:\
```

```
03/11/2001 11:08a 22,528 sample.dmp
 1 File(s) 22,528 bytes
 0 Dir(s) 1,422,536,704 bytes free
```

The prompts have the following meaning:

1. USERID connecting to the database for the export
2. Password for user connecting
3. BUFFER size parameter with a default of 4096 bytes
4. FILE parameter indicates the export file name
5. TABLE or USER mode export
6. GRANTS export Yes or No with a default value of Yes
7. ROWS prompt to include table data
8. COMPRESS to compress table extents into one extent

If the export completes successfully, a file named sample.dmp will be located in the directory specified by the FILE parameter containing all the data specified by the export.

## Command Line mode

The Command Line mode requires at least one parameter to be specified at runtime. This mode supports all available parameters of the utility. This is the most common method and is often used with the PARFILE parameter for ease of execution. To obtain a list of available parameters, execute the following command:

```
C:\>EXP help=y
```

```
Export: Release 8.1.6.0.0 - Production on Sun Mar 11 15:22:51 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

You can let Export prompt you for parameters by entering the EXP command followed by your username/password:

```
Example: EXP SCOTT/TIGER
```

Or, you can control how Export runs by entering the EXP command followed

by various arguments. To specify parameters, you use keywords:

```
Format: EXP KEYWORD=value or KEYWORD=(value1,value2,...,valueN)
Example: EXP SCOTT/TIGER GRANTS=Y TABLES=(EMP,DEPT,MGR)
 or TABLES=(T1:P1,T1:P2), if T1 is partitioned table
```

USERID must be the first parameter on the command line.

| Keyword     | Description (Default)                            | Keyword      | Description (Default)      |
|-------------|--------------------------------------------------|--------------|----------------------------|
| USERID      | username/password                                | FULL         | export entire file (N)     |
| BUFFER      | size of data buffer                              | OWNER        | list of owner usernames    |
| FILE        | output files (EXPDAT.DMP)                        | TABLES       | list of table names        |
| COMPRESS    | import into one extent (Y)                       | RECORDLENGTH | length of IO record        |
| GRANTS      | export grants (Y)                                | INCTYPE      | incremental export type    |
| INDEXES     | export indexes (Y)                               | RECORD       | track incr. export (Y)     |
| ROWS        | export data rows (Y)                             | PARFILE      | parameter filename         |
| CONSTRAINTS | export constraints (Y)                           | CONSISTENT   | cross-table consistency    |
| LOG         | log file of screen output                        | STATISTICS   | analyze objects (ESTIMATE) |
| DIRECT      | direct path (N)                                  | TRIGGERS     | export triggers (Y)        |
| FEEDBACK    | display progress every x rows (0)                |              |                            |
| FILESIZE    | maximum size of each dump file                   |              |                            |
| QUERY       | select clause used to export a subset of a table |              |                            |

The following keywords apply only to transportable tablespaces  
 TRANSPORT\_TABLESPACE export transportable tablespace metadata (N)  
 TABLESPACES list of tablespaces to transport

Export terminated successfully without warnings.

**Here is an example of a parameter file that you can use with the Export utility. This is a Table Mode export, exporting the LOCATIONS, INSTRUCTORS, and COURSES tables. The file is called exppar.par.**

```
USERID = STUDENT
TABLES = (LOCATIONS, INSTRUCTORS, COURSES)
DIRECT = Y
FILE = SAMPLE.DMP
ROWS = Y
GRANTS = Y
INDEXES = N
```

**To run the Export utility using this parameter file, simply type the following command:**

```
C:\>EXP parfile=exppar.par
```

```
Export: Release 8.1.6.0.0 - Production on Sun Mar 11 15:28:26 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Password:
```

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
```

```

With the Partitioning option
JServer Release 8.1.6.0.0 - Production
Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
Note: indexes on tables will not be exported

```

```

About to export specified tables via Direct Path ...
. . exporting table LOCATIONS 3 rows exported
. . exporting table INSTRUCTORS 8 rows exported
. . exporting table COURSES 9 rows exported
Export terminated successfully without warnings.

```



You can include the password in the parameter file, which means that the user will not be prompted for it, but doing so is a dangerous security breach. Storing passwords in script files is never a good idea.

To perform a FULL database export, simply set the FULL parameter to Y. Note that the user must have the EXP\_FULL\_DATABASE role enabled to perform this type of export. Also, it cannot be used in conjunction with Table, User, or Tablespace mode exports. The pound sign (#) signifies a comment, which means that the line is ignored by the export.

```

USERID = "SYS AS SYSDBA"
#TABLES = (LOCATIONS, INSTRUCTORS, COURSES)
DIRECT = Y
FILE = SAMPLE.DMP
ROWS = Y
GRANTS = Y
INDEXES = N
FULL = Y
FEEDBACK = 5

```

## Exporting using Oracle Enterprise Manager

To use the Export utility through Oracle Enterprise Manager, you need to perform the following steps.

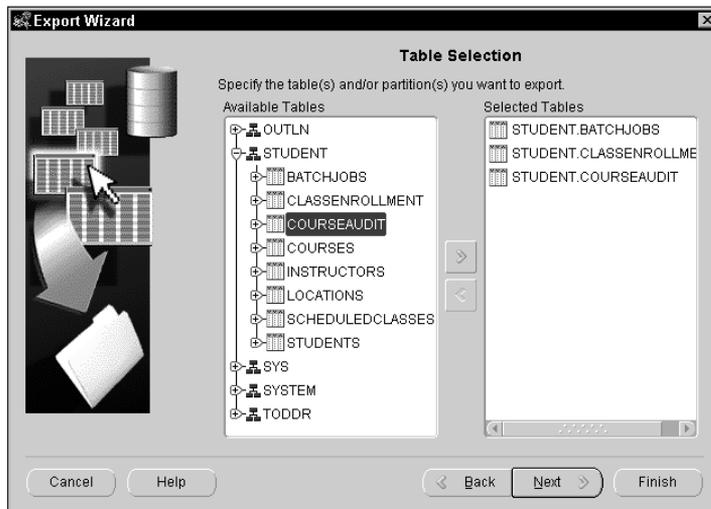
### STEP BY STEP: Exporting Using Oracle Enterprise Manager

1. Launch the Oracle Enterprise Manager Console.
2. Enter the administrator username and password for the Management Server.
3. Expand the Database Folder.
4. Right-click the database to run the export against. Be sure to set the preferred credentials for the Node before starting the export. This is required because the export is going to create a file on the OS.
5. Specify the type of export as show in Figure 15-2.



**Figure 15-2:** Select export type

6. Enter the file to be used as the export file as shown in Figure 15-3.
7. Specify the objects or users to export, depending on the mode selected in Step 5.



**Figure 15-3:** Select export objects

8. Figure 15-4 lists the associated objects depending on the mode selected. Specify the associated objects, depending on the mode selected in Step 5.



Figure 15-4: Specify associated objects

9. Figure 15-5 is a screen shot of the parameters screen. Specify schedule parameters.



Figure 15-5: Schedule job

## Importing Data Using the Import Utility

The Import utility works only when a file has been generated from the Export utility. It does not need to be from the same OS or version of Oracle database, but it must have been generated by the Export utility. When performing imports, users must be aware of things such as privileges, space in the database, and constraints in order to perform successful imports. These issues are discussed in the “Import behavior” section. The Import utility has the same three methods as the Export utility: Interactive mode, Command Line mode, and through Oracle Enterprise Manager. As with the Export, the Command Line mode is the most popular as well as the most flexible. Table 15-3 lists the Import parameters as well as the ones available to the Interactive Mode (IM).

**Table 15-3**  
**Import Utility Parameters**

| <i>Parameter</i> | <i>Default</i> | <i>IM</i> | <i>Meaning</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------|----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| USERID           |                | (         | Specifies the username performing the import. Depending on the type of import being performed, this may need to be a privileged user. A privileged user is one with the IMP_FULL_DATABASE role enabled.<br><br>Example of connecting as a privileged user:<br>USERID="sys/password as sysdba"                                                                                                                                                                      |
| ANALYZE          | Y              |           | Specifies whether the import executes an ANALYZE command during the import or loads statistics taken from the export file.                                                                                                                                                                                                                                                                                                                                         |
| BUFFER           | OS DEP         | (         | Specifies the size in bytes in an array insert operation by the import.<br><br><i>Note:</i> The parameter is used only for Conventional Path exports and not Direct Path exports.                                                                                                                                                                                                                                                                                  |
| COMMIT           | N              |           | When set to Y, a commit is issued after the number of rows inserted determined by the parameter BUFFER has been reached. A commit is always issued when all rows for a table have been loaded. The benefit of setting the BUFFER parameter and the COMMIT parameter to Y is that less rollback space will be required. Without setting these parameters, if a table has two million rows, a commit will be issued only after the two million rows have been loaded |

| <b>Parameter</b> | <b>Default</b> | <b>IM</b> | <b>Meaning</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONSTRAINTS      | Y              |           | Specifies that table constraints will or will not be imported                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| DESTROY          | N              |           | <p>Specifies that existing datafiles for a tablespace should be reused. The tablespace will be dropped first, destroying the contents of the tablespace.</p> <p>Ensure that the parameter is set to N when creating another database on the same machine because the creation of the tablespaces will destroy the existing database files if on the same machine.</p> <p>This parameter should be used only if a database is being rebuilt.</p>                                                                                              |
| FEEDBACK         | 0              |           | A non-zero value indicates that a progress meter should be used. A value of 10 means that a dot will appear after every 10 rows have been imported. This means that DBAs can check the progress of an import easily.                                                                                                                                                                                                                                                                                                                         |
| FILE             | Expdat.dmp     | (         | <p>Specifies the binary file name for the import file. The file must have been created with the Export utility. If the FILESIZE parameter is used, when this limit is reached, the export process stops writing to the current file and starts writing to another file. If not enough filenames are specified, the export will prompt the user for a new filename.</p> <p>The user performing the import does need to be the same as the user who performed the export; however, the import user must have read permissions on the file.</p> |
| FILESIZE         | 0              |           | Specifies the maximum dump file size for the export. This tells the import how big the export files are so that it knows when to switch.                                                                                                                                                                                                                                                                                                                                                                                                     |
| FROMUSER         |                |           | This is a comma-separated list of users who own objects in the export file and whose ownership of those objects is to change on the import. This is used in conjunction with the TOUSER parameter for changing ownership of objects on the import. Must be a privileged user to perform this type of import.                                                                                                                                                                                                                                 |

*Continued*

Table 15-3 (continued)

| <i>Parameter</i> | <i>Default</i> | <i>IM</i> | <i>Meaning</i>                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FULL             | N              |           | Specifies that the import should import the entire file.                                                                                                                                                                                                                                                                                                                                                                                                     |
| GRANTS           | Y              | (         | Import object grants from the export file. If this is a USER mode import, it imports grants only on objects owned by that user.                                                                                                                                                                                                                                                                                                                              |
| HELP             | N              |           | A value of Y displays a help window listing all available parameters. When this option is specified, all other parameters are disabled.                                                                                                                                                                                                                                                                                                                      |
| IGNORE           | N              |           | <p>If set to Y, table creation errors are not reported. If the table already exists, it simply loads the rows or data. If set to N, the errors are reported and logged and the table is then skipped by the import, which means that no data will be loaded.</p> <p>Generally, if re-creating objects to change storage parameters or tablespace locations, re-create the objects and then set the IGNORE flag = Y so that only the data will be loaded.</p> |
| INCTYPE          | none           |           | Specifies type of incremental import where the list of possible values are: SYSTEM which tells the import to load a new version of the SYSTEM objects and RESTORE which imports all objects in the export file except for SYSTEM objects.                                                                                                                                                                                                                    |
| INDEXES          | Y              |           | Specifies that indexes should or should not be imported. If set to N, then the indexes will be created after the data is imported. If set to Y, the indexes are maintained during the import, which is usually slower than creating them after the import has completed.                                                                                                                                                                                     |
| INDEXFILE        | none           |           | This is a filename for storing the index creation syntax. The file is an ASCII file, which means that it can be modified. The file contains the create index syntax as well as all the storage parameters for the indexes, so changing them in the file and running the file after the import has completed can be very beneficial.                                                                                                                          |

| <i>Parameter</i>        | <i>Default</i>  | <i>IM</i> | <i>Meaning</i>                                                                                                                                                                                                                                                                                                                                             |
|-------------------------|-----------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         |                 |           | <p>The script also contains the CREATE TABLE syntax and all the storage parameters for the table. The commands are commented out. The comments can be removed to create a script to create all tables and indexes for a database.</p> <p>If the CONSTRAINTS parameter was set, the file will also contain all the syntax for creating the constraints.</p> |
| LOG                     | none            |           | Specifies a log file name of the import messages.                                                                                                                                                                                                                                                                                                          |
| PARFILE                 |                 |           | File for storing import parameters that can be used at runtime. This simplifies the import, making it easier to change and to script.                                                                                                                                                                                                                      |
| RECALCULATE _STATISTICS | N               |           | A value of Y causes object statistics to be generated during the import. If set to N, no statistics will be generated. If not generating statistics during the import, they can be loaded from the export file.                                                                                                                                            |
| RECORD                  | Y               |           | If Y, a record of the INCREMENTAL export is recorded in the data dictionary.                                                                                                                                                                                                                                                                               |
| RECORDLENGTH            | OS<br>DEPENDENT |           | <p>Specifies the length in bytes for the file record.</p> <p>This is required if the import file is being transferred from an operating system that has a different default value.</p>                                                                                                                                                                     |
| ROWS                    | Y               | (         | Specifies whether the data is to be imported with the tables.                                                                                                                                                                                                                                                                                              |
| SHOW                    | N               |           | If Y, the contents of the file are displayed and not actually loaded into the database.                                                                                                                                                                                                                                                                    |
| TABLES                  |                 | (         | <p>Specifies a list of tables to be imported. An asterisk (*) specifies all tables in the export file. When used, this initiates a table mode import.</p> <p>Example:</p> <p>TABLES=(ORACLE.EMP, ORACLE.DEPT)</p>                                                                                                                                          |

*Continued*

Table 15-3 (continued)

| <i>Parameter</i>     | <i>Default</i> | <i>IM</i> | <i>Meaning</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|----------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TABLESPACES          |                |           | List of tablespaces to be imported when the TRANSPORT_TABLESPACE is set to Y. Cannot be used in conjunction with Table Mode, User Mode or Database Mode exports.                                                                                                                                                                                                                                                                                                                                                                                        |
| TOUSER               | none           |           | Specifies a list of users who will become the owner of objects from the users specified by the FROMUSER parameter. If the FROMUSER parameter specifies multiple users and the TOUSER has only one, the one user becomes the owner of all the objects from those users. If TOUSER has the same number of users as FROMUSER, the first user in the TOUSER will become the owner of objects owned by the first user referenced by the FROMUSER parameter.<br><br>This process of changing ownership applies only to objects referenced in the import file. |
| TRANSPORT_TABLESPACE | N              |           | Indicates that a Tablespace ModeODE import is being performed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| TTS_OWNERS           | none           |           | List of users who own objects in the transportable tablespace set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| VOLSIZE              |                |           | Specifies the maximum import file size on each tape volume.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## Interactive mode

To run an Interactive mode import, simply type **imp** or **imp userid** from the OS prompt. The user will then be prompted with several questions. Each question has a default value that can be accepted by simply pressing Enter at the prompt. As with the Interactive Export, a limited number of parameters can be changed.

Here is the list of parameters for Interactive mode imports that can be changed:

- ♦ FILE
- ♦ BUFFER
- ♦ SHOW
- ♦ IGNORE
- ♦ GRANTS

- ♦ ROWS
- ♦ FULL

Here is an example of an Interactive mode import. The @@< indicates when the utility prompts the user for an answer.

```
C:\>IMP
Import: Release 8.1.6.0.0 - Production on Thu Mar 15 15:51:18 2001
(c) Copyright 1999 Oracle Corporation. All rights reserved.
Username: student @@< 1
Password: @@< 2
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
Import file: EXPDAT.DMP > @@< 3
Enter insert buffer size (minimum is 8192) 30720> @@< 4
Export file created by EXPORT:V08.01.06 via conventional path
Warning: the objects were exported by SCOTT, not by you
import done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
List contents of import file only (yes/no): no > n @@< 5
Ignore create error due to object existence (yes/no): no > n @@< 6
Import grants (yes/no): yes > y @@< 7
Import table data (yes/no): yes > n @@< 8
Import entire export file (yes/no): no > y @@< 9
. importing SCOTT's objects into STUDENT
About to enable constraints...
Import terminated successfully without warnings.
```

The prompts have the following meaning:

1. USERID connecting to the database for the import
2. Password for user connecting
3. FILE parameter indicates the import file name
4. BUFFER size parameter with a default of 8192 bytes
5. SHOW when set to Y lists the import file contents only

6. IGNORE parameter used to ignore or continue processing if there are DDL creation errors
7. GRANTS import Yes or No with a default value of Yes
8. ROWS prompt to include table data when Y or just create table(s) without loading the data when N
9. Import TYPE—Y will import the entire file and N will allow for USER or TABLE mode imports.

## Command Line mode

As does the Export utility, the Command Line mode of the Import utility takes advantage of all the Import parameters, making it more flexible and powerful than the Interactive mode. A help facility is available by simply using the HELP parameter at runtime. The Help Mode lists the parameters as well as the default values.

```
C:\>IMP HELP=Y
```

```
Import: Release 8.1.6.0.0 - Production on Thu Mar 15 16:19:54 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

You can let Import prompt you for parameters by entering the IMP command followed by your username/password:

```
Example: IMP SCOTT/TIGER
```

Or, you can control how Import runs by entering the IMP command followed by various arguments. To specify parameters, you use keywords:

```
Format: IMP KEYWORD=value or KEYWORD=(value1,value2,...,valueN)
```

```
Example: IMP SCOTT/TIGER IGNORE=Y TABLES=(EMP,DEPT) FULL=N
or TABLES=(T1:P1,T1:P2), if T1 is partitioned table
```

USERID must be the first parameter on the command line.

| Keyword | Description (Default)             | Keyword      | Description (Default)   |
|---------|-----------------------------------|--------------|-------------------------|
| USERID  | username/password                 | FULL         | import entire file (N)  |
| BUFFER  | size of data buffer               | FROMUSER     | list of owner usernames |
| FILE    | input files (EXPDAT.DMP)          | TOUSER       | list of usernames       |
| SHOW    | just list file contents (N)       | TABLES       | list of table names     |
| IGNORE  | ignore create errors (N)          | RECORDLENGTH | length of IO record     |
| GRANTS  | import grants (Y)                 | INCTYPE      | incremental import type |
| INDEXES | import indexes (Y)                | COMMIT       | commit array insert (N) |
| ROWS    | import data rows (Y)              | PARFILE      | parameter filename      |
| LOG     | log file of screen output         | CONSTRAINTS  | import constraints (Y)  |
| DESTROY | overwrite tablespace datafile (N) |              |                         |

```

INDEXFILE write table/index info to specified file
SKIP_UNUSABLE_INDEXES skip maintenance of unusable indexes (N)
ANALYZE execute ANALYZE statements in dump file (Y)
FEEDBACK display progress every x rows(0)
TOID_NOVALIDATE skip validation of specified type ids
FILESIZE maximum size of each dump file
RECALCULATE_STATISTICS recalculate statistics (N)

```

```

The following keywords apply only to transportable tablespaces
TRANSPORT_TABLESPACE Import transportable tablespace metadata (N)
TABLESPACES Tablespaces to be transported into database
DATAFILES Datafiles to be transported into database
TTS_OWNERS Users who own data in the transportable tablespace set

```

Import terminated successfully without warnings.

**To perform an import, a valid export file must exist. Understanding the behavior of imports, such as which objects are created first and what privileges are required to perform the import, are very important to understand and are covered in the “Import behavior” section.**

**The first example simply lists the contents of an import file. This is accomplished by setting the SHOW parameter to Y.**

```
C:\>IMP USERID=STUDENT FILE=EXPDAT.DMP SHOW=Y
```

```
Import: Release 8.1.6.0.0 - Production on Thu Mar 15 16:28:13 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Password:
```

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
```

```

Export file created by EXPORT:V08.01.06 via conventional path
import done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
. importing STUDENT's objects into STUDENT
"CREATE TABLE "AUD_SAL" ("C_DATE" DATE, "C_USER" VARCHAR2(30), "EMPNO" NUMBE
"R(10, 0), "BEFORE_SALARY" NUMBER(10, 0), "NEW_SALARY" NUMBER(10, 0)) PCTFR
"EE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 40960 NEXT
" 40960 MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROU
"PS 1 BUFFER_POOL DEFAULT) TABLESPACE "CERTDB"
. . skipping table "AUD_SAL"

"CREATE TABLE "BATCHJOBS" ("JOBID" NUMBER(6, 0) NOT NULL ENABLE, "JOBNAME" V
"ARCHAR2(30) NOT NULL ENABLE, "STATUS" VARCHAR2(30) NOT NULL ENABLE, "LASTUP
"DATED" DATE NOT NULL ENABLE) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
" LOGGING STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505 PCTIN
"CREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "CE"

```

```

"RTDB"
. . skipping table "BATCHJOBS"

"CREATE UNIQUE INDEX "BATCHJOBS_JOBID_PK" ON "BATCHJOBS" ("JOBID") PCTFREE"
" 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 M"
"AXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFA"
"ULT) TABLESPACE "CERTDB" LOGGING"
"CREATE TABLE "BONUS" ("ENAME" VARCHAR2(10), "JOB" VARCHAR2(9), "SAL" NUMBER"
", "COMM" NUMBER) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STO"
"RAGE(INITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREAS"
"E 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "SYSTEM"
. . skipping table "BONUS"

"CREATE TABLE "CLASSENROLLMENT" ("CLASSID" NUMBER(*,0) NOT NULL ENABLE, "STU"
"DENTNUMBER" NUMBER(*,0) NOT NULL ENABLE, "STATUS" CHAR(10) NOT NULL ENABLE,"
" "ENROLLMENTDATE" DATE NOT NULL ENABLE, "PRICE" NUMBER(9, 2) NOT NULL ENABL"
"E, "GRADE" CHAR(4), "COMMENTS" VARCHAR2(2000)) PCTFREE 10 PCTUSED 40 INITR"
"ANS 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MA"
"XEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAU"
"LT) TABLESPACE "CERTDB"
. . skipping table "CLASSENROLLMENT"

"CREATE UNIQUE INDEX "PK_CLASSID_STUDENTNUMBER" ON "CLASSENROLLMENT" ("CLASS"
"ID" , "STUDENTNUMBER") PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL"
" 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FR"
"EELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "CERTDB" LOGGING"
"CREATE TABLE "COURSEAUDIT" ("COURSENUMBER" NUMBER(*,0) NOT NULL ENABLE, "CH"
"ANGE" VARCHAR2(30) NOT NULL ENABLE, "DATECHANGED" DATE NOT NULL ENABLE, "PR"
"ICE" NUMBER(9, 2), "CHANGEDBY" VARCHAR2(15)) PCTFREE 10 PCTUSED 40 INITRAN"
"S 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXE"
"XTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT"
") TABLESPACE "CERTDB"
. . skipping table "COURSEAUDIT"

"CREATE UNIQUE INDEX "COURSEAUDIT_PK" ON "COURSEAUDIT" ("COURSENUMBER" , "CH"
"ANGE" , "DATECHANGED") PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL"
" 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FR"
"EELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "CERTDB" LOGGING"
"CREATE TABLE "COURSES" ("COURSENUMBER" NUMBER(*,0) NOT NULL ENABLE, "COURSE"
"NAME" VARCHAR2(200) NOT NULL ENABLE, "REPLACESCOURSE" NUMBER(*,0), "RETAILP"
"RICE" NUMBER(9, 2) NOT NULL ENABLE, "DESCRIPTION" VARCHAR2(2000)) PCTFREE "
"10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 40960 NEXT 40"
"960 MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS "
"1 BUFFER_POOL DEFAULT) TABLESPACE "CERTDB"
. . skipping table "COURSES"

"CREATE UNIQUE INDEX "PK_COURSENUMBER" ON "COURSES" ("COURSENUMBER") PCTFR"
"EE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1"
" MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DE"
"FAULT) TABLESPACE "CERTDB" LOGGING"
"CREATE TABLE "DEPT" ("DEPTNO" NUMBER(2, 0), "DNAME" VARCHAR2(14), "LOC" VAR"
"CHAR2(13)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAGE(I"

```

```
"NITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 50 F"
"REELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "SYSTEM"
. . skipping table "DEPT"
```

```
"CREATE UNIQUE INDEX "PK_DEPT" ON "DEPT" ("DEPTNO") PCTFREE 10 INITRANS 2 "
"MAXTRANS 255 STORAGE(INITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 21474"
"83645 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TAB"
"LESPACE "SYSTEM" LOGGING"
"CREATE TABLE "EMP" ("EMPNO" NUMBER(4, 0), "ENAME" VARCHAR2(10), "JOB" VARCH"
"AR2(9), "MGR" NUMBER(4, 0), "HIREDATE" DATE, "SAL" NUMBER(7, 2), "COMM" NUM"
"BER(7, 2), "DEPTNO" NUMBER(2, 0)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRAN"
"S 255 LOGGING STORAGE(INITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 2147"
"483645 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TA"
"BLESPACE "SYSTEM"
. . skipping table "EMP"
```

```
"CREATE UNIQUE INDEX "PK_EMP" ON "EMP" ("EMPNO") PCTFREE 10 INITRANS 2 MAX"
"TRANS 255 STORAGE(INITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 21474836"
"45 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLES"
"PACE "SYSTEM" LOGGING"
"CREATE TABLE "EMPLOYEES" ("EMPNO" NUMBER(10, 0), "SALARY" NUMBER(10, 0) NOT"
" NULL ENABLE) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAG"
"E(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREEL"
"ISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "CERTDB"
. . skipping table "EMPLOYEES"
```

```
"CREATE TABLE "INSTRUCTORS" ("INSTRUCTORID" NUMBER(*,0) NOT NULL ENABLE, "SA"
"LUTATION" CHAR(4), "LASTNAME" VARCHAR2(30) NOT NULL ENABLE, "FIRSTNAME" VAR"
"CHAR2(30) NOT NULL ENABLE, "MIDDLEINITIAL" VARCHAR2(5), "ADDRESS1" VARCHAR2"
"(50), "ADDRESS2" VARCHAR2(50), "CITY" VARCHAR2(30), "STATE" CHAR(2), "COUNT"
"RY" VARCHAR2(30), "POSTALCODE" CHAR(10), "OFFICEPHONE" CHAR(15), "HOMEPHONE"
" " CHAR(15), "CELLPHONE" CHAR(15), "EMAIL" VARCHAR2(50), "INSTRUCTOR" TYPE "CH"
"AR(10) NOT NULL ENABLE, "PERDIEMCOST" NUMBER(9, 2), "PERDIEMEXPENSES" NUMBE"
"R(9, 2), "COMMENTS" VARCHAR2(2000)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTR"
"ANS 255 LOGGING STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 50"
"5 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESP"
"ACE "CERTDB"
. . skipping table "INSTRUCTORS"
```

```
"CREATE UNIQUE INDEX "PK_INSTRUCTORID" ON "INSTRUCTORS" ("INSTRUCTORID") P"
"CTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTEN"
"TS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POO"
"L DEFAULT) TABLESPACE "CERTDB" LOGGING"
"CREATE TABLE "LOCATIONS" ("LOCATIONID" NUMBER(*,0) NOT NULL ENABLE, "LOCATI"
"ONNAME" VARCHAR2(50) NOT NULL ENABLE, "ADDRESS1" VARCHAR2(50), "ADDRESS2" V"
"ARCHAR2(50), "CITY" VARCHAR2(30), "STATE" CHAR(2), "COUNTRY" VARCHAR2(30), "
" " "POSTALCODE" CHAR(10), "TELEPHONE" CHAR(15), "FAX" CHAR(15), "CONTACT" VARC"
"HAR2(50), "DESCRIPTION" VARCHAR2(2000)) PCTFREE 10 PCTUSED 40 INITRANS 1 M"
"AXTRANS 255 LOGGING STORAGE(INITIAL 65536 NEXT 40960 MINEXTENTS 1 MAXEXTENT"
"S 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TAB"
"LESPACE "TOOLS"
```

```

. . skipping table "LOCATIONS"

"GRANT ALTER ON "LOCATIONS" TO PUBLIC"
"GRANT DELETE ON "LOCATIONS" TO PUBLIC"
"GRANT INDEX ON "LOCATIONS" TO PUBLIC"
"GRANT INSERT ON "LOCATIONS" TO PUBLIC"
"GRANT SELECT ON "LOCATIONS" TO PUBLIC"
"GRANT UPDATE ON "LOCATIONS" TO PUBLIC"
"GRANT REFERENCES ON "LOCATIONS" TO PUBLIC"
"CREATE UNIQUE INDEX "PK_LOCATIONID" ON "LOCATIONS" ("LOCATIONID") PCTFREE"
" 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 M"
"AXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFA"
"ULT) TABLESPACE "CERTDB" LOGGING"
"CREATE TABLE "SALGRADE" ("GRADE" NUMBER, "LOSAL" NUMBER, "HISAL" NUMBER) P"
"CTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 65536 "
"NEXT 65536 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 50 FREELISTS 1 FR"
"EELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "SYSTEM"
. . skipping table "SALGRADE"

"CREATE TABLE "SCHEDULEDCLASSES" ("CLASSID" NUMBER(*,0) NOT NULL ENABLE, "CO"
"URSENUMBER" NUMBER(*,0) NOT NULL ENABLE, "LOCATIONID" NUMBER(*,0) NOT NULL "
"ENABLE, "CLASSROOMNUMBER" NUMBER(*,0) NOT NULL ENABLE, "INSTRUCTORID" NUMBE"
"R(*,0) NOT NULL ENABLE, "STARTDATE" DATE NOT NULL ENABLE, "DAYSURATION" NU"
"MBER(*,0) NOT NULL ENABLE, "STATUS" CHAR(10) NOT NULL ENABLE, "COMMENTS" VA"
"RCHAR2(2000)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAG"
"E(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREEL"
"ISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "CERTDB"
. . skipping table "SCHEDULEDCLASSES"

"CREATE UNIQUE INDEX "PK_CLASSID" ON "SCHEDULEDCLASSES" ("CLASSID") PCTFRE"
"E 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 "
"MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEF"
"AULT) TABLESPACE "CERTDB" LOGGING"
"CREATE TABLE "STUDENTS" ("STUDENTNUMBER" NUMBER(*,0) NOT NULL ENABLE, "SALU"
"TATION" CHAR(4), "LASTNAME" VARCHAR2(30) NOT NULL ENABLE, "FIRSTNAME" VARCH"
"AR2(30) NOT NULL ENABLE, "MIDDLEINITIAL" VARCHAR2(5), "ADDRESS1" VARCHAR2(5"
"0), "ADDRESS2" VARCHAR2(50), "CITY" VARCHAR2(30), "STATE" CHAR(2), "COUNTRY"
"" VARCHAR2(30), "POSTALCODE" CHAR(10), "HOMEPHONE" CHAR(15), "WORKPHONE" CH"
"AR(15), "EMAIL" VARCHAR2(50), "COMMENTS" VARCHAR2(2000)) PCTFREE 10 PCTUSE"
"D 40 INITRANS 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 40960 NEXT 40960 MINEX"
"TENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_"
"POOL DEFAULT) TABLESPACE "CERTDB"
. . skipping table "STUDENTS"

"CREATE UNIQUE INDEX "PK_STUDENTNUMBER" ON "STUDENTS" ("STUDENTNUMBER") PC"
"TFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTENT"
"S 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL"
" DEFAULT) TABLESPACE "CERTDB" LOGGING"
"CREATE PROCEDURE TEST_123"
"IS"
" V_VARI VARCHAR2(100);"
"BEGIN"

```

```

" SELECT COUNT(*)"
" INTO V_VARI"
" FROM LOCATIONS"
" WHERE 1=1;"
"END;"
"ALTER PROCEDURE "TEST_123" COMPILE_TIMESTAMP '2001-03-04:21:36:17'"
"CREATE TRIGGER "STUDENT".AUD_SAL_T"
" BEFORE"
" UPDATE ON EMPLOYEES"
" FOR EACH ROW"
" "
"BEGIN"
" INSERT INTO AUD_SAL VALUES (SYSDATE,USER,:OLD.EMPNO,:OLD.SALARY,:NEW.S"
"ALARY);"
" END;"
"ALTER TRIGGER "AUD_SAL_T" ENABLE"
"ALTER TABLE "BATCHJOBS" ENABLE CONSTRAINT "BATCHJOBS_JOBID_PK"
"ALTER TABLE "CLASSENROLLMENT" ENABLE CONSTRAINT "PK_CLASSID_STUDENTNUMBER"
"ALTER TABLE "COURSEAUDIT" ENABLE CONSTRAINT "COURSEAUDIT_PK"
"ALTER TABLE "COURSES" ENABLE CONSTRAINT "PK_COURSENUMBER"
"ALTER TABLE "DEPT" ENABLE CONSTRAINT "PK_DEPT"
"ALTER TABLE "EMP" ENABLE CONSTRAINT "PK_EMP"
"ALTER TABLE "INSTRUCTORS" ENABLE CONSTRAINT "PK_INSTRUCTORID"
"ALTER TABLE "LOCATIONS" ENABLE CONSTRAINT "PK_LOCATIONID"
"ALTER TABLE "SCHEDULEDCLASSES" ENABLE CONSTRAINT "PK_CLASSID"
"ALTER TABLE "STUDENTS" ENABLE CONSTRAINT "PK_STUDENTNUMBER"
"ALTER TABLE "CLASSENROLLMENT" ENABLE CONSTRAINT "FK_CLASSENROLLMENT_CLASSID"
""
"ALTER TABLE "CLASSENROLLMENT" ENABLE CONSTRAINT "FK_CLASSENROLLMENT_STUDENT"
"NUM"
"ALTER TABLE "COURSEAUDIT" ENABLE CONSTRAINT "FK_COURSEAUDIT_COURSENUMBER"
"ALTER TABLE "EMP" ENABLE CONSTRAINT "FK_DEPTNO"
"ALTER TABLE "SCHEDULEDCLASSES" ENABLE CONSTRAINT "FK_SCHEDCLASS_COURSENUM"
"ALTER TABLE "SCHEDULEDCLASSES" ENABLE CONSTRAINT "FK_SCHEDCLASSES_LOCATIONI"
"D"
"ALTER TABLE "SCHEDULEDCLASSES" ENABLE CONSTRAINT "FK_SCHEDCLASSES_INSTID"
Import terminated successfully without warnings.

```

The **SHOW** option lists the contents and also the order by which objects will be created and constraints will be enabled. The objects are not actually created.

The next example is used to import objects into the user **SCOTT**'s schema. The **FROMUSER** and **TOUSER** parameters will be used. Note that the user **SCOTT** must be the one performing the import or a privileged user. As with the export, the **PARFILE** parameter can be used to list parameters for the import. This simplifies the import process. Here is an example of the parameter file for performing this import:

```

C:\>type imppar.par
USERID=SCOTT
FILE=EXPDAT.DMP
FROMUSER=STUDENT

```

```
TOUSER=SCOTT
ROWS=N
GRANTS=N
CONSTRAINTS=N
```

To run the import, simply type **imp parfile=imppar.par** from the OS command prompt. In this example, the user is prompted for the password for the user SCOTT.

One of the other very powerful features of the Import utility is the INDEXFILE parameter. This parameter allows you to create an ASCII file with all the index information contained in the export file. This gives you a script for creating indexes. You also can modify the script to relocate objects. The file contains everything from the export file; however, only the index creation information is not commented out. The comments can be removed to give you a script for creating all objects referenced in the export file. If the file was created by a FULL database export, all the objects in the database will exist in the file. Following is a copy of the parameter file for performing this type of import. Note that the INDEXFILE parameter is set to OBJ\_INFO.SQL. This setting means that none of the tables or indexes will be created; only the OBJ\_INFO.SQL file will be created with all the information from the import file in ASCII form.

```
C:\>type imppar.par
USERID=SYSTEM
FILE=EXPDAT.DMP
ROWS=N
GRANTS=Y
CONSTRAINTS=Y
FULL=Y
INDEXFILE=OBJ_INFO.SQL

C:\>DIR OBJ_INFO.SQL
Volume in drive C has no label.
Volume Serial Number is EC56-2FC2

Directory of C:\

03/15/2001 04:47p 15,833 OBJ_INFO.SQL
 1 File(s) 15,833 bytes
 0 Dir(s) 1,342,181,376 bytes free
```

Here is a sample of what the OBJ\_INFO.SQL file looks like. Note that only the index creation information is not commented out.

```
REM CREATE TABLE "SYSTEM"."AUD_SAL" ("C_DATE" DATE, "C_USER"
REM VARCHAR2(30), "EMPNO" NUMBER(10, 0), "BEFORE_SALARY" NUMBER(10, 0),
REM "NEW_SALARY" NUMBER(10, 0)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS
REM 255 LOGGING STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS
REM 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
REM TABLESPACE "CERTDB" ;
REM CREATE TABLE "SYSTEM"."BATCHJOBS" ("JOBID" NUMBER(6, 0) NOT NULL
REM ENABLE, "JOBNAME" VARCHAR2(30) NOT NULL ENABLE, "STATUS" VARCHAR2(30)
```

```

REM NOT NULL ENABLE, "LASTUPDATED" DATE NOT NULL ENABLE) PCTFREE 10
REM PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 40960 NEXT
REM 40960 MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST
REM GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "CERTDB" ;
CONNECT SYSTEM;
CREATE UNIQUE INDEX "SYSTEM"."BATCHJOBS_JOBID_PK" ON "BATCHJOBS" ("JOBID")
PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960
MINEXTENTS 1 MAXEXTENTS 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT) TABLESPACE "CERTDB" LOGGING ;
REM ALTER TABLE "SYSTEM"."BATCHJOBS" ADD CONSTRAINT "BATCHJOBS_JOBID_PK"
REM PRIMARY KEY ("JOBID") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
REM STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505
REM PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
REM TABLESPACE "CERTDB" ENABLE ;
REM CREATE TABLE "SYSTEM"."BONUS" ("ENAME" VARCHAR2(10), "JOB"
REM VARCHAR2(9), "SAL" NUMBER, "COMM" NUMBER) PCTFREE 10 PCTUSED 40
REM INITRANS 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 65536 NEXT 65536
REM MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 50 FREELISTS 1
REM FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "SYSTEM" ;
REM CREATE TABLE "SYSTEM"."CLASSENROLLMENT" ("CLASSID" NUMBER(*,0) NOT
REM NULL ENABLE, "STUDENTNUMBER" NUMBER(*,0) NOT NULL ENABLE, "STATUS"
REM CHAR(10) NOT NULL ENABLE, "ENROLLMENTDATE" DATE NOT NULL ENABLE,
REM "PRICE" NUMBER(9, 2) NOT NULL ENABLE, "GRADE" CHAR(4), "COMMENTS"
REM VARCHAR2(2000)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING
REM STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505
REM PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
REM TABLESPACE "CERTDB" ;
CREATE UNIQUE INDEX "SYSTEM"."PK_CLASSID_STUDENTNUMBER" ON
"CLASSENROLLMENT" ("CLASSID" , "STUDENTNUMBER") PCTFREE 10 INITRANS 2
MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS 505
PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "CERTDB" LOGGING ;
....
REM ALTER TABLE "SYSTEM"."STUDENTS" ADD CONSTRAINT "PK_STUDENTNUMBER"
REM PRIMARY KEY ("STUDENTNUMBER") USING INDEX PCTFREE 10 INITRANS 2
REM MAXTRANS 255 STORAGE(INITIAL 40960 NEXT 40960 MINEXTENTS 1 MAXEXTENTS
REM 505 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
REM TABLESPACE "CERTDB" ENABLE ;
REM ALTER TABLE "SYSTEM"."CLASSENROLLMENT" ADD CONSTRAINT
REM "FK_CLASSENROLLMENT_CLASSID" FOREIGN KEY ("CLASSID") REFERENCES
REM "SCHEDULEDCLASSES" ("CLASSID") ENABLE NOVALIDATE ;
REM ALTER TABLE "SYSTEM"."CLASSENROLLMENT" ADD CONSTRAINT
REM "FK_CLASSENROLLMENT_STUDENTNUM" FOREIGN KEY ("STUDENTNUMBER")
REM REFERENCES "STUDENTS" ("STUDENTNUMBER") ENABLE NOVALIDATE ;
REM ALTER TABLE "SYSTEM"."COURSEAUDIT" ADD CONSTRAINT
REM "FK_COURSEAUDIT_COURSENUMBER" FOREIGN KEY ("COURSENUMBER") REFERENCES
REM "COURSES" ("COURSENUMBER") ENABLE NOVALIDATE ;
REM ALTER TABLE "SYSTEM"."EMP" ADD CONSTRAINT "FK_DEPTNO" FOREIGN KEY
REM ("DEPTNO") REFERENCES "DEPT" ("DEPTNO") ENABLE NOVALIDATE ;
REM ALTER TABLE "SYSTEM"."SCHEDULEDCLASSES" ADD CONSTRAINT
REM "FK_SCHEDCLASS_COURSENUM" FOREIGN KEY ("COURSENUMBER") REFERENCES
REM "COURSES" ("COURSENUMBER") ENABLE NOVALIDATE ;
REM ALTER TABLE "SYSTEM"."SCHEDULEDCLASSES" ADD CONSTRAINT

```

```
REM "FK_SCHEDCLASSES_LOCATIONID" FOREIGN KEY ("LOCATIONID") REFERENCES
REM "LOCATIONS" ("LOCATIONID") ENABLE NOVALIDATE ;
REM ALTER TABLE "SYSTEM"."SCHEDULEDCLASSES" ADD CONSTRAINT
REM "FK_SCHEDCLASSES_INSTID" FOREIGN KEY ("INSTRUCTORID") REFERENCES
REM "INSTRUCTORS" ("INSTRUCTORID") ENABLE NOVALIDATE ;
REM ALTER TABLE "SYSTEM"."BATCHJOBS" ENABLE CONSTRAINT
REM "BATCHJOBS_JOBID_PK" ;
REM ALTER TABLE "SYSTEM"."CLASSENROLLMENT" ENABLE CONSTRAINT
REM "PK_CLASSID_STUDENTNUMBER" ;
REM ALTER TABLE "SYSTEM"."COURSEAUDIT" ENABLE CONSTRAINT "COURSEAUDIT_PK" ;
REM ALTER TABLE "SYSTEM"."COURSES" ENABLE CONSTRAINT "PK_COURSENUMBER" ;
REM ALTER TABLE "SYSTEM"."DEPT" ENABLE CONSTRAINT "PK_DEPT" ;
REM ALTER TABLE "SYSTEM"."EMP" ENABLE CONSTRAINT "PK_EMP" ;
REM ALTER TABLE "SYSTEM"."INSTRUCTORS" ENABLE CONSTRAINT
REM "PK_INSTRUCTORID" ;
REM ALTER TABLE "SYSTEM"."LOCATIONS" ENABLE CONSTRAINT "PK_LOCATIONID" ;
REM ALTER TABLE "SYSTEM"."SCHEDULEDCLASSES" ENABLE CONSTRAINT
REM "PK_CLASSID" ;
REM ALTER TABLE "SYSTEM"."STUDENTS" ENABLE CONSTRAINT "PK_STUDENTNUMBER" ;
REM ALTER TABLE "SYSTEM"."CLASSENROLLMENT" ENABLE CONSTRAINT
REM "FK_CLASSENROLLMENT_CLASSID" ;
REM ALTER TABLE "SYSTEM"."CLASSENROLLMENT" ENABLE CONSTRAINT
REM "FK_CLASSENROLLMENT_STUDENTNUM" ;
REM ALTER TABLE "SYSTEM"."COURSEAUDIT" ENABLE CONSTRAINT
REM "FK_COURSEAUDIT_COURSENUMBER" ;
REM ALTER TABLE "SYSTEM"."EMP" ENABLE CONSTRAINT "FK_DEPTNO" ;
REM ALTER TABLE "SYSTEM"."SCHEDULEDCLASSES" ENABLE CONSTRAINT
REM "FK_SCHEDCLASS_COURSENUM" ;
REM ALTER TABLE "SYSTEM"."SCHEDULEDCLASSES" ENABLE CONSTRAINT
REM "FK_SCHEDCLASSES_LOCATIONID" ;
REM ALTER TABLE "SYSTEM"."SCHEDULEDCLASSES" ENABLE CONSTRAINT
REM "FK_SCHEDCLASSES_INSTID" ;
```

## Importing using Oracle Enterprise Manager

To use the Import utility through Oracle Enterprise Manager, you need to perform the following steps. These are the steps to follow when performing a FULL import.

### STEP BY STEP: Importing Using Oracle Enterprise Manager

1. Launch the Oracle Enterprise Manager Console.
2. Enter the administrator username and password for the Management Server.
3. Expand the Database folder.
4. Right-click the database to run the export against. Be sure to set the preferred credentials for the Node before starting the Import.
5. Specify the type of import as in Figure 15-6.

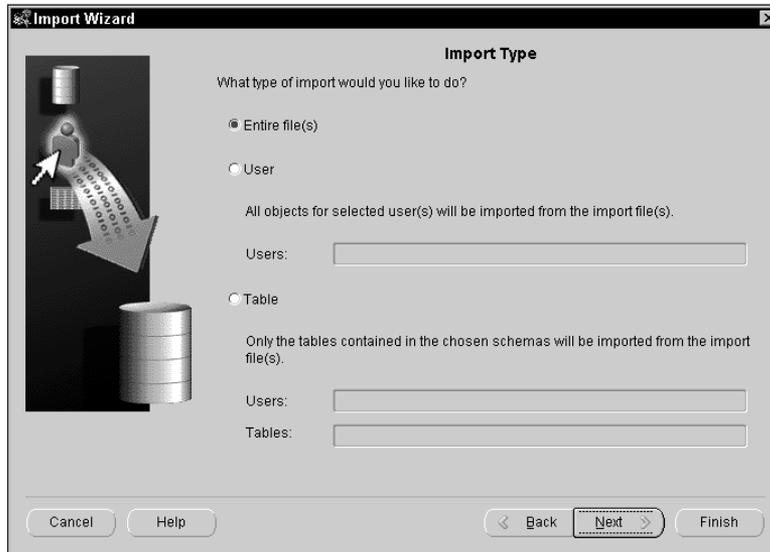


Figure 15-6: Select import type

6. Enter the file for the import file.
7. Specify the associated objects, depending on the mode selected in Step 5 as in Figure 15-7.

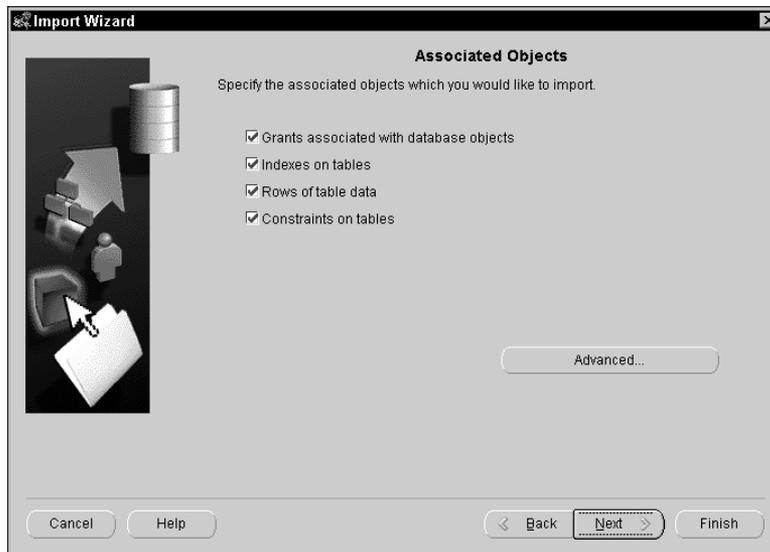


Figure 15-7: Specify associated objects

8. Specify schedule parameters as in Figure 15-8.



Figure 15-8: Schedule job

## Import behavior

When importing objects, the user performing the import must have the privilege to execute the command. If a table is being created, the user performing the import must have the CREATE TABLE privilege as well as a QUOTA parameter in the tablespace where the table is being created. The same holds true for all objects: the user must have the required privilege.

The user must also have, at a minimum, the CREATE SESSION privilege to run either the Import or Export utility.

## Order of import

Objects of a table are imported in the same order as they appear in the export file. The order of objects is as follows:

1. Type definitions
2. Table definitions
3. Table data

#### 4. Table indexes

#### 5. Integrity constraints, views, procedures, and triggers

#### 6. Bitmap, functional, and domain indexes

The sequence is important; it is intended to prevent objects from being rejected during import. In the following example, two tables are in the export file: EMP and DEPT. There is a FOREIGN KEY constraint in the EMP table on the DEPTNO column referencing the DEPTNO column of the DEPT table. For the import to succeed, the FOREIGN KEY constraint cannot be enabled until the DEPT table is first created, the data is loaded into the DEPT table, and the PRIMARY KEY constraint on the DEPT table is enabled. By using the SHOW parameter of the import, you can see the order with which import will be performed.

```
C:\>IMP full=y show=y
```

```
Import: Release 8.1.6.0.0 - Production on Thu Mar 15 23:14:52 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Username: student
```

```
Password:
```

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
```

```
With the Partitioning option
```

```
JServer Release 8.1.6.0.0 - Production
```

```
Export file created by EXPORT:V08.01.06 via conventional path
import done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
. importing STUDENT's objects into STUDENT
"CREATE TABLE "EMP" ("EMPNO" NUMBER(4, 0), "ENAME" VARCHAR2(10), "JOB" VARCH"
"AR2(9), "MGR" NUMBER(4, 0), "HIREDATE" DATE, "SAL" NUMBER(7, 2), "COMM" NUM"
"BER(7, 2), "DEPTNO" NUMBER(2, 0)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRAN"
"S 255 LOGGING STORAGE(INITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 2147"
"483645 PCTINCREASE 50 FREELISTS 1 FREELIST ,GROUPS 1 BUFFER_POOL DEFAULT) TA"
"BLESPEACE "SYSTEM""
. . skipping table "EMP"

"CREATE UNIQUE INDEX "PK_EMP" ON "EMP" ("EMPNO") PCTFREE 10 INITRANS 2 MAX"
"TRANS 255 STORAGE(INITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 21474836"
"45 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLES"
"PACE "SYSTEM" LOGGING"
"CREATE TABLE "DEPT" ("DEPTNO" NUMBER(2, 0), "DNAME" VARCHAR2(14), "LOC" VAR"
"CHAR2(13)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING STORAGE(I"
"NITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 50 F"
"REELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "SYSTEM""
. . skipping table "DEPT"

"CREATE UNIQUE INDEX "PK_DEPT" ON "DEPT" ("DEPTNO") PCTFREE 10 INITRANS 2 "
"MAXTRANS 255 STORAGE(INITIAL 65536 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 21474"
"83645 PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TAB"
```

```
"LESPACE "SYSTEM" LOGGING"
"ALTER TABLE "EMP" ENABLE CONSTRAINT "PK_EMP""
"ALTER TABLE "DEPT" ENABLE CONSTRAINT "PK_DEPT""
"ALTER TABLE "EMP" ENABLE CONSTRAINT "FK_DEPTNO""
Import terminated successfully without warnings.
```

From the import, you can see that the FOREIGN KEY constraint on the EMP table is not enabled until the PRIMARY KEY on the DEPT table is enabled. This will prevent the constraint from being invalidated.

The process is not entirely perfect and objects such as procedures may be created before the objects they reference are created. You generally can rectify this situation by rerunning the import and setting the IGNORE parameter to Y or by querying the data dictionary view DBA\_OBJECTS and finding all the objects where STATUS=INVALID. These objects can simply be recompiled or rebuilt.

```
SELECT OBJECT_NAME ,
 OWNER,
 OBJECT_TYPE ,
 STATUS
FROM DBA_OBJECTS
WHERE STATUS='INVALID';
```

### Considerations for importing into existing tables

When loading data into an existing table, you should disable all FOREIGN KEY constraints on that table to eliminate rejections. This is most common when the referential integrity constraint on a table references itself. An example of this situation is an employee table with a manager column that references employee numbers in the same table. After the data has been loaded, simply enable the FOREIGN KEY constraint.

The user performing the import must also have INSERT privileges on the table or have the INSERT ANY system privilege.

### Placement of objects

One of the main reasons for performing exports and imports is to relocate data. The tablespace information is specified during the export; therefore, if the tablespace exists and the user has a QUOTA on that tablespace, the table will be created in the same tablespace. If the tablespace does not exist, the import process will create it in the default tablespace of the user performing the import. If the user does not have sufficient quota in this tablespace, the creation will fail.

Relocating data this way, as you can imagine, is very frustrating. What happens when you have several tables that you wish to relocate to several different tablespaces? Having multiple default tablespaces is not possible. So, your two options are to perform several different imports — changing the default tablespace to the desired tablespace before performing the import, which can be very time consuming; or to use the INDEXFILE parameter of the import and create an ASCII

file with all the table and index creation syntax. Simply modify this file, correctly locate the objects on the appropriate tablespace, and then run this script. Doing so will pre-create the objects in the correct tablespaces before the data is loaded. The IGNORE parameter must be set to Y when the actual import is performed.

## Export and import guidelines

The following is a list of things to remember when performing exports and imports:

- ♦ Learn the Command Line mode for both exports and imports. This gives you the full use of all parameters.
- ♦ Use parameter files to simplify exports and imports. You can run imports and exports simply by using the PARFILE parameter.  

```
EXP PARFILE=EXPPAR.PAR
```
- ♦ Avoid setting CONSISTENT=Y on exports. This can lead to “snapshot too old” errors. If you need a consistent image of the database, consider enabling RESTRICTED SESSION to prevent users from connecting to the database.
- ♦ Do not use the COMPRESS=Y option if the table being exported is subject to many deletes. The COMPRESS option attempts to create a single extent large enough to hold all extents for an object. If you have a large number of DELETES or many empty blocks, using the COMPRESS option can lead to wasted space.
- ♦ Ensure that sufficient memory is available.
- ♦ Ensure that the export file does not reside on the same physical disk as the datafiles for the database. The reading from the export file and writing to the datafiles will cause conflicts if they are on the same disk.
- ♦ Use Direct Path exports when available.
- ♦ Try to create indexes after the data has been loaded. Doing so lets you tune and optimize index creation, which can make the index creation much quicker.

## NLS considerations for imports and exports

Proper setting of the NLS parameters can improve the efficiency of imports and exports. Here are some things to consider.

### Export and character set conversion

One of the disadvantages of using Direct Path exports is that the character set of the export session must be that of the database. This means that if a client is running a different character set than the database, the export will fail. To fix the problem, set the appropriate NLS\_LANG parameter on the client. When Conventional Path exports are performed, the export is done using the character set of the client. The export process will convert from the database character set to the client character set.



Chapter 20 covers the NLS parameters.

### Import and character set conversion

As with the export, the ideal scenario is for the import session to have the same character set as the target database. Imports have another wrinkle—the character set of the source database. The potential exists for two character set conversions, the first from the export file to the client session and the second from the client session to the target database. The two conversions will adversely affect the performance of the import.

Try to avoid character set conversions. They are not only slower but also can lead to unknown characters in the database. If a character exists in the export file that has no equivalent character in the converted character set, the default “unknown character” is used. To ensure that all characters are correctly converted, the target database character set must be a superset of the converting character set.

Ideally, the character set of the source database, the import session, and the target database should all be the same, eliminating the need for conversions, which will speed up the import process.

## Transportable Tablespaces

The movement of data between databases is often a full-time job in certain organizations. When data is moved from an OLTP database to a DSS database, the task is generally very time consuming and difficult. Most times, STAGING databases are used. A STAGING database is a transitional database between an OLTP and DSS database. The transition is required because the table structure and layouts are often different. Also, data often is moved from DSS database to data marts. With both OLTP to DSS and DSS to data marts, the DBA can use either Direct Path loading, through SQL\*Loader, or parallel DML. Both types of transfer require a high level of skill and understanding. Transportable tablespaces have simplified this task and dramatically sped it up.

Using transportable tablespaces allows for datafiles to be transferred between identical databases without the need to export and import the data. All that gets exported and imported is the data dictionary or metadata information for each tablespace. The time it now takes to transfer data is the time it takes to transfer files between servers. This feature is very beneficial for companies that publish data to distributors, salespeople, and customers. Datafiles can be placed on a CD and then integrated into existing databases very quickly and simply. A company that upgrades its product lists and pricing can simply publish a datafile from a tablespace. Customers can simply integrate this datafile into their existing databases and be done with it. Another practical application for this technology is

for census data. Companies can easily integrate large amounts of demographic information by integrating datafiles into their existing databases.

## Implementing Transportable Tablespaces

This section covers the steps for using transportable tablespaces. A subsequent section discusses the guidelines for their use.

### STEP BY STEP: Transporting Tablespaces

1. Make tablespace(s) being transported read-only.

Because only datafiles are being transferred, the level of focus moves from a database down to the tablespace level. Eventually, a datafile will be copied between databases; therefore, it is critical for that datafile to be in a quiescent state. Because metadata is also going to be exported, the database needs to be open, which means that it cannot be shut down. So, the tablespace or tablespaces that are going to be transferred should be made read-only, guaranteeing that the datafiles belonging to those tablespaces will not be changing.

```
ALTER TABLESPACE TOOLS READ ONLY;
```

2. Export the metadata or data dictionary information from the source database for the tablespaces being transported.

The tablespaces being transported do not need to have any special designation. Nothing needs to be done to make a tablespace a “transportable tablespace.” After the tablespace(s) has been made read-only, use the Export utility to export the metadata for the tablespaces(s) being transported. This won’t take very long and the export file will be very small relative to the datafile. Two parameters of the export should be specified: `TRANSPORT_TABLESPACE`, which specifies that the export’s purpose is to just export metadata; and `TABLESPACES`, which is a comma-delimited list of tablespaces being transported. If just data is being transferred between databases, you should set the `TRIGGERS`, `CONSTRAINTS`, and `GRANTS` export parameter to `N` so that the associated triggers, constraints, and grants are not be exported. Here is an example of the export parameter file for performing the transportable tablespace export as well as the export itself:

```
C:\>type trans_exp.par
USERID="sys/change_on_install as sysdba"
FILE=TOOLS_TRANS.DMP
TRANSPORT_TABLESPACE=Y
TABLESPACES=TOOLS
TRIGGERS=N
CONSTRAINTS=N
GRANTS=N
```

```

C:\>EXP PARFILE=TRANS_EXP.PAR

Export: Release 8.1.6.0.0 - Production on Sat Mar 17 15:48:50
2001

(c) Copyright 1999 Oracle Corporation. All rights reserved.

Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 -
Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
Export done in WE8ISO8859P1 character set and WE8ISO8859P1
NCHAR character set
Note: table data (rows) will not be exported
Note: grants on tables/views/sequences/roles will not be
exported
Note: constraints on tables will not be exported
About to export transportable tablespace metadata...
For tablespace TOOLS ...
. exporting cluster definitions
. exporting table definitions
. . exporting table LOCATIONS
. . exporting table LOCATIONS
. end transportable tablespace metadata export
Export terminated successfully without warnings.

```

3. Copy the datafiles and the export file belonging to the tablespace(s) to the target database. Put the datafiles in the desired location on the target database. The directory structure and filename do not have to match between the two databases.

The data dictionary view `DBA_DATA_FILES` can be queried to get a list of datafiles belonging to each tablespace.

```

SVRMGR> SELECT FILE_NAME
 2> FROM DBA_DATA_FILES
 3> WHERE TABLESPACE_NAME='TOOLS';

FILE_NAME

C:\CERTDB\DISK1\TOOLS01.DBF

```

4. Make the tablespace(s) in the source database read-write again.

```
ALTER TABLESPACE TOOLS READ WRITE
```

5. Import the metadata into the target database.

After the files have been copied to the target database, the export file needs to be imported. As with the export, two parameters need to be specified. Set the `TRANSPORT_TABLESPACE=Y` parameter; also, set the `DATAFILES` parameter to

specify the location of the datafiles on the target database. Specify the names as they appear to the OS even if they are different from the source database. Enough information is in the header of the datafile for the import process to associate it with a tablespace. The FROMUSER and TOUSER parameters can be used to transfer ownership of objects between schemas from the source to the target databases. If the FROMUSER and TOUSER parameters are not included, the objects are imported under the same schema as that in the source database. If these users do not exist, the import returns an error.

```
C:\>TYPE TRANS_IMP.PAR
USERID="sys/change_on_install as sysdba"
FILE=TOOLS_TRANS.DMP
TRANSPORT_TABLESPACE=Y
DATAFILES=(c:\CERTDB\DISK1\TOOLS01.DBF)
```

```
C:\>IMP PARFILE=TRANS_IMP.PAR
```

```
Import: Release 8.1.6.0.0 - Production on Sat Mar 17 16:14:00
2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 -
Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
```

```
Export file created by EXPORT:V08.01.06 via conventional path
About to import transportable tablespace(s) metadata...
import done in WE8ISO8859P1 character set and WE8ISO8859P1
NCHAR character set
. importing SYS's objects into SYS
. importing SCOTT's objects into SCOTT
. . importing table "LOCATIONS"
. importing STUDENT's objects into STUDENT
. . importing table "LOCATIONS"
Import terminated successfully without warnings.
```

## 6. Alter the tablespace to be read-write

```
ALTER TABLESPACE TOOLS READ WRITE
```

## Transportable Tablespace uses and guidelines

Following are a few guidelines and reasons for using transportable tablespaces.

- ♦ Doing so moves the entire tablespace and tablespace data.
- ♦ Media recovery is supported and a backup should be performed after the transport to ensure full recoverability.

- ♦ If constraints are to be maintained, consider relocating tables to the same tablespace or tablespaces so that they can more easily be transported.
- ♦ The source and target database must share
  - The same OS
  - The same version of Oracle8i or higher
  - The same block size
  - The same character set and national language character set
- ♦ Tablespaces being transported must be self-contained.
  - All partitions of a partitioned table must be on the exported tablespaces.
  - LOBs must be exported with the tables
  - All indexes belonging to the tables being transported must be exported as well. Therefore, if a table has an index in another tablespace, that tablespace must be transported as well.
- ♦ The following objects cannot be exported as part of a transportable tablespace export:
  - Tables containing nested tables or VARRAYs
  - Bitmap indexes

## Checking for self-contained tablespaces

Before performing a transportable tablespace export, the objects (tables) in the tablespace(s) being exported must be self-contained. That is, all partitions of a partitioned table and all LOBs of a table must be included in the list of tablespaces. If constraints are to be maintained, all objects referenced by FOREIGN KEY constraints must be exported as well. If a table has indexes, the tablespace containing those indexes must be transported as well to make the tablespaces self-contained. If the TOOLS tablespace contains a table called EMPLOYEE that has a FOREIGN KEY pointing to the DEPARTMENTS table in the DATA01 tablespace, the DATA01 tablespace should also be transported.

Oracle has a supplied procedure called `DBMS_TTS.TRANSPORT_SET_CHECK`, which verifies that tablespaces are self-contained. The procedure accepts two arguments: a comma-separated list of tablespaces and a Boolean expression indicating whether constraints should be checked as well. The procedure reports all exceptions to a data dictionary view called `TRANSPORT_SET_VIOLATIONS`.

In this example, a FOREIGN KEY constraint on the SCHEDULEDCLASSES table references a PRIMARY KEY constraint in the LOCATIONS table. The SCHEDULEDCLASSES table is on the CERTDB tablespace and the LOCATIONS table is on the TOOLS tablespace.

```
SQL> EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK(ts_list => 'CERTDB',
 incl_constraints => TRUE)

SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;

VIOLATIONS

Constraint SYS_C001166 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SYS_C001167 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SYS_C001168 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SYS_C001169 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SYS_C001170 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SYS_C001171 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SYS_C001172 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SYS_C001173 between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint PK_CLASSID between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint FK_SCHEDCLASS_COURSENUM between table STUDENT.LOCATIONS in tablespace
TOOLS and table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint FK_SCHEDCLASSES_INSTID between table STUDENT.LOCATIONS in tablespace
TOOLS and table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Constraint SCHED_LOCID between table STUDENT.LOCATIONS in tablespace TOOLS and
table STUDENT.SCHEDULEDCLASSES in tablespace CERTDB

Index STUDENT.SYS_C001273 in tablespace CERTDB enforces primary constraints of
table STUDENT.LOCATIONS in tablespace TOOLS

13 rows selected.
```

**It is clear from this that the CERTDB and TOOLS tablespace are not self contained and would need to be transported at the same time if constraints are to be maintained.**

Now, in the following example, the procedure is run specifying both the CERTDB and TOOLS tablespaces. Notice that there are no exceptions in the TRANSPORT\_SET\_VIOLATIONS data dictionary view.

```
SQL EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK(ts_list=>'CERTDB,TOOLS',
 incl_constraints => TRUE);

SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;

no rows selected
```

There is also a function called DBMS\_TTS.ISSELFCONTAINED, which returns a TRUE if the tablespace(s) specified is self-contained. The function accepts the same arguments as the procedure.

## Key Point Summary

In preparing for the Oracle8i DBA: Architecture and Administration exam, please keep these points in mind regarding reorganizing data:

This chapter looked at the circumstances of when and why data needs to be moved as well as the four main options for performing the move. The circumstances for reorganizing data are:

- ♦ Data may need to be moved to reorganize a fragmented table.
- ♦ Relocating objects onto different tablespaces for better distribution of IO.
- ♦ To change ownership of objects.
- ♦ To move data between databases.

The methods for performing the move all have their benefits, and the decision as to which method to use depends on the desired result. The four methods discussed in this chapter were:

- ♦ The CREATE TABLE as SELECT statement.
- ♦ The ALTER TABLE MOVE command.
- ♦ The Import and Export utilities.
- ♦ Using transportable tablespaces.

There are three different interfaces that can be used for running the Import and Export utilities:

- ♦ Command Line Interface
- ♦ Interactive mode
- ♦ Oracle Enterprise Manager

The Import and Export Utilities support four different modes for running. The mode that is chosen depends upon the desired outcome of the import or export. The modes are:

- ♦ Table Mode
- ♦ User Mode
- ♦ Tablespace Mode
- ♦ Database Mode



## STUDY GUIDE

---

Now that you have learned about relocating data and the Oracle tools available for performing the move, you should test your understanding by reviewing the assessment questions and performing the exercises that follow.

### Assessment Questions

1. The Export utility reads data from the database and writes it to \_\_\_\_\_.
  - A. A data dictionary table
  - B. A special user table for storing exported data
  - C. An ASCII file
  - D. A binary file
  - E. The database buffer cache
2. Why is a Direct Path export faster than a Conventional Path export?
  - A. Because data is written directly from the datafiles to the export file, bypassing the database buffer cache
  - B. Because the SQL COMMAND processing layer is bypassed
  - C. Because it can be run in parallel
  - D. Because the evaluation buffer is bypassed when data is written to the export file
  - E. Both B and D
3. Imports in Oracle can read files generated only by which one of the following?
  - A. Exports performed on the same OS as the database performing the import
  - B. Exports performed on the same version of Oracle as the database performing the import
  - C. Both A and B
  - D. Any valid Oracle export
  - E. Any binary or ASCII file

4. What role is required to perform a FULL database export?
  - A. EXP\_ANY\_DATABASE
  - B. CONNECT
  - C. RESOURCE
  - D. EXP\_DATABASE
  - E. EXP\_FULL\_DATABASE
5. What role is required to perform a FULL database import?
  - A. IMP\_FULL\_DATABASE
  - B. IMP\_ANY\_DATABASE
  - C. CONNECT
  - D. RESOURCE
  - E. IMPORT\_DATABASE
6. When a User mode export is being performed by a nonprivileged user, what objects are going to be exported?
  - A. All objects referenced by the TABLES parameter
  - B. All objects owned by the user specified by the FROMUSER parameter
  - C. Only objects that the user performing the export owns
  - D. Only table definitions and no data will be exported
  - E. Objects owned by the users referenced in the OBJECT\_OWNER parameter
7. What does the INDEXES parameter tell the import to do?
  - A. Whether indexes should be imported
  - B. Provide a filename for the import to store the index creation syntax in but does not create the indexes
  - C. Provide a filename for the import to store the index, table, and constraint syntax in but does not process the import
  - D. Validate the structure of the index during the import
  - E. Rebuild indexes during the import

8. The FROMUSER and TOUSER parameters of the Export utility can be used for what?
  - A. Change the name of tables being exported
  - B. Change the ownership of the database from one user to another in the export file
  - C. Change permissions on objects from one user to another in the export file
  - D. Change ownership of objects from one user to another during the export
  - E. None of the above; the FROMUSER and TOUSER serve as parameters only of the IMPORT
9. Transportable tablespaces can be used for what?
  - A. Moving tables from one tablespace to another
  - B. Changing ownership of objects from one user to another
  - C. Moving rows from one table to another
  - D. Moving tablespaces from one physical disk to another on the same database
  - E. Moving tablespaces between databases
10. What information is exported during a transportable tablespace export?
  - A. All rows in the tablespace being transported
  - B. All rows and table definitions from the tablespace being transported
  - C. The metadata for the entire database
  - D. Only the metadata for the tablespaces referenced in the TRANSPORT\_TABLESPACE parameter of the export
  - E. Only the metadata for the tablespaces referenced in the TABLESPACES parameter of the export
11. What must happen before a transportable tablespace export can occur?
  - A. The tablespaces being transported must be made read-only
  - B. The tablespaces being transported must be made read-write
  - C. The database must be made read-only
  - D. The database must be shut down to at least the mount state
  - E. A full database export

## Scenarios

1. You are a DBA and the company you are working for wants to create a test database for testing the General Ledger application. Several modifications are being made to the software and the company wants to test them prior to implementation. The database is 500GB in size; however, the General Ledger portion is only 5GB. All the objects for the General Ledger application are owned by one user called GL\_OWNER. The test server has only 20GB of storage, so re-creating the entire database would be impossible. You have also been informed that this test database will need to be re-created several times after the tests are run to get the database to a consistent state.
  - A. Based on this information, how would you go about creating this test database?
  - B. What considerations should you make for re-creating the database after test runs have completed?
2. You have been hired as a consultant by JOBS JOBS JOBS Ltd to help set up a data warehouse. Your job is to populate the data in the staging area. This is not the actual database for the data warehouse. The data from the staging area will be transferred to the data warehouse as part of a separate process. The structure of the database in the data warehouse will be different from the production database; therefore, indexes and constraints won't be an issue and don't need to be part of the staging database. The production database is 20GB in size and contains 24 months of data.
  - A. What would be the most efficient way to populate the staging database?
  - B. Based on your recommendations, outline the steps involved in creating the staging database. Be sure to include detailed recommendations, such as the import and export parameters used.

## Lab Exercises

### Lab 15-1 Performing User-Based Exports and Imports

1. Connect to your instance using Server Manager Line mode (svrmgrl) as the user STUDENT with a password of ORACLE.
2. Determine the objects owned by the user STUDENT.
3. Perform an export for all objects owned by the user STUDENT. Name the export file student\_exp.dmp and make sure that it is located in the C:\CERTDB directory. Make sure that constraints and grants are not exported, because this will cause problems with a future exercise.

4. Create a new user called TODD using the following syntax. You will need to connect to the database using Server Manager Line mode (svrmgrl) as a user with sysdba privileges.

```
CREATE USER TODD IDENTIFIED BY ROSS
 DEFAULT TABLESPACE TOOLS QUOTA UNLIMITED ON TOOLS;

GRANT CONNECT TO TODD;
```

5. Use the Import utility to view the contents of the export file student\_exp.dmp. You should use a privileged user account such as SYS or SYSTEM.
6. Use the Import utility to import the export file created in Question 3 into the user TODD's schema. You will need to connect as a privileged user and use the FROMUSER and TOUSER parameters of the import.
7. Connect to Server Manager Line mode (svrmgrl) as user TODD with a password of ROSS and verify that this user now owns objects.

## Lab 15-2 Transportable Tablespaces

1. Connect to your instance using Server Manager Line mode (svrmgrl) as a user with SYSDBA privileges.
2. Create a new tablespace called TRANSPORT with a datafile called transport.dbf that is 1MB in size located on DISK5.
3. Connect as the user STUDENT with the password ORACLE.
4. Create a table called TRANS\_TEST with one column called COL1 that is a number(10) on tablespace TRANSPORT.
5. Insert a row into the TRANS\_TEST table with a value of 1234567890 for COL1.
6. Connect as a user with SYSDBA privileges.
7. The TRANSPORT tablespace will be transported as part of transportable tablespace export. Make the tablespace read-only.
8. Verify that the TRANSPORT tablespace is self-contained.
9. Perform a transportable tablespace export of the TRANSPORT tablespace. Name the export file TRANSPORT.DMP and locate the file in the C:\CERTDB directory.
10. Back up the datafile for the TRANS\_TEST tablespace to DISK6. Call the backup transport.dbf.
11. To simulate a database that does not have the TRANSPORT tablespace, drop the TRANSPORT tablespace. You will need to include the INCLUDING CONTENTS option of the DROP TABLESPACE command.
12. Because the DROP TABLESPACE command does not remove the file at the OS, you will need to do it. Delete the file on DISK5 called transport.dbf.

13. Perform a transportable tablespace import. The datafile for the tablespace called TRANSPORT exists on DISK6 called transport.dbf, and the export file should be called TRANSPORT.DMP located in the C:\CERTDB directory.
14. Connect to your instance using Server Manager Line mode (svrmgrl) as a user with SYSDBA privileges and make the TRANSPORT tablespace read-write.
15. Verify that the TRANSPORT tablespace exists and that the TRANS\_TEST exists.

## Answers to Chapter Questions

### Chapter Pre-Test

1. Table, User, Tablespace, and Database modes.
2. Objects owned by the user SYS, and also ORDSYS, CTXSYS, MDSYS, ORDPLUGINS, are not exported during a full database export. Because the user SYS owns the data dictionary objects and these objects are recreated when the database is re-created, they do not need to be exported. This is why it is critical that the user SYS does not own objects in the database except for the data dictionary objects.
3. The Conventional Path export uses SQL SELECT statements to extract data from tables. Rows are transferred to an evaluation buffer and then written to the export file. The Direct Path mode reads data directly, skipping the SQL command-processing layer. The Direct Path mode bypasses the database buffer cache and performs writes directly to disk. This is a much quicker action.
4. Only Conventional Path exports are supported.
5. Use the FROMUSER(username[|,username]) and TOUSER(username) options. The FROMUSER consists of a list of users whose objects you want to change the ownership of; the TOUSER becomes the owner of all objects from the export file owned by the users in the FROMUSER list.
6. Use the COMMIT and BUFFER options. If both are used, a commit will occur after array insert performed is specified by the BUFFER parameter.
7. Transportable tablespaces provide a very fast and efficient mechanism for transferring large volumes of data between databases. This transfer can be done without the expense of exporting and importing all the data. All that is required is an export and import of the metadata versus all the rows.
8. The main limitation of transportable tablespaces is that the tablespaces(s) being exported must be self-contained. The databases must also be on the same operating system, use the same version of release 8.1 or higher, have the same block size, and use the same character set.

## Assessment Questions

1. **D.** Exports write data to a binary file. Refer to the section “The Import and Export utilities” for more information.
2. **E.** A Direct Path export is faster than a Conventional Path export because the SQL COMMAND processing layer and the evaluation buffer are both bypassed. For more information, refer to the section “Export types.”
3. **D.** An Oracle import can be performed using any valid Oracle export. It does not necessarily have to be from the same OS or same version of Oracle. For more information, refer to the section “The Import and Export utilities.”
4. **E.** EXP\_FULL\_DATABASE is the role required. This role is assigned to the DBA role. Refer to the section “Export modes” for more information.
5. **A.** IMP\_FULL\_DATABASE is the role required. This role is assigned to the DBA role. Refer to section “Export modes” for more information.
6. **C.** Only objects owned by the user performing the export will be exported. Refer to the section “Export modes” for more information.
7. **A.** The INDEXES parameter tells the import whether indexes should be imported. More information on the import parameters can be found in Table 15-3.
8. **E.** The FROMUSER and TOUSER parameters are ones of the Import utility only.
9. **E.** Transportable tablespaces provide a very efficient means of moving tablespaces between databases. More information on transportable tablespaces can be found in section “Transportable Tablespaces.”
10. **E.** Only the metadata for the tablespaces referenced by the TABLESPACES parameter will be exported. No data or rows are exported, only metadata. Refer to the section “Implementing Transportable Tablespaces” for more information.
11. **A.** The tablespaces referenced in a transportable tablespace export must be made read-only before you perform the export.

## Scenarios

1. **A.** Based on the information provided, the easiest way to create the test database would be to do either a TABLE- or USER-based export and import. Because the database is so large, exporting and importing the entire database would not be efficient, especially if it will be re-created several times after each test run.

The first thing that needs to happen is an export of the data in the production system. Picking a time when very little activity is happening on the database would be good. The fact that all the objects that need to be exported are owned by the user GL\_OWNER simplifies the export. After the export has completed, copy the export file to the test server.

On the test server, you first need to create a database. The database must exist before the data can be imported. After the database has been created, create a user called `GL_OWNER` and a tablespace or tablespaces for storing the objects from the export file. Then, simply perform the import.

**B.** To re-create the database after a test run has completed, drop and re-create the database and then import the file again. Because the export file is small relative to the production database, this should be the fastest method for recreating the test database. Also, because the test runs are going to be rerun on the same data, using transportable tablespaces will not be possible because the data would have changed.

2. **A.** The most efficient way to populate the staging area will be to use transportable tablespaces.  
**B.** The steps involved in creating the staging database is to first create the database and then import or plug in the transportable tablespaces. The database should need to be created only once. After this step has been completed, you can do updates to the staging database by plugging in the transportable tablespaces.
3. Create the staging database.
4. Pick the tablespaces that need to be transported. Use the `DBMS_TTS.TRANSPORT_SET_CHECK` procedure to make sure that these tablespaces are self contained. Because constraints do not need to be part of the staging database, the check can focus on partitioned tables and tables with LOBs.
5. Make the tablespaces that are going to be transported read-only.
6. Export the metadata for those tablespaces. Here is an example of an export parfile assuming that the `DATA01` and `DATA02` tablespaces are going to be transported.

```
USERID="sys/change_on_install as sysdba"
FILE=trans_exp.dmp
TRANSPORT_TABLESPACES=Y
TABLESPACES='DATA01,DATA02'
CONSTRAINTS=N
GRANTS=N
TRIGGERS=N
```

7. Copy the `trans_exp.dmp` file and datafiles for the `DATA01` and `DATA02` tablespaces to the server where the staging database was created in Step 1.
8. Make the `DATA01` and `DATA02` tablespaces read-write in the production database.
9. Import the transportable tablespaces in the staging database.
10. Make the `DATA01` and `DATA02` tablespaces in the staging database read-write.

## Lab Exercises

### Lab 15-1

2.

```
SQL> SELECT OBJECT_NAME,OBJECT_TYPE
 2 FROM USER_OBJECTS;
```

3.

```
C:\CERTDB>EXP USERID=STUDENT OWNER=STUDENT COMPRESS=N FILE=C:\CERTDB\STUDENT_EXP
.DMP CONSTRAINTS=N GRANTS=N
```

Export: Release 8.1.6.0.0 - Production on Sun Mar 18 20:41:43 2001

(c) Copyright 1999 Oracle Corporation. All rights reserved.

Password:

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
Note: grants on tables/views/sequences/roles will not be exported
Note: constraints on tables will not be exported
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user STUDENT
. exporting object type definitions for user STUDENT
About to export STUDENT's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export STUDENT's tables via Conventional Path ...
. . exporting table BATCHJOBS 3 rows exported
. . exporting table CLASSENROLLMENT 7 rows exported
. . exporting table COURSEAUDIT 0 rows exported
. . exporting table COURSES 9 rows exported
. . exporting table INSTRUCTORS 8 rows exported
. . exporting table LOCATIONS 3 rows exported
. . exporting table SCHEDULEDCLASSES 3 rows exported
. . exporting table STUDENTS 11 rows exported
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting snapshots
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
```

```
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully without warnings.
```

**5.**

```
C:\CERTDB>IMP USERID=SYS/CHANGE_ON_INSTALL
FILE=C:\CERTDB\STUDENT_EXP.DMP SHOW=Y FULL=Y
```

**6.**

```
C:\CERTDB>IMP USERID=SYS/CHANGE_ON_INSTALL FILE=STUDENT_EXP.DMP FROMUSER=STUDENT
TOUSER=TODD
```

Import: Release 8.1.6.0.0 - Production on Sun Mar 18 20:38:28 2001

(c) Copyright 1999 Oracle Corporation. All rights reserved.

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
```

Export file created by EXPORT:V08.01.06 via conventional path

Warning: the objects were exported by STUDENT, not by you

```
import done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
. importing STUDENT's objects into TODD
. . importing table "BATCHJOBS" 3 rows imported
. . importing table "CLASSENROLLMENT" 7 rows imported
. . importing table "COURSEAUDIT" 0 rows imported
. . importing table "COURSES" 9 rows imported
. . importing table "INSTRUCTORS" 8 rows imported
. . importing table "LOCATIONS" 3 rows imported
. . importing table "SCHEDULEDCLASSES" 3 rows imported
. . importing table "STUDENTS" 11 rows imported
Import terminated successfully without warnings.
```

```
C:\CERTDB>EXP USERID=STUDENT OWNER=STUDENT COMPRESS=N FILE=C:\CERTDB\STUDENT_EXP
.DMP CONSTRAINTS=N GRANTS=N
```

**7.**

```
SQL> SELECT OBJECT_NAME,OBJECT_TYPE
2 FROM USER_OBJECTS;
```

**Lab 15-2****2.**

```
SVRMGR> CREATE TABLESPACE TRANSPORT DATAFILE
2 'C:\CERTDB\DISK5\TRANSPORT.DBF' SIZE 1m;
```

3.

```
CONNECT STUDENT/ORACLE
```

4.

```
SVRMGR> CREATE TABLE TRANS_TEST (COL1 NUMBER(10)) TABLESPACE
TRANSPORT;
```

5.

```
INSERT INTO TRANS_TEST VALUES (1234567890);
```

6.

```
CONNECT INTERNAL
```

7.

```
ALTER TABLESPACE TRANSPORT READ ONLY
```

8.

```
SVRMGR> EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK(ts_list =>
'TRANSPORT', incl_constraints=>TRUE);
```

```
Statement processed.
```

```
SVRMGR> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
VIOLATIONS
```

```

```

```
0. rows selected.
```

9.

```
C:\CERTDB>EXP USERID='sys/change_on_install as sysdba'
FILE=C:\CERTDB\TRANSPORT.DMP TRANSPORT_TABLESPACE=Y TABLESPACES=TRANSPORT
```

```
Export: Release 8.1.6.0.0 - Production on Sun Mar 18 22:09:14 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
Note: table data (rows) will not be exported
About to export transportable tablespace metadata...
For tablespace TRANSPORT ...
. exporting cluster definitions
. exporting table definitions
. . exporting table TRANS_TEST
```

```
. exporting referential integrity constraints
. exporting triggers
. end transportable tablespace metadata export
Export terminated successfully without warnings.
```

**10.**

```
C:\CERTDB>COPY C:\CERTDB\DISK5\TRANSPORT.DBF
C:\CERTDB\DISK6\TRANSPORT.DBF
```

**11.**

```
SVRMGR> DROP TABLESPACE TRANSPORT INCLUDING CONTENTS;
Statement processed.
```

**12.**

```
DEL C:\CERTDB\DISK5\TRANSPORT.DBF
```

**13.**

```
C:\CERTDB>IMP USERID='SYS/CHANGE_ON_INSTALL as SYSDBA'
FILE=C:\CERTDB\TRANSPORT.
DMP TRANSPORT_TABLESPACE=Y DATAFILES=C:\CERTDB\DISK6\TRANSPORT.DBF
```

Import: Release 8.1.6.0.0 - Production on Sun Mar 18 22:28:37 2001

(c) Copyright 1999 Oracle Corporation. All rights reserved.

Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production  
With the Partitioning option  
JServer Release 8.1.6.0.0 - Production

```
Export file created by EXPORT:V08.01.06 via conventional path
About to import transportable tablespace(s) metadata...
import done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character
set
. importing SYS's objects into SYS
. importing STUDENT's objects into STUDENT
.. importing table "TRANS_TEST"
Import terminated successfully without warnings.
```

**14.**

```
ALTER TABLESPACE TRANSPORT READ WRITE
```

**15.**

```
SVRMGR> SELECT TABLE_NAME
FROM DBA_TABLES
WHERE TABLESPACE_NAME='TRANSPORT';
TABLE_NAME
```

```

TRANS_TEST
```

```
SVRMGR> SELECT NAME FROM V$TABLESPACE;
NAME
```

```

SYSTEM
RBS
USERS
TEMP
TOOLS
INDX
CERTDB
TRANSPORT
```

```
SVRMGR> SELECT * FROM STUDENT.TRANS_TEST;
```

```
COL1

1234567890
```

# Managing Oracle Security

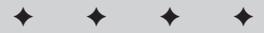
---

**U**sing Oracle to store critical data is a great idea. Ensuring that only those individuals who should have access to the data do is even better. This part of the book deals with database security, from understanding how configure security profiles, creating and managing users, assigning permissions and the different types of permissions available, all the way to simplifying the administration of permissions using roles.

This part starts with Chapter 15, which outlines the different password management features that are available in Oracle8i and explains how to create and manage profiles to enforce password management. Chapter 16 continues the security discussion by helping you learn how to create, drop, and modify users and the various parameters that can be specified for a user account.

Chapters 18 and 19 deal with the assignment of permissions in Oracle8i, starting with the types of privileges that can be granted in Oracle—system and object—followed by the ways that you can audit and track user activity in the database using Oracle’s built-in auditing features. In larger database environments, the assignment of permissions may become an administrative headache, so you will learn how roles can streamline the application of permissions and make your life easier. Both chapters will also outline the various data dictionary views that will tell you what permissions have been granted, and show what users are doing in the audit trail, if auditing has been enabled.

## V



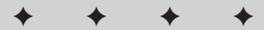
### In This Part

**Chapter 16**  
Managing Password Security and Resources

**Chapter 17**  
Managing Users

**Chapter 18**  
Managing Privileges

**Chapter 19**  
Managing Roles





# Managing Password Security and Resources

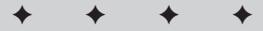
---

## EXAM OBJECTIVES

- ◆ Managing Password Security and Resources
  - Manage passwords using profiles
  - Administer profiles
  - Control use of resources using profiles
  - Obtain information about profiles, password management, and resources

# 16

CHAPTER



## CHAPTER PRE-TEST

1. In order to ensure that the amount of CPU time a user takes up during the execution of a single SQL statement is tracked by Oracle, what `INIT.ORA` parameter must be configured?
2. Is it possible to have one set of rules for password verification for one group of users and a different set of password verification rules for a second set of users? If so, how can this be accomplished; if not, why not?
3. What data dictionary views allow a DBA to determine which profile has been assigned to the user `JSmith` and what the profile limits are?
4. Who can drop profiles?
5. If you set the value for the `PASSWORD_REUSE_MAX` limit to 5 in a profile, what value must be specified for `PASSWORD_REUSE_TIME`?
6. Is it possible to have profiles in an Oracle database that are not assigned to any users?
7. What do you need to do to ensure that `COMPOSITE_LIMIT` is properly configured?
8. When will an account that has been locked out because it exceeded `FAILED_LOGIN_ATTEMPTS` be unlocked?
9. What tools can you use to create, alter, or drop profiles?
10. What possible values can the password verification function of a profile return?

In most organizations, the data that is stored in an Oracle database contains sensitive information that needs to be protected. Access to the data should only take place using appropriate tools and by those individuals who have been specifically granted permissions to the database and its data. Furthermore, only the DBA and those to whom the DBA has granted permission should manipulate objects in the database (that is, creation, modification, and removal).

Oracle provides many complementary ways to ensure that your databases are properly protected. This includes using system and object permissions to grant and revoke access to database data and the manipulation of database objects, quotas to limit the use of disk resources, profiles to control the use of system resources, as well as the management of passwords. All of these make up the security domain within Oracle. In this chapter you will find out the role played by profiles to manage password security and resource usage.

## Overview of Oracle Security

In an Oracle database, the DBA determines which users are allowed to access a database. The settings that can be defined for users to determine the level of access are the user's security domain. The security domain exists in each database that the user wants access to, which means that the DBA of each database has control over its users and what actions each user is allowed to perform. If an individual needs to gain access to more than one database, user accounts need to be established for the individual in each database to which access is required.

The security domain contains a number of elements including:

- ♦ **Authentication Mechanism** — The Authentication Mechanism determines how a user will be authenticated when requesting access to the database. Oracle supports database authentication (where the user account and password are stored in the database and verified by Oracle), operating system authentication (where the user account and password are created outside of Oracle and then mapped to the Oracle database), and network authentication (where the user account is authenticated by an external source such as a RADIUS server, or a Windows NT/2000 domain controller).



Operating system authentication is covered in chapters 2 and 17. Database authentication is covered in Chapter 17. Network authentication is beyond the scope of the "Oracle8i: Architecture and Administration" exam and is not covered in this book. For more information on network authentication please refer to the *Oracle8i Administrator's Guide* in the Oracle documentation set.

- ♦ **Tablespace Quotas** — Tablespace quotas determine how much physical disk space a user's objects may consume on a tablespace. The DBA assigns tablespace quotas to users to ensure that objects created do not fill up a tablespace and prevent other users' objects from growing as needed. The DBA

can assign no quotas to a user, in which case the user will not be able to create segments even if he or she has appropriate permissions, or any amount of disk space in any tablespace in the database.



Assigning tablespace quotas to users is covered in more detail in Chapter 17.

- ♦ **Default Tablespace** — When a user account is created in the database, the DBA should assign a default tablespace to the user. The default tablespace will be used to store segments that the user creates for which the user does not specify a tablespace in the CREATE statement. In order for the creation to succeed, the user must also have a quota on the default tablespace. If the DBA does not specify a default tablespace when creating a user account, Oracle assigns the SYSTEM tablespace as the default tablespace for the user (not recommended).
- ♦ **Temporary Tablespace** — A temporary tablespace is also assigned to the user account when it is first created. Unless otherwise specified, the user's temporary tablespace will be SYSTEM. You should always specify a temporary tablespace for the user and ensure that the tablespace you select is one created with the CREATE TEMPORARY TABLESPACE command.



Assigning a default tablespace and a temporary tablespace to users is covered in more detail in Chapter 17.

- ♦ **Account Locking and Password Management** — After you have created a user account, good security practices should ensure that anyone not authorized to access the database and attempting to do so is not successful, that users change their passwords on a regular basis, that passwords be of a minimum length and, possibly, adhere to certain rules established by the DBA or corporate policy, and so on. This is accomplished by creating a profile and specifying password management and account lockout rules within the profile.
- ♦ **Resource Limits** — Just because a user has access to the database and may have been granted permissions to query or modify data, does not mean that he or she should take over resources on the server and prevent others from using the database. Configuring a profile and specifying resource limits allows you to control the usage of system resources for an individual command or for the user's session.



Creating profiles to control resource usage and account lockout/password management policy is covered in more detail later in this chapter.

- ♦ **Direct Privileges** — The Oracle security model adheres to a simple principle: That which is not explicitly permitted is implicitly denied. This means that if a user needs to gain access to a table to query it, or wants to create, alter, or drop objects, he or she will need permissions to do so. The DBA or the object owner determines what those permissions are and can assign them directly to the user. However, direct assignment of permissions (privileges) is normally not performed as it may make the DBA's job more difficult.



Assigning privileges to users is covered in more detail in Chapter 18.

- ♦ **Role Privileges**— To make the assignment and management of privileges easier when many users need to perform similar tasks, Oracle uses the concept of roles. Roles are containers for permissions that can be assigned to users or other roles. Any privileges that have been assigned to a role are also automatically available to the user to whom the role has been assigned. This makes it easier to revoke or grant many privileges to a single user — simply grant or revoke the role.



Assigning privileges to roles and assigning roles to users is covered in more detail in Chapter 18.

## Profiles

One aspect of the Oracle security domain deals with ensuring that password management and account lockout policies for the database are adhered to. These may be set at the enterprise level and need to be enforced at the database level. Furthermore, as a DBA, you may need to ensure that a database is available to all users and that no one user is able to invoke a `SELECT` statement that performs a large query and consumes all system resources, as an example. The creation and management of account lockout and password policies, as well as limiting resource usage for a user's session or an individual SQL statement, is handled in Oracle by the use of profiles.

A profile is an Oracle object that allows you to set both password management and resource limits. The types of things that you can specify in a profile include:

- ♦ The length of time a password is valid for.
- ♦ The minimum time between password changes.
- ♦ The number of passwords to keep in history so that a user does not use the same password over and over.
- ♦ Rules for password complexity and length.
- ♦ Rules for account lockout, including the number of invalid attempts within a defined period of time.
- ♦ The maximum CPU time allowed a user for a session or a single SQL statement.
- ♦ The amount of IO that a user may perform in a session or while executing a single SQL statement.
- ♦ The maximum inactivity period before a user is automatically disconnected from the database.
- ♦ The maximum amount of time that a user may be connected in a single session.

- ♦ The maximum number of simultaneous sessions to the database that a user may have.
- ♦ The maximum amount of memory that a user session may consume in a Multi-Threaded Server (MTS) environment.

## Overview of profiles

In every Oracle database a single profile is created when you create the database. This profile is called `DEFAULT` and places no limits on either password and account lockout, or on resource utilization. In other words, the settings for both resource usage and password management are initially configured as unlimited. You can change the settings of the `DEFAULT` profile to conform to your requirements and they will then be applied to all users in the database assigned the `DEFAULT` profile.

Alternatively, a database administrator may create additional profiles dealing with password or account lockout issues, resource management settings, or both. Once created, a profile can be assigned to a user account as it is created, or it can be assigned to the user with the `ALTER USER` command. Any settings in the profile will then apply to the user the next time he or she connects to the database. A user may only have one profile active at one time, so you need to ensure that the settings within the profile match the requirements of each user. For this reason, you may want to investigate the characteristics and needs of the users of your database to determine what profiles need to be created.

When deciding to make use of profiles, it is important to understand what settings are always applied and which require that you change your database and instance configuration.

Because of the very nature of security requirements, Oracle ensures that password management and account lockout settings in profiles are always enforced. Any settings dealing with security policy are considered important enough that simply configuring them enables them. This can cause phone calls to the support desk if not properly implemented and understood by the DBA.

The utilization of system resources, such as CPU, and disk IO is not automatically enforced by Oracle. In order to have these aspects of a profile limit a user's actions, you need to enable them using either an `INIT.ORA` parameter or by changing the value of the `RESOURCE_LIMIT` initialization parameter with the `ALTER SYSTEM` command.

In order to enforce profile settings dealing with what are known as *kernel* resources (that is, use of CPU, disk IO, and so forth), you need to ensure that the following line is in the `INIT.ORA` file:

```
RESOURCE_LIMIT=TRUE
```

Another method is to issue the following command while connected to the instance as a DBA:

```
SQL> ALTER SYSTEM SET RESOURCE_LIMIT=TRUE;

System altered.

SQL>
```

If all you want to do is enforce password limits in your profiles, you do not need to enable the `RESOURCE_LIMIT` parameter in the `INIT.ORA` file.

## Profile settings

Understanding when certain settings are enforced is only one part of understanding how profiles work in Oracle. Another element is understanding what can be configured within a profile. The types of things you can configure are broken down into password management settings, or kernel resource settings.

### Password management

**Objective**

Manage passwords using profiles

When setting security elements of profile relating to password management and account lockout, the types of things you can configure can be roughly grouped in the following categories:

- ♦ **Account Locking** — Allows you to configure the rules under which a user account is locked if a user does not provide the correct password after a specified number of attempts. You can also specify how long the account will remain locked, which may include forever (or until a DBA unlocks the account).
- ♦ **Password Aging and Expiration** — Oracle allows you to limit the length of time a password is allowed to be used, the grace period after the password expires for the user to specify a new password, and the minimum length of time that a user must keep a password before being allowed to change it.
- ♦ **Password History** — You can specify the number of passwords that Oracle should remember for each user to whom the profile applies. This, in conjunction with the minimum password age, prevents users from using the same password over and over again by changing it back after being forced to change their password because it has expired.
- ♦ **Password Complexity Rules** — In order to ensure that users do not use obvious passwords or those that only contain lower case characters, for example, the DBA can create a password verification function in the database that is applied against the profile. The password complexity rules specified in the function are checked to ensure that the user's new password meets the criteria before the password change completes.

The specific profile settings dealing with password management that are allowed in an Oracle8i profile are the following:

- ♦ **FAILED\_LOGIN\_ATTEMPTS**— The number of failed login attempts that will cause the account to become locked. If someone attempts to login to the database using an account with this profile setting, Oracle will track the number of login failures. Once the value of **FAILED\_LOGIN\_ATTEMPTS** is crossed the user account becomes locked out.
- ♦ **PASSWORD\_LOCK\_TIME**— The number of days that the account will remain locked out after **FAILED\_LOGIN\_ATTEMPTS** has been crossed.

If the value of this parameter is set to **UNLIMITED**, the DBA will need to manually unlock the account using the **ALTER USER** command. The DBA can always unlock an account at any time, even before this setting unlocks it manually, as shown in the following example:

```
ALTER USER student ACCOUNT UNLOCK;
```

If you want to specify a time period less than a day, you can use any expression that evaluates to the time period you desire. For example, to lock an account for an hour you would specify a **PASSWORD\_LOCK\_TIME** of 1/24.

- ♦ **PASSWORD\_LIFE\_TIME**— The maximum number of days that a password may be used before it expires. When **PASSWORD\_LIFE\_TIME** is reached, the user is prompted to change his or her password if connecting to the instance using SQL\*Plus. Using other Oracle client software, or third-party tools, will require the programmer to trap for the password expiration and deal with it appropriately.
- ♦ **PASSWORD\_GRACE\_TIME**— After **PASSWORD\_LIFE\_TIME** is reached, and following the first successful login attempt from that point, **PASSWORD\_GRACE\_TIME** specifies the number of days that the user is allowed to use the old password and cancel the password change. The user must change his or her password within the period indicated by **PASSWORD\_GRACE\_TIME** after the first successful login attempt after **PASSWORD\_LIFE\_TIME** is reached. If the user fails to do so, the account is locked out after **PASSWORD\_GRACE\_TIME** expires.

Understanding the way this parameter works is important in ensuring that the DBA is not swamped with many requests to unlock accounts. The series of steps that Oracle uses to enforce this is the following:

1. **PASSWORD\_LIFE\_TIME** is reached. The next time the user logs on, he or she is prompted to change the password.
2. The user logs on and is prompted to change his or her password, after a warning that it has expired is displayed. At this point the counter for **PASSWORD\_GRACE\_TIME** starts. The user has **PASSWORD\_GRACE\_TIME** number of days to change his or her password or the account will be locked out.

- 3a. The user successfully changes his or her password. The counter for `PASSWORD_LIFE_TIME` is reset and the password will be valid for the number of days specified by `PASSWORD_LIFE_TIME`. `PASSWORD_GRACE_TIME` is no longer active.

OR

- 3b. `PASSWORD_GRACE_TIME` period expires (that is, the number of days since the last successful login attempt has exceeded the value of the parameter). At this point, the user account will be locked out.

- ♦ **PASSWORD\_REUSE\_TIME**— The minimum number of days before a password may be reused. This setting essentially says that a user, after changing his or her password, may not attempt to change his or her password back to one that they have used previously, unless the `PASSWORD_REUSE_TIME` period for the old password has expired.

In order to enforce this capability, Oracle keeps track of each password that the user has ever used. When a user changes his or her password, the counter for `PASSWORD_REUSE_TIME` for that password is started. If the user attempts to change his or her password to any password whose `PASSWORD_REUSE_TIME` has not yet expired, he or she will be shown an error and will not be able to perform the change. Once the `PASSWORD_REUSE_TIME` counter is reached, the user is free to change his or her password back to one he or she has used previously.

When the value of `PASSWORD_REUSE_TIME` is set to any value other than `DEFAULT` or `UNLIMITED`, the value of `PASSWORD_REUSE_MAX` **must** be set to `UNLIMITED`. In other words, you can effectively use either `PASSWORD_REUSE_TIME` or `PASSWORD_REUSE_MAX`, but not both at the same time.

- ♦ **PASSWORD\_REUSE\_MAX**— This parameter specifies the maximum number of times that each password may be used by a user. If the user attempts to change his or her password back to one whose `PASSWORD_REUSE_MAX` parameter has been crossed, he or she will be shown an error and the password change will not succeed.

The idea behind this parameter is to limit an individual from changing his or her password back to the same one over and over again. By setting this parameter, Oracle will keep track of the number of times a user uses each password and not allow any of the passwords that he or she has used to be used more times than `PASSWORD_REUSE_MAX`. Once a single password reaches the value of this parameter, it cannot be used again unless the parameter is changed to `UNLIMITED`.

When the value of `PASSWORD_REUSE_MAX` is set to any value other than `DEFAULT` or `UNLIMITED`, the value of `PASSWORD_REUSE_TIME` **must** be set to `UNLIMITED`. In other words, you can effectively use either `PASSWORD_REUSE_TIME` or `PASSWORD_REUSE_MAX`, but not both at the same time.

- ♦ **PASSWORD\_VERIFY\_FUNCTION**—This parameter specifies the name of a PL/SQL function created in the SYS schema that returns a Boolean (TRUE or FALSE) value to indicate whether the new password that the user has selected satisfies the complexity conditions specified in the function. This code in the function can be any valid PL/SQL code and may be used to enforce your unique password requirements.

The function specification is as follows:

```
function_name (
 user_id IN VARCHAR2(30),
 old_password IN VARCHAR2(30),
 new_password IN VARCHAR2(30))
RETURN BOOLEAN
```

The names of the `user_id`, `old_password`, and `new_password` parameters can be any valid name in PL/SQL and do not need to be those shown in the specification example.

As is evident from the above list of settings that can be specified for password management in Oracle, it is possible to lock down a system quite tightly, or leave it as loose as needed. The default behavior when you create a database is to configure all of the above parameters to UNLIMITED in the DEFAULT user profile, which all users receive by default unless another profile is specified for the user. If the default behavior is not what you desire, you may always alter it. Oracle provides a sample of what can be accomplished in a script that ships with the product.

### Sample password management script

Creating and configuring password management settings from scratch may be more work than a DBA needs or wants to perform. For this reason, and to show you what is possible, Oracle ships a script in the `ORACLE_HOME/rdbms/admin` directory called `utlpwdmg.sql` that modifies the DEFAULT profile and configures a password verify function.

To change the settings of the DEFAULT profile, the script issues the following command:

```
ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIFY_FUNCTION verify_function;
```

The password verify function, called *verify\_function* (as shown in the previous ALTER PROFILE command), ensures that a password is at least four characters long, is not the same as the username (or at least differs from it by three characters), contains

one alphabetic, one numeric, and one symbol character, and is different from the old password by at least four characters. The function also ensures that the password is not set to something obvious such as “oracle” or “database,” or “welcome,” as well as others.

As mentioned previously, if any of the password management settings are specified in a profile that is assigned to a user, they will be enforced regardless of the value of the RESOURCE\_LIMIT parameter. For this reason, you should ensure that you assign profiles to users that you want enforced on the next login attempt, and notify users of the change.

## Resource management

### Objective

Control use of resources using profiles

Oracle profiles also allow you to limit the use of system resources by a user. The limit can be specified for the session (known as a *per session* limit) or for a single SQL statement (known as a *per call* limit). Unlike password management settings, resource limits are only enforced when the value of the RESOURCE\_LIMIT parameter is set to true. The default for the RESOURCE\_LIMIT parameter is false, meaning that any resource management settings in a profile will be ignored, even if set.

The resource management settings that can be configured in a profile in Oracle8i are listed in Table 16-1.

**Table 16-1**  
**Profile Resource Management Settings**

| <b>Setting</b>  | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPU_PER_SESSION | <p>The total CPU time, measured in hundreds of a second, that a user is allowed to consume during a session. Once the limit is reached, the user's session is terminated with an Oracle server error message.</p> <p>This parameter specifies actual CPU time used by the session. This means that if the user session infrequently accesses the database, it may take some time to exhaust this value, whereas another user who has more activity in the database may reach it first.</p> <p>To reset this limit, the user needs to disconnect from the instance and connect again. After each connection, the limit is re-initialized (that is, each session starts with a clean slate and can use the CPU up to the value specified by CPU_PER_SESSION).</p> |

*Continued*

Table 16-1 (continued)

| <i>Setting</i>    | <i>Description</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPU_PER_CALL      | <p>The total CPU time, measured in hundredths of a second, that a user is allowed to consume for a single SQL statement. Once the limit is reached, the SQL statement is aborted and the transaction it is a part of is rolled back. The user's session remains connected.</p> <p>This parameter specifies actual CPU time used by the SQL statement and is intended to prevent users from issuing queries or database modifications that would not allow other users to continue to make use of the database. An example of this is a cross-join, or Cartesian product, of several tables without any join conditions.</p> <p>The limit is reset on every call to the database.</p>                                                                                                                                                                                                                                                                                                                                                              |
| SESSIONS_PER_USER | <p>The maximum number of concurrent sessions that a user may have at one time. If the user attempts to have more simultaneous connections to the database than allowed by SESSIONS_PER_USER, an Oracle error will be returned.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| CONNECT_TIME      | <p>The maximum amount of time, specified in minutes, that a user may remain connected to the instance. If one of the user's sessions exceeds the value of CONNECT_TIME, that session will be terminated and the user automatically disconnected.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| IDLE_TIME         | <p>The maximum amount of time, specified in minutes, that a user's session may remain connected to the instance while not performing any database activity. Oracle will keep track of the elapsed time from the last command that performed database activity until the IDLE_TIME value is reached. If the user does not initiate any database actions before IDLE_TIME is reached, the session will be terminated and an error returned to the user.</p> <p>It is important to note that IDLE_TIME is kept track of at the database instance level and not on the client side. When using SQL*Plus, issuing SQL*Plus commands, such as DEFINE or SET commands, will not perform any database activity and therefore not reset the IDLE_TIME counter. However, a DESCRIBE command when dealing with database objects will reset IDLE_TIME.</p> <p>The idea behind this parameter is to ensure that users don't walk away from their desk for prolonged periods leaving their workstations and the database vulnerable to unauthorized access.</p> |

| <b>Setting</b>            | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LOGICAL_READS_PER_SESSION | The number of blocks (both physical – from disk – and logical – from the database buffer cache) that the user is allowed to read during his or her session. Once the number of blocks specified by this parameter are read, the user will need to start another session in order to access data in the database.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| LOGICAL_READS_PER_CALL    | <p>The number of blocks (both physical – from disk – and logical – from the database buffer cache) that the user is allowed to read when executing an SQL statement. Once the number of blocks specified by this parameter is read, the SQL statement will be terminated and any transaction that it is a part of will be rolled back. The user's session will remain connected and other commands can be executed.</p> <p>This parameter, like CPU_PER_CALL, is intended to prevent a single SQL statement by a user from consuming all system resources and not allow other users to continue to perform their work.</p>                                                                                                                 |
| PRIVATE_SGA               | <p>In a Multi-Threaded Server (MTS) environment, this parameter specifies the maximum number of bytes that a user's session can occupy in the SGA. This includes memory used to hold the values of bind and session variables, cursors, and other session-specific parameters.</p> <p>If you are not connected to the database instance with an MTS connection, this parameter is ignored.</p>                                                                                                                                                                                                                                                                                                                                             |
| COMPOSITE_LIMIT           | <p>Specifies a numerical value that is the weighted average of four resource limits:</p> <p>CPU_PER_SESSION<br/>CONNECT_TIME<br/>LOGICAL_READS_PER_SESSION<br/>PRIVATE_SGA</p> <p>Setting COMPOSITE_LIMIT will enable Oracle to monitor all four of these parameter values and when the combination of all exceeds the value specified by COMPOSITE_LIMIT, the user's session will be terminated.</p> <p>As each environment and database is different, Oracle allows you to configure the cost of each of the four parameters using the ALTER RESOURCE COST command. The default cost of each of the four components of COMPOSITE_LIMIT is 0, which means that this setting in the profile is not enforced until you configure costs.</p> |

### Configuring resource costs for COMPOSITE\_LIMIT

The COMPOSITE\_LIMIT resource limit can be a handy way to ensure that a combination of users IO, CPU time, connect time, and, in the case of an MTS configuration, memory usage in the SGA does not exceed a pre-defined threshold. COMPOSITE\_LIMIT is a session-level limit, which means that when it is crossed the user's session is terminated. Using it, you can ensure that a user who hogs the CPU and also incurs a lot of IO is terminated more quickly than if any one of these parameters were set individually.

In order to make effective use of COMPOSITE\_LIMIT, you must configure costs to each of the four resources that make it up. The default configuration assigns a cost of zero (0) to all of the four elements thereby ensuring that COMPOSITE\_LIMIT is not enforced. To configure the weight of the resource limits that make up COMPOSITE\_LIMIT, you would issue the ALTER RESOURCE COST command, when connected as a DBA. The syntax of the command is as follows:

```
ALTER RESOURCE COST
 [CPU_PER_SESSION va lue]
 [CONNECT_TIME va lue]
 [LOGICAL_READS_PER_SESSION va lue]
 [PRIVATE_SGA va lue]
```

The value applied to any of the limits can be any whole integer and will be used to multiply the limit in the calculation to come up with the COMPOSITE\_LIMIT value. Any limit whose value has been set to 0 will not be included in the calculation. Similarly, when initially configuring the resource costs for any of the resource limits that make up COMPOSITE\_LIMIT, any resource limit that is omitted will have its value set to zero.

For example, issuing the following command will set the multiplier for CONNECT\_TIME to 10, meaning that each connected minute will represent a value of 10 when being used to determine if COMPOSITE\_LIMIT is exceeded. Similarly, as shown in the example, CPU\_PER\_SESSION will be assigned a multiplier of 3 and LOGICAL\_READS\_PER\_SESSION a multiplier of 1. PRIVATE\_SGA will not be included in the calculation for COMPOSITE\_LIMIT because it was not included in the command.

```
SQL> ALTER RESOURCE COST
2 CONNECT_TIME 10
3 CPU_PER_SESSION 3
4 LOGICAL_READS_PER_SESSION 1;
```

Resource cost altered.

```
SQL>
```

Using this example, if we had set the value for COMPOSITE\_LIMIT in the currently active user profile to 500, any combination of the resource limits previously shown will disconnect the user session. For example, a ten-minute connection using 1 second of CPU time and reading 101 blocks would force a disconnection of the user. The formula to calculate this is:

```
(CONNECT_TIME * 10) + (CPU_PER_SESSION * 3) +
LOGICAL_READS_PER_SESSION
```

If you recall, CPU\_PER\_SESSION is measured in hundredths of a second, so the composite limit calculation for the previous example would be

```
(10 minutes * 10) + ((1 second *100) * 3) + 101 = 501
```

As is also shown by this example, setting COMPOSITE\_LIMIT too low and configuring multipliers can actually cause the user to be disconnected from the instance after a small amount of activity. When setting COMPOSITE\_LIMIT and configuring resource costs, ensure that you set their values at such a level so that users will be able to perform their assigned duties and not be arbitrarily disconnected from the database. As a general rule, monitor disconnections and ensure that you periodically adjust the COMPOSITE\_LIMIT and resource costs.

To determine the currently running resource costs for your session, you can query the RESOURCE\_COST data dictionary view, as shown here:

```
SQL> SELECT * FROM RESOURCE_COST;

RESOURCE_NAME UNIT_COST

CPU_PER_SESSION 3
LOGICAL_READS_PER_SESSION 1
CONNECT_TIME 10
PRIVATE_SGA 0

SQL>
```

You should note that COMPOSITE\_LIMIT and resource costs will not be enforced unless the RESOURCE\_LIMIT parameter has been set to TRUE.

## Creating and managing profiles

### Objective

Administer profiles

Now that you have an understanding of what settings can be configured in a profile, you need to know how to configure profiles in your Oracle database. You can create and modify profiles, as well as assign them to users, using Oracle DDL statements, or through Oracle Enterprise Manager or DBA Studio. In order to create, alter or drop profiles you must be a DBA, or have been granted the CREATE PROFILE, ALTER PROFILE or DROP PROFILE system privilege, respectively. To assign profile to users, you need to have been granted the CREATE USER or ALTER USER system privilege.



For information on granting and revoking system privileges, please refer to Chapter 18.

## Creating profiles

The most common way to create a profile is to issue the CREATE PROFILE DDL command when connected to the instance as a DBA. The syntax of the CREATE PROFILE command is as follows:

```
CREATE PROFILE profile_name LIMIT
 [FAILED_LOGIN_ATTEMPTS value]
 [PASSWORD_LOCK_TIME value]
 [PASSWORD_LIFE_TIME value]
 [PASSWORD_GRACE_TIME value]
 [PASSWORD_REUSE_MAX | PASSWORD_REUSE_TIME value]
 [PASSWORD_VERIFY_FUNCTION function_name|NULL|DEFAULT]
 [SESSIONS_PER_USER value]
 [CPU_PER_SESSION value]
 [CPU_PER_CALL value]
 [CONNECT_TIME value]
 [IDLE_TIME value]
 [LOGICAL_READS_PER_SESSION value] value]
 [LOGICAL_READS_PER_CALL value]
 [COMPOSITE_LIMIT value]
 [PRIVATE_SGA bytes [K|M]]
```

When creating a profile, any value specified for resource limits must be a whole integer. In other words, you cannot specify a value of 1/2 for CONNECT\_TIME. However, the majority of password management limits do support expressions. Therefore, it is perfectly fine to specify a PASSWORD\_LOCK\_TIME of 15/1440 to indicate a lockout period of 15 minutes. In the following example, a PASSWORD\_GRACE\_TIME of 12 hours, and a PASSWORD\_LOCK\_TIME of 30 minutes are being specified, along with some resource limits:

```
SQL> CREATE PROFILE developer_profile LIMIT
2 PASSWORD_GRACE_TIME 12/24
3 PASSWORD_LIFE_TIME 45
4 PASSWORD_LOCK_TIME 30/1440
5 FAILED_LOGIN_ATTEMPTS 5
6 PASSWORD_VERIFY_FUNCTION DEFAULT
7 PASSWORD_REUSE_TIME 5
8 PASSWORD_REUSE_MAX UNLIMITED
9 IDLE_TIME 30
10 CONNECT_TIME 600
11* SESSIONS_PER_USER UNLIMITED
SQL> /
```

Profile created.

```
SQL>
```

When specifying limits for a profile, you do not need to include all of the limits in the CREATE PROFILE statement but only those settings that you want to specify for the profile being created. All of the other limits in the profile will be assigned a value of DEFAULT automatically, which means that they will inherit the settings of the DEFAULT profile. If you want to explicitly assign the DEFAULT profile's setting to

a resource or password management limit, you can specify the keyword `DEFAULT` when creating the profile. As you may have also noticed in the previous example, to specify that a resource or password management limit should not be hindered in any way, you can specify the keyword `UNLIMITED` to configure a setting.

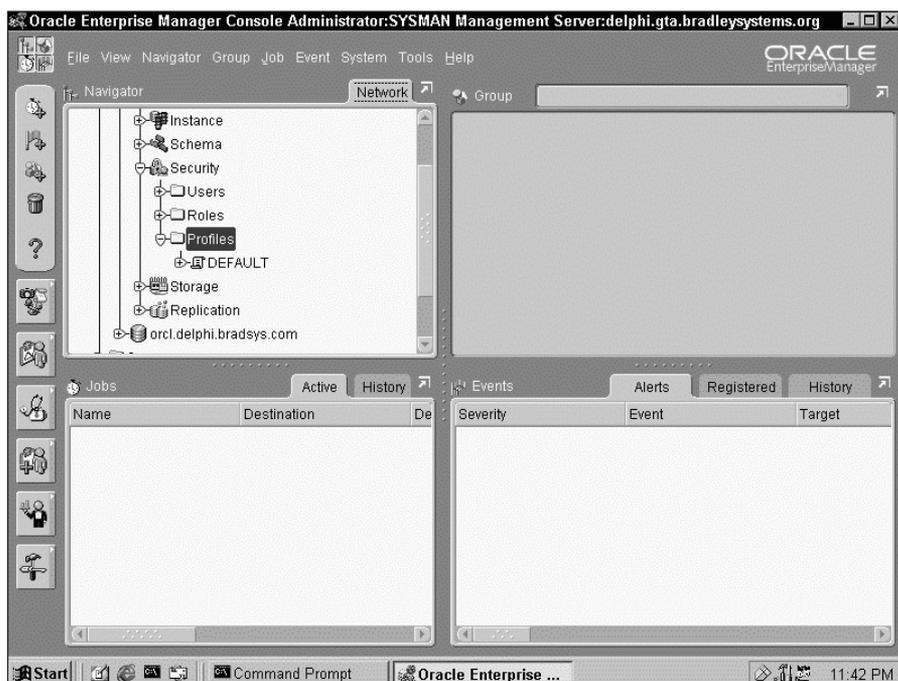
Once you have created a profile, you can assign it to a user using the `ALTER USER` command, if the user already exists in the database, or when adding a user to the database with the `CREATE USER` command. These will be covered in the next chapter.

### Creating profiles using Oracle Enterprise Manager

You can also create profiles with Oracle Enterprise Manager using the following steps:

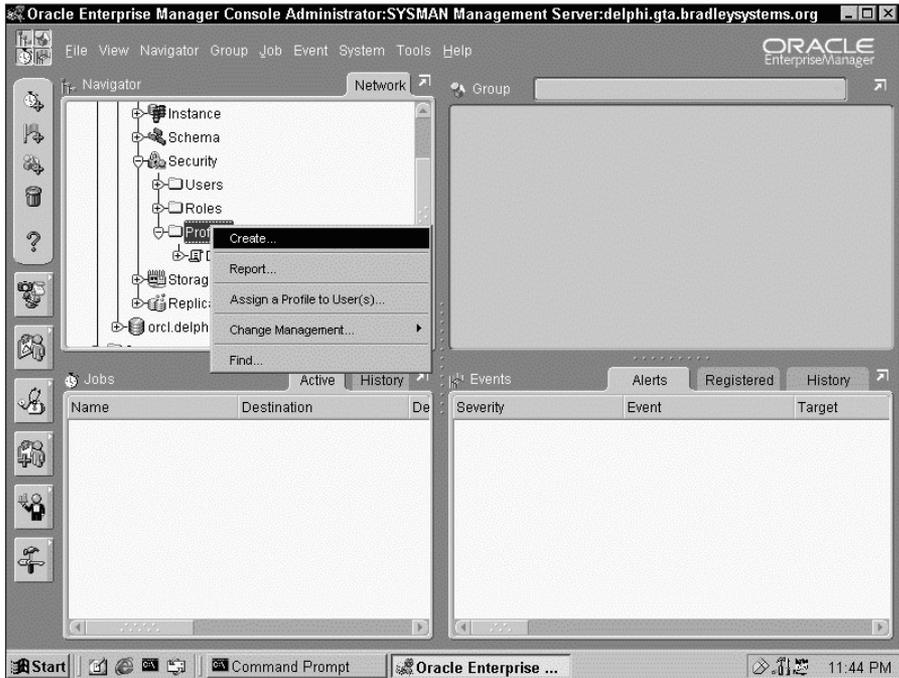
#### STEP BY STEP: Creating a Profile Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to create the profile.
2. Expand the database and then Security, and then Profiles, as shown in Figure 16-1.



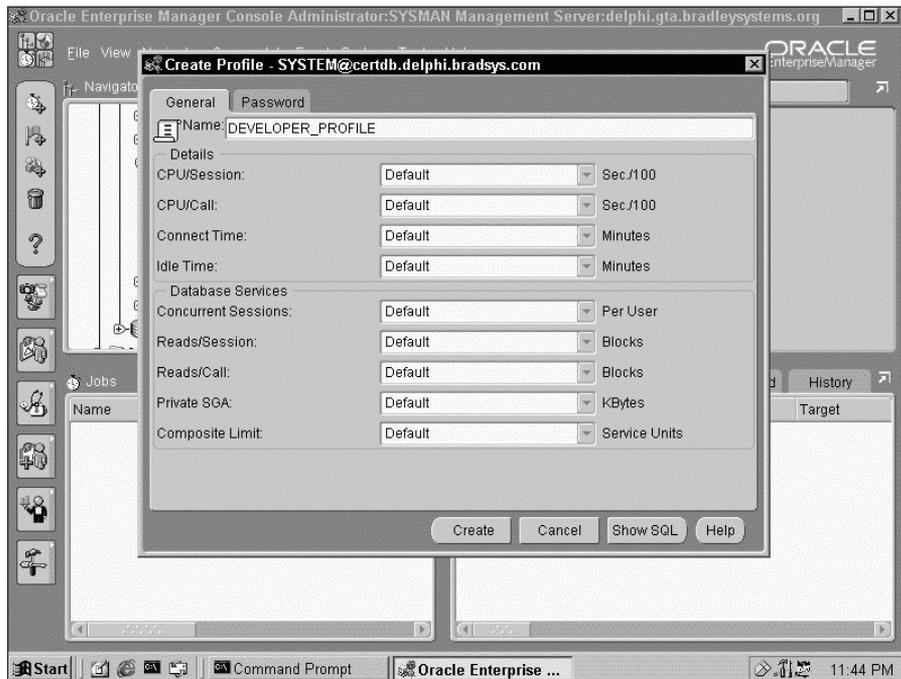
**Figure 16-1:** Expanding the Security node of the database where you want to create the profile will display the Profiles container.

3. Right-click on Profiles and then select Create, as shown in Figure 16-2, to create a new profile.



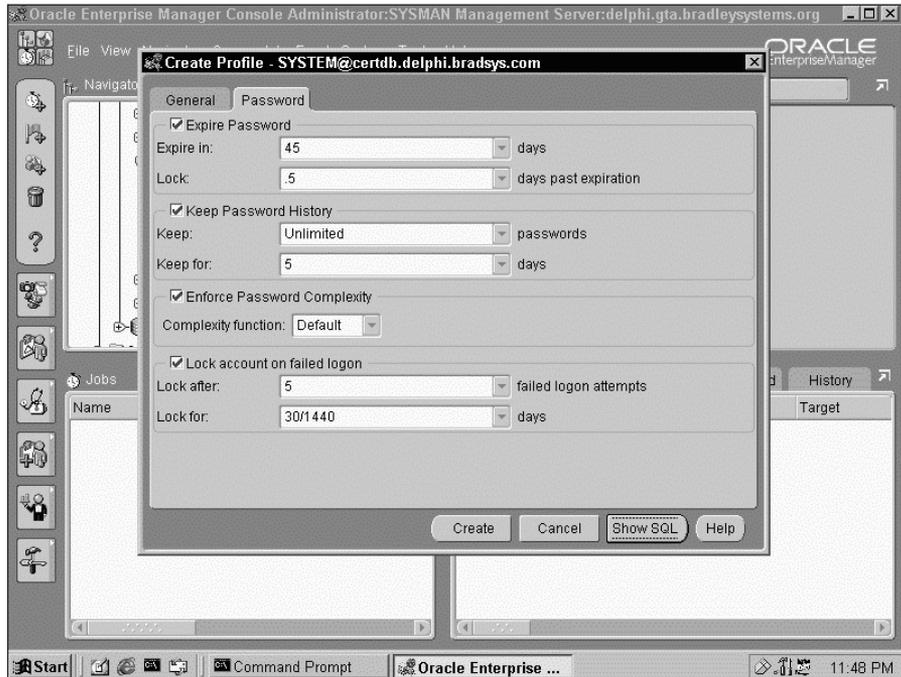
**Figure 16-2:** Right-click on Profiles and select “Create” to create a new profile.

4. In the dialog box that comes up, enter a name for the profile and then select resource limits on the General tab, as shown in Figure 16-3.



**Figure 16-3:** Enter a name for the profile and provide resource limits on the General tab.

5. To enter password management limits, select the Password tab and enter appropriate values, as shown in Figure 16-4.



**Figure 16-4:** Click on the Password tab to enter password management limits.

- When you have made all your selections, click on “Create” to create the profile. If there are no errors, you will be shown a dialog box indicating success, as presented in Figure 16-5.

## Modifying profiles

In some situations, it may be necessary to change profile settings to provide users with the functionality they expect, or to deal with changes in the database, such as when the number of blocks in the database have grown and LOGICAL\_READS\_PER\_CALL is being triggered regularly, for example. To modify a profile you need to be connected to the database as a DBA and issue the ALTER PROFILE command, whose syntax is as follows:

```
ALTER PROFILE profile_name LIMIT
 [FAILED_LOGIN_ATTEMPTS value]
 [PASSWORD_LOCK_TIME value]
 [PASSWORD_LIFE_TIME value]
 [PASSWORD_GRACE_TIME value]
 [PASSWORD_REUSE_MAX | PASSWORD_REUSE_TIME value]
 [PASSWORD_VERIFY_FUNCTION function_name|NULL|DEFAULT]
 [SESSIONS_PER_USER value]
```

```

[CPU_PER_SESSION value]
[CPU_PER_CALL value]
[CONNECT_TIME value]
[IDLE_TIME value]
[LOGICAL_READS_PER_SESSION value]
[LOGICAL_READS_PER_CALL value]
[COMPOSITE_LIMIT value]
[PRIVATE_SGA bytes [K|M]]

```

As was the case when creating a profile, you only need to include those resource and password management limits that you want to change in the ALTER PROFILE command. Any settings not specified will remain as set when the profile was initially created. For example, to add a CPU\_PER\_CALL limit to the developer\_profile created previously, you would issue the following command:

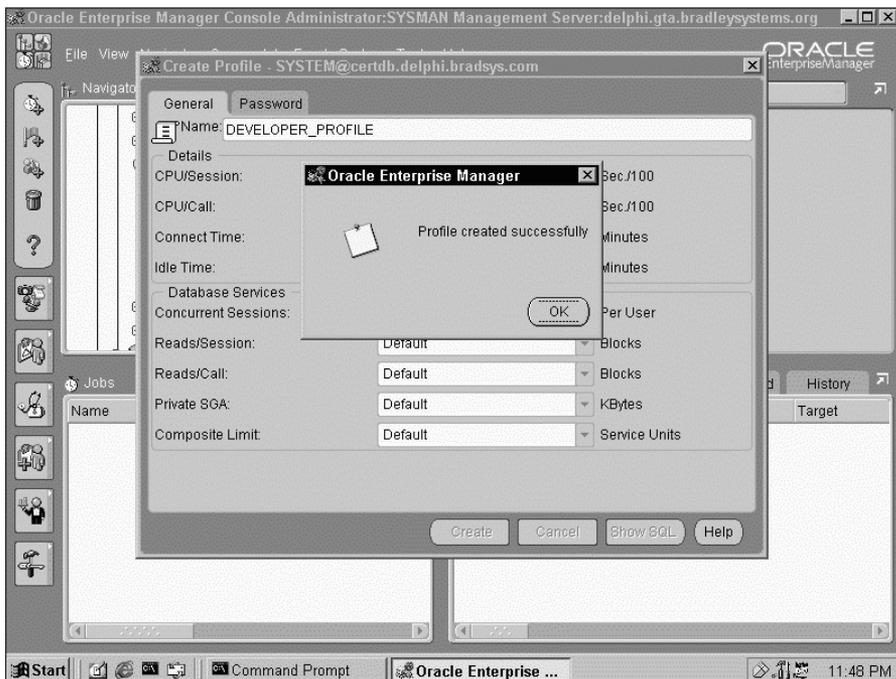
```

SQL> ALTER PROFILE developer_profile LIMIT
 2 CPU_PER_CALL 500;

```

Profile altered.

SQL>



**Figure 16-5:** You have successfully created a new profile.

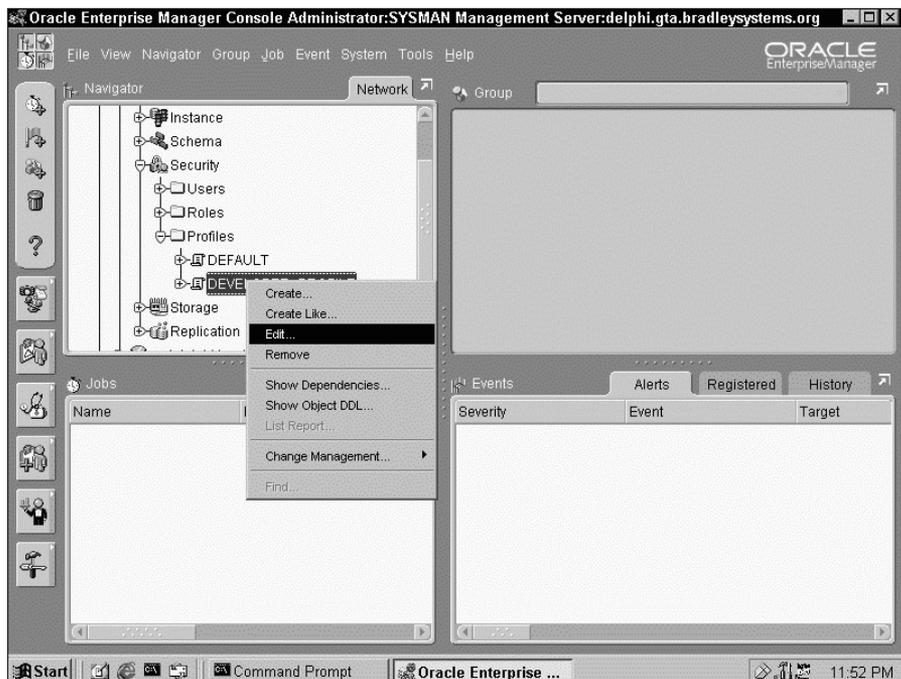
Any changes made to the profile will not be applied to the current user session but only to future sessions. This means that if a user was currently connected to the instance and that user was assigned the developer\_profile, the CPU\_PER\_CALL limit specified previously would not take effect until the user disconnected from the instance and connected again. Any new connections by users assigned the profile will immediately have the new setting take effect.

### Changing profiles using Oracle Enterprise Manager

You can also alter profiles with Oracle Enterprise Manager using the following steps:

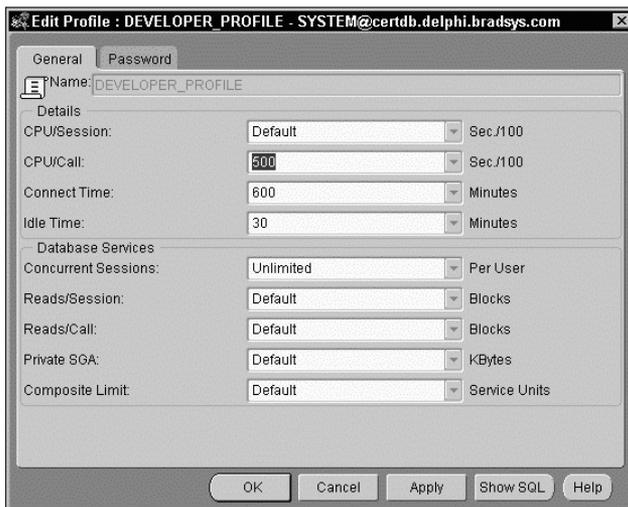
#### STEP BY STEP: Modifying a Profile Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database where the profile you want to modify is located.
2. Expand the database and then Security, and then Profiles.
3. Right-click on the profile you wish to modify and then select Edit, as shown in Figure 16-6.



**Figure 16-6:** Right-click on the profile you wish to change and select “Edit”.

4. In the Edit Profile dialog box make the necessary changes, and then select Apply, as shown in Figure 16-7. Click OK when you are done.



**Figure 16-7:** Make changes to the profile in the Edit Profile dialog box and then select Apply. Click OK to exit the dialog box.

## Dropping profiles

The easiest way to drop a profile is to use the DROP PROFILE command, with the following syntax:

```
DROP PROFILE profile_name [CASCADE]
```

Once you issue the DROP PROFILE command, Oracle checks to see if any users currently have the profile assigned. If so, you will be presented with an error as in the following example:

```
SQL> DROP PROFILE developer_profile;
DROP PROFILE developer_profile
*
ERROR at line 1:
ORA-02382: profile DEVELOPER_PROFILE has users assigned, cannot drop without
CASCADE
```

```
SQL>
```

In order to drop a profile that is assigned to users, and at the same time have the DEFAULT profile assigned to any affected users, you need to include the CASCADE clause in the DROP PROFILE command. You should, prior to doing this, verify which users have the profile assigned by querying the DBA\_USERS data dictionary view, as in the following example:

```
SQL> SELECT USERNAME FROM DBA_USERS
2 WHERE PROFILE='DEVELOPER_PROFILE';
```

```
USERNAME

STUDENT
```

```
SQL>
```

Knowing which users have the profile assigned allows you to assign other profiles to the users explicitly, or, if you decide to use the CASCADE option, which users will now have the DEFAULT profile assigned, and thereby deal with any consequences this may represent.

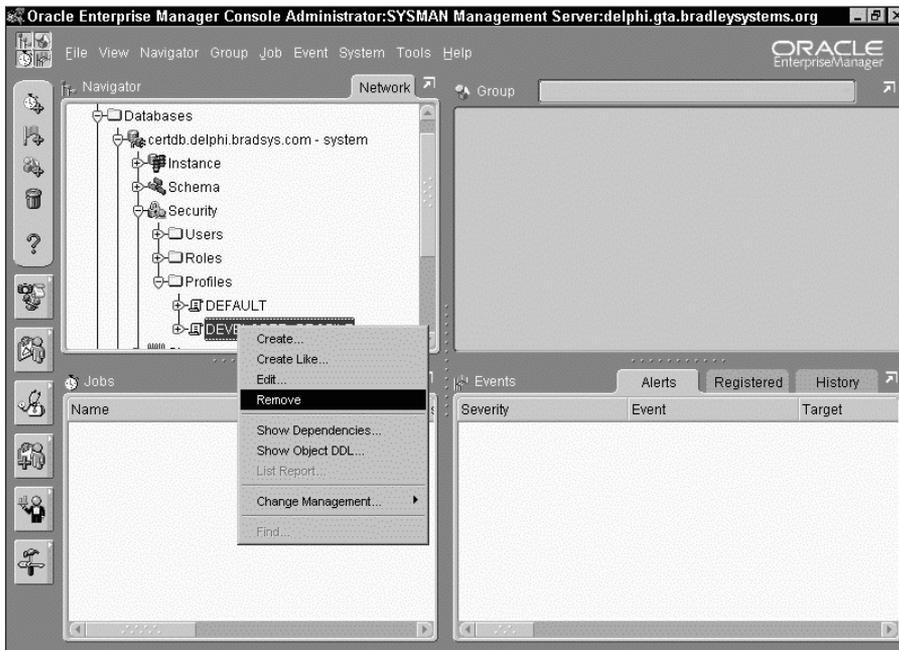
It is not possible to drop the DEFAULT profile, although you can modify its values with the ALTER PROFILE command.

### Dropping profiles using Enterprise Manager

You can also drop profiles using Oracle Enterprise Manager, as shown in the following steps:

#### STEP BY STEP: Deleting a Profile Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database where the profile you want to delete is located.
2. Expand the database then Security, and then Profiles.
3. Right-click on the profile you wish to delete and select Remove, as shown in Figure 16-8
4. If the profile is currently assigned to a user, you will be presented with a warning, as shown in Figure 16-9. You then have the option to select “Yes” and delete the profile and assign all affected users the DEFAULT profile, or cancel the operation by selecting “No.”



**Figure 16-8:** Right-click on the profile and select “Remove” to delete it.



**Figure 16-9:** If the profile to be deleted is assigned to users, you will be prompted to confirm your actions



**Tip**

If you are using DBA Studio and not the Oracle Enterprise Manager console the process for adding, modifying or dropping profiles is similar to what has been presented, with one deviation—your main interface will only list the database registered and will be similar to that shown in Figure 16-10.

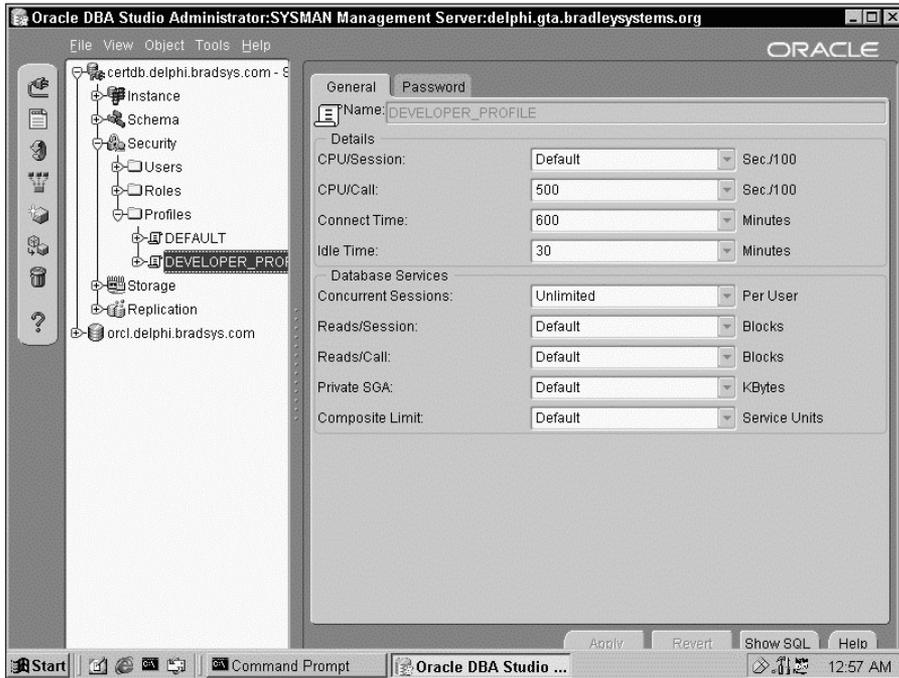


Figure 16-10: DBA Studio presents a list of instances in an Explorer-style interface.

## Getting information on profiles

### Objective

Obtain information about profiles, password management, and resources

To determine which profiles exist in the database and what their settings are, you can query the `DBA_PROFILES` data dictionary view. In order to be able to query this view you must connect to the instance as a DBA. The information presented is similar to the following:

```
SQL> col resource_name format a27
SQL> col profile format a20
SQL> col limit format a10
SQL> SELECT * FROM DBA_PROFILES
 2 WHERE PROFILE='DEVELOPER_PROFILE';
```

| PROFILE           | RESOURCE_NAME     | RESOURCE | LIMIT     |
|-------------------|-------------------|----------|-----------|
| DEVELOPER_PROFILE | COMPOSITE_LIMIT   | KERNEL   | DEFAULT   |
| DEVELOPER_PROFILE | SESSIONS_PER_USER | KERNEL   | UNLIMITED |

|                   |                           |          |           |
|-------------------|---------------------------|----------|-----------|
| DEVELOPER_PROFILE | CPU_PER_SESSION           | KERNEL   | DEFAULT   |
| DEVELOPER_PROFILE | CPU_PER_CALL              | KERNEL   | 500       |
| DEVELOPER_PROFILE | LOGICAL_READS_PER_SESSION | KERNEL   | DEFAULT   |
| DEVELOPER_PROFILE | LOGICAL_READS_PER_CALL    | KERNEL   | DEFAULT   |
| DEVELOPER_PROFILE | IDLE_TIME                 | KERNEL   | 30        |
| DEVELOPER_PROFILE | CONNECT_TIME              | KERNEL   | 600       |
| DEVELOPER_PROFILE | PRIVATE_SGA               | KERNEL   | DEFAULT   |
| DEVELOPER_PROFILE | FAILED_LOGIN_ATTEMPTS     | PASSWORD | 5         |
| DEVELOPER_PROFILE | PASSWORD_LIFE_TIME        | PASSWORD | 45        |
| DEVELOPER_PROFILE | PASSWORD_REUSE_TIME       | PASSWORD | 5         |
| DEVELOPER_PROFILE | PASSWORD_REUSE_MAX        | PASSWORD | UNLIMITED |
| DEVELOPER_PROFILE | PASSWORD_VERIFY_FUNCTION  | PASSWORD | DEFAULT   |
| DEVELOPER_PROFILE | PASSWORD_LOCK_TIME        | PASSWORD | .0208     |
| DEVELOPER_PROFILE | PASSWORD_GRACE_TIME       | PASSWORD | .5        |

16 rows selected.

SQL>

The RESOURCE column of the DBA\_PROFILES view tells you whether the profile limit deals with password management, such as PASSWORD\_LOCK\_TIME, or is a kernel limit that prevents a user to whom the profile has been assigned from taking over the system, or aspects of it. Any value of DEFAULT in the LIMIT column indicates that the specific profile limit will inherit its value from the DEFAULT profile. Should the value of the corresponding limit for the DEFAULT profile change, it will also change in this profile.

If you wanted to determine which profiles were assigned to which users, or to which user a specific profile is assigned, you can query the DBA\_USERS view, as shown in the following examples:

```
SQL> SELECT USERNAME, PROFILE FROM DBA_USERS;
```

| USERNAME | PROFILE           |
|----------|-------------------|
| SYS      | DEFAULT           |
| SYSTEM   | DEFAULT           |
| OUTLN    | DEFAULT           |
| DBSNMP   | DEFAULT           |
| STUDENT  | DEVELOPER_PROFILE |

```
SQL> SELECT USERNAME FROM DBA_USERS
2 WHERE PROFILE='DEFAULT';
```

| USERNAME |
|----------|
| SYS      |

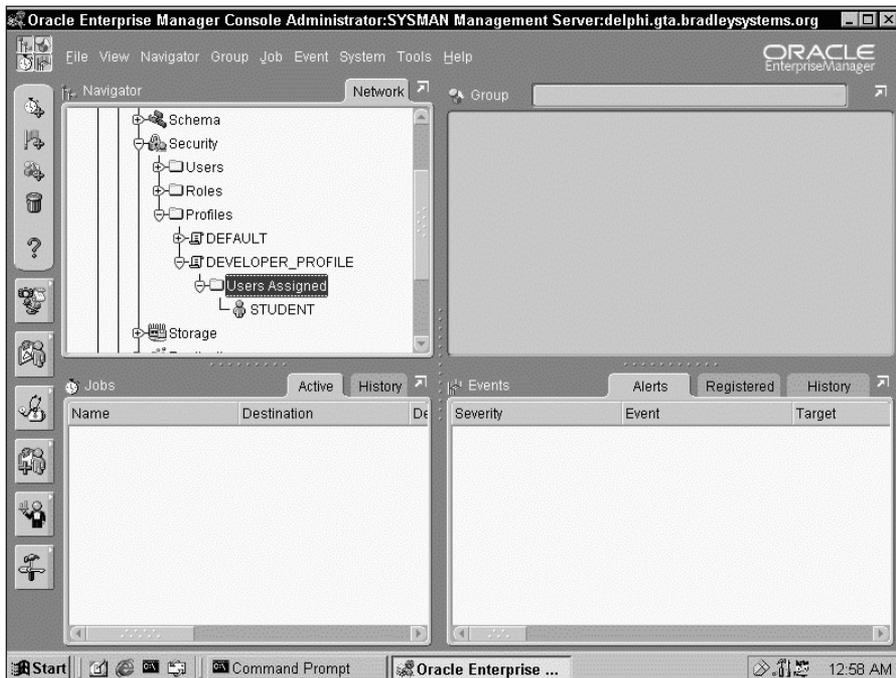
```

SYSTEM
OUTLN
DBSNMP

```

```
SQL>
```

You can also use Oracle Enterprise Manager or DBA Studio to view profile information, as shown in the steps outlined previously. To determine which users a profile has been assigned to, expand the “Users Assigned” node under the profile, as shown in Figure 16-11. You can also use this interface to assign the profile to a user.



**Figure 16-11:** The Users Assigned node of the profile lists users to whom the profile has been assigned using the CREATE USER or ALTER USER statement.

If you wanted to determine if an account is locked out, or when it was locked out, as well as when an account’s password expires, you can query these columns in the DBA\_USERS view, as shown here:

```

SQL> col username format a15
SQL> col account_status format a20
SQL> SELECT USERNAME, ACCOUNT_STATUS, LOCK_DATE, EXPIRY_DATE
 2 FROM DBA_USERS;

```

```

USERNAME ACCOUNT_STATUS LOCK_DATE EXPIRY_DA

```

```

SYS OPEN
SYSTEM OPEN
OUTLN OPEN
DBSNMP OPEN
STUDENT OPEN
 22-JUL-01

SQL>

```

## Key Point Summary

In preparing for the “Oracle®i DBA: Architecture and Administration” exam, please keep these points regarding password and resource limits, and profiles in mind:

- ♦ Profiles are a named set of limits for managing password expiration and aging, account lockout, password history, password verification, and kernel resource limits.
- ♦ A profile can be assigned to a user. A user may have one and only one active profile.
- ♦ Password management limits in a profile are always enforced by Oracle.
- ♦ Kernel resource limits are only enforced if the value of the RESOURCE\_LIMIT Oracle initialization parameter is set to TRUE. The default value for this parameter is FALSE.
- ♦ Profiles can be created using Oracle Enterprise Manager (OEM) or the CREATE PROFILE command.
- ♦ You can modify profiles using the ALTER PROFILE command or OEM. Any changes to a profile will take affect the next time a user to whom the profile has been assigned connects to the database.
- ♦ You can drop profiles using the DROP PROFILE command or OEM. If the profile to be dropped is currently assigned to users, you must also specify the CASCADE option to have the DEFAULT profile automatically assigned to all affected users.
- ♦ Only DBAs, or users granted the CREATE PROFILE, ALTER PROFILE, or DROP PROFILE privileges can manipulate profiles.
- ♦ You can assign a profile to a user with the ALTER USER or CREATE USER commands.
- ♦ Information of profile settings is available by querying the DBA\_PROFILES data dictionary view, or through OEM.
- ♦ To determine which profiles have been assigned to users, you can query the DBA\_USERS view, or use OEM.



# STUDY GUIDE

---

In preparation for the “Oracle8i: Architecture and Administration” exam, use the study guide section to test your knowledge of profiles. Perform the labs, answer the assessment questions and review the scenarios. This will let you determine which areas you may need to brush up on.

## Assessment Questions

1. You execute the following CREATE PROFILE command:

```
1 CREATE PROFILE MYPROFILE LIMIT
2 PASSWORD_LIFE_TIME 45
3 PASSWORD_REUSE_MAX 5
4 PASSWORD_REUSE_TIME 2
5 FAILED_LOGIN_ATTEMPTS 3
6 PASSWORD_LOCK_TIME 1/1440
7 PRIVATE_SGA 512K;
```

Which lines will cause the operation to fail? (Choose two correct answers.)

- A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5
  - F. 6
  - G. 7
2. If you configure a PASSWORD\_LOCK\_TIME of 1 and a FAILED\_LOGIN\_ATTEMPTS of 5 in a profile, when will the account be locked out and for how long? (Choose two correct answers.)
- A. On the fifth logon attempt.
  - B. On the sixth logon attempt.
  - C. The account will be locked out for one minute.
  - D. The account will be locked out for one hour.
  - E. The account will be locked out for one day.

3. How can you assign a profile called MYPROFILE that you have just created to an existing user in the database called Henry? (Choose the best answer.)
- A. ALTER PROFILE MYPROFILE ASSIGN HENRY
  - B. ALTER USER HENRY MYPROFILE
  - C. ALTER PROFILE HENRY ASSIGN MYPROFILE
  - D. ALTER HENRY PROFILE MYPROFILE
  - E. ALTER USER HENRY PROFILE MYPROFILE
4. If you configure a PASSWORD\_GRACE\_TIME of 2 and a PASSWORD\_LIFE\_TIME of 30 in a profile called MYPROFILE, when will the grace period begin? (Choose the best answer.)
- A. When the old password expires.
  - B. When the profile is assigned to the user.
  - C. When the administrator changes the user's password.
  - D. The first time the user logs on after the password has expired.
  - E. The third time the user is prompted to change his or her password.
5. If you configure a PASSWORD\_GRACE\_TIME of 2 and a PASSWORD\_LIFE\_TIME of 30 in a profile called MYPROFILE, what happens if the user does not change his or her password before the grace period expires? (Choose the best answer.)
- A. The account will be locked out.
  - B. The administrator will need to change the user's password.
  - C. The user will be allowed to keep his or her existing password.
  - D. The user will not be able to connect to the instance until the password is changed when prompted.
  - E. The user will be assigned a default password of ORACLE.

6. You have configured an `IDLE_TIME` of 30 and a `PASSWORD_LIFE_TIME` of 30 in a profile assigned to Henry. Henry mentions that he was prompted to change his password the last time he connected to the instance, and complements you on how the database lets him continue where he left off after coming back from his one-hour lunch break. Why is the `IDLE_TIME` not being enforced? (Choose the best answer.)
- A. Henry has issued the command `ALTER SESSION SET RESOURCE_LIMIT=FALSE` to disable it.
  - B. The `INIT.ORA` file contains the parameter `RESOURCE_LIMIT=TRUE`.
  - C. You did not issue the command `ALTER SYSTEM SET RESOURCE_LIMIT=FALSE`.
  - D. You did not issue the command `ALTER USER HENRY SET RESOURCE_LIMIT=TRUE`.
  - E. The `INIT.ORA` file does not contain the parameter `RESOURCE_LIMIT=TRUE`.

7. You create a profile with the following command:

```
CREATE PROFILE MYPROFILE LIMIT
PASSWORD_LIFE_TIME 45
PASSWORD_REUSE_MAX 5
PASSWORD_REUSE_TIME UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PRIVATE_SGA 512K;
```

You assign the profile to Henry. While monitoring the memory usage of your instance, you notice that Henry's session is using 2MB of RAM. Why is the `PRIVATE_SGA` limit not being enforced? (Choose all correct answers.)

- A. Henry has issued the command `ALTER SESSION SET RESOURCE_LIMIT=FALSE` to disable it.
  - B. Henry is connected to the instance using a dedicated server connection.
  - C. Henry is connected to the instance using a multi-threaded server connection.
  - D. You issued the command `ALTER SYSTEM SET RESOURCE_LIMIT=TRUE` while connected to the instance.
  - E. The `INIT.ORA` file includes a `RESOURCE_LIMIT=FALSE` entry.
8. You configure a `COMPOSITE_LIMIT` of 100 for a profile and assign the profile to WalterR. You modify the `INIT.ORA` file to include the line `RESOURCE_LIMIT=TRUE`. You configure resource costs using the following command:

```
ALTER RESOURCE COST
CONNECT_TIME 5
CPU_PER_SESSION 2
LOGICAL_READS_PER_SESSION 1;
```

You connect to the instance as WalterR to test your configuration by executing a SQL statement that hogs the CPU and notice that the process completes after 20 minutes. Why is the composite limit not being enforced? (Choose the best answer.)

- A. You are allowed to use 50 minutes of CPU time before for each session (100 minutes / 2).
  - B. The command only used a small fraction of the CPU time during the 20 minutes it took to execute the statement.
  - C. You need to restart the instance.
  - D. Oracle logged you in as INTERNAL instead of WalterR because you are a DBA.
  - E. The limit was enforced but you were not notified.
9. If you create a profile and specify limits for only some of the profile settings, what value is automatically assigned to any password management of resource limit you do not include in your CREATE PROFILE statement? (Choose the best answer.)
- A. DEFAULT
  - B. 0
  - C. UNLIMITED
  - D. 9999
  - E. You must specify a value for all profile limits.
10. What command can you use to unlock John's user account before the value of PASSWORD\_LOCK\_TIME for the profile assigned to John (MYPROFILE) is reached? (Choose the best answer.)
- A. ALTER PROFILE MYPROFILE UNLOCK JOHN
  - B. ALTER USER JOHN UNLOCK
  - C. ALTER USER JOHN ACCOUNT UNLOCK
  - D. ALTER USER JOHN UNLOCK ACCOUNT
  - E. You cannot unlock the account before PASSWORD\_LOCK\_TIME is reached.

## Scenario

1. As the DBA for MajorPharma, you have been asked to design a set of security policies for your databases that adhere to corporate standards. You currently do not make use of profiles in any of the four Oracle databases that you are responsible for. The requirements that need to be satisfied, and that all users in the company must adhere to, are the following:
  - Passwords must be changed every 45 days. Users must change their passwords the first time they logon after the password has expired.
  - Passwords must contain at least one alphabetic and one other character and must be a minimum of 5 characters long and a maximum of 12 characters long.
  - Users who do not perform any activity on the database for a period of 10 minutes need to specify their username and password again in order to gain access to the database.
  - Once a user has changed his or her password, he or she cannot use the previous password for ten password changes.
  - Users have three chances to enter their password. If they fail to successfully authenticate after those three chances, they need to contact an administrator to gain access to the database again.

You need to ensure that all four of your databases have the same settings. What is the easiest way to apply the same changes to all four databases?

Outline how you will create or modify profiles with the settings necessary to incorporate these requirements. Determine what other changes to the instances may also be required. Write out the SQL commands that will be needed to satisfy these requirements.

## Lab Exercises

### Lab 16-1 Creating and Testing Profiles

1. Using Server Manager line mode or SQL\*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a profile called USER\_PROFILE with the following settings:
  - Passwords expire after 30 days. The user will have two days to change the password.
  - No more than three failed login attempts will be allowed. If this limit is exceeded the account should be locked out until a DBA unlocks it.
  - The user cannot use more than five seconds of CPU time for each SQL statement submitted to the database.

- The user will be automatically disconnected if no database activity takes place for 20 minutes.

Enable the enforcement of resource limits in the database.

3. Assign the profile to the user STUDENT using the following command:

```
ALTER USER STUDENT PROFILE USER_PROFILE;
```

4. Attempt to connect to the CERTDB instance as user STUDENT four times, each time specifying an incorrect password until you receive an error message. After you receive the error message, attempt to connect to the CERTDB instance as user STUDENT with a password of ORACLE. What happens?
5. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER and query the DBA\_USERS view to determine if the user STUDENT is locked out.
6. Unlock the STUDENT user account by issuing the command:  

```
ALTER USER STUDENT ACCOUNT UNLOCK;
```
7. Connect to the CERTDB instance as user STUDENT with a password of ORACLE. Were you successful?

## Lab 16-2 Enabling Oracle's Default Password Management and Modifying Profiles

1. Using Server Manager line mode or SQL\*Plus Connect to the CERTDB instance as a user with SYSDBA privileges.
2. Review and execute the UTLPWDMG.SQL script in the ORACLE\_HOME/RDBMS/ADMIN directory.
3. While connected as user SYSTEM, attempt to change your password to SYSTEM using the following command:

```
ALTER USER SYSTEM IDENTIFIED BY SYSTEM;
```

What happens and why?

4. Connect to the CERTDB instance as user STUDENT with a password of ORACLE. Attempt to change your password to DATABASE. What happens and why?
5. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER. Modify the USER\_PROFILE profile so that no password verification takes place when a user changes his or her password.
6. Connect to the CERTDB instance as user STUDENT with a password of ORACLE. Attempt to change your password to DATABASE. What happens and why? Change your password back to ORACLE. Were you successful?

## Lab 16-3 Dropping Profiles

1. Using Server Manager line mode or SQL\*Plus Connect to the CERTDB instance as user SYSTEM with a password of MANAGER. Attempt to drop the USER\_PROFILE. What happens and why?
2. Drop the USER\_PROFILE and ensure that any user currently assigned the profile will get the DEFAULT profile.
3. Query the DBA\_PROFILES and DBA\_USERS view to verify your work.

# Answers to Chapter Questions

## Chapter Pre-Test

1. In order to ensure that the amount of CPU time a user takes up during a single SQL statement is tracked by Oracle against a resource limit you establish, you need to ensure that the RESOURCE\_LIMIT Oracle initialization parameter is set to TRUE. You can accomplish this by connecting to the instance as a DBA and issuing the command ALTER SYSTEM SET RESOURCE\_LIMIT=TRUE. If you want to ensure that the parameter is set that way each time the instance starts, you need to modify the INIT.ORA file.
2. It is possible to have more than one set of password verification rules in a database. You will need to connect as SYS and create several database password verification functions, each with a unique name. You can then create profiles with settings as required for each group of users and then assign the appropriate password verification function to the profile when including the PASSWORD\_VERIFY\_FUNCTION password management limit in the profile. Assigning each profile to the users who should have them can be accomplished with the CREATE USER or ALTER USER command. The next time the user makes a password change, the appropriate function for his or her profile will be called to verify if the new password satisfies the rules in the function.
3. To determine what profile has been assigned to JSmith and what limits are configured in the profile, the DBA can query the DBA\_USERS and DBA\_PROFILES view. Joining the two views in a single SQL statement answers both questions, as follows:

```
SQL> col username format a12
SQL> col profile format a12
SQL> col resource_name format a25
SQL> col limit format a15
SQL> SELECT USERNAME, U.PROFILE, RESOURCE_NAME, LIMIT
 2 FROM DBA_USERS U, DBA_PROFILES P
 3 WHERE U.PROFILE=P.PROFILE
 4* AND U.USERNAME='JSMITH';
```

```
SQL>
```

| USERNAME | PROFILE | RESOURCE_NAME             | LIMIT           |
|----------|---------|---------------------------|-----------------|
| JSMITH   | DEFAULT | COMPOSITE_LIMIT           | UNLIMITED       |
| JSMITH   | DEFAULT | FAILED_LOGIN_ATTEMPTS     | 3               |
| JSMITH   | DEFAULT | SESSIONS_PER_USER         | UNLIMITED       |
| JSMITH   | DEFAULT | PASSWORD_LIFE_TIME        | 60              |
| JSMITH   | DEFAULT | CPU_PER_SESSION           | UNLIMITED       |
| JSMITH   | DEFAULT | PASSWORD_REUSE_TIME       | 1800            |
| JSMITH   | DEFAULT | CPU_PER_CALL              | UNLIMITED       |
| JSMITH   | DEFAULT | PASSWORD_REUSE_MAX        | UNLIMITED       |
| JSMITH   | DEFAULT | LOGICAL_READS_PER_SESSION | UNLIMITED       |
| JSMITH   | DEFAULT | PASSWORD_VERIFY_FUNCTION  | VERIFY_FUNCTION |
| JSMITH   | DEFAULT | LOGICAL_READS_PER_CALL    | UNLIMITED       |
| JSMITH   | DEFAULT | PASSWORD_LOCK_TIME        | .0006           |
| JSMITH   | DEFAULT | IDLE_TIME                 | UNLIMITED       |
| JSMITH   | DEFAULT | PASSWORD_GRACE_TIME       | 10              |
| JSMITH   | DEFAULT | CONNECT_TIME              | UNLIMITED       |
| JSMITH   | DEFAULT | PRIVATE_SGA               | UNLIMITED       |

```
16 rows selected.
```

```
SQL>
```

- Database administrators (that is, DBAs) and users who have been granted the `DROP PROFILE` system privilege can drop profiles.
- If you set the value of `PASSWORD_REUSE_MAX` to 5 in a profile, you must set the value of `PASSWORD_REUSE_TIME` to `UNLIMITED`. Failure to do so will cause Oracle to generate an error that you are specifying conflicting profile limits.
- You can create as many profiles as you wish in an Oracle database. It is not necessary to assign them to users right after you create them. You may do so at any time after the profile is created. If you do not assign a profile to a user, its limits will not be enforced and it will simply take up space in the data dictionary in the `SYSTEM` tablespace.
- If you specify a value for `COMPOSITE_LIMIT` in a profile, you also need to configure resource costs using the `ALTER RESOURCE COST` command. The default cost for all 4 resources that make up `COMPOSITE_LIMIT` is 0, which means that none will be included in the calculation for `COMPOSITE_LIMIT` until you configure resource costs.

You also need to ensure that the `RESOURCE_LIMIT` initialization parameter has been set to `TRUE` if you want to enforce `COMPOSITE_LIMIT`.

- An account that has been locked out because `FAILED_LOGIN_ATTEMPTS` has been exceeded will be unlocked when the value of `PASSWORD_LOCK_TIME` for the profile is exceeded, or when the DBA manually unlocks the account with the `ALTER USER ... ACCOUNT UNLOCK` command.

9. To create, alter or drop profiles, you can issue the CREATE PROFILE, ALTER PROFILE, and DROP PROFILE command, respectively, in SQL\*Plus, Server Manager line mode, or SQL\*Worksheet. You can also perform these actions using Oracle Enterprise Manager or DBA Studio.
10. The password verification function (PASSWORD\_VERIFY\_FUNCTION) parameter of a profile can only return one of three possible values—NULL, FALSE, or TRUE. Any value other than TRUE is considered a failure and the password is not changed.

## Assessment Questions

1. **C, D.** The CREATE PROFILE command specifies conflicting values for PASSWORD\_REUSE\_MAX and PASSWORD\_REUSE\_TIME. If one of these values is specified, the other must be set to UNLIMITED. Oracle will return an error indicating there is a conflict with these two lines. The actual error message will indicate the error is on line 1 with an explanation that you have conflicting values for the above-mentioned parameters, as follows:

```
SQL> CREATE PROFILE MYPROFILE LIMIT
 2 PASSWORD_LIFE_TIME 45
 3 PASSWORD_REUSE_MAX 5
 4 PASSWORD_REUSE_TIME 2
 5 FAILED_LOGIN_ATTEMPTS 3
 6 PASSWORD_LOCK_TIME 1/1440
 7* PRIVATE_SGA 512K
SQL> /
CREATE PROFILE MYPROFILE LIMIT
*
ERROR at line 1:
ORA-28006: conflicting values for parameters
PASSWORD_REUSE_TIME and
PASSWORD_REUSE_MAX

SQL>
```

2. **A, E.** The account will be locked out the fifth time a user specified an invalid password because the value of FAILED\_LOGIN\_ATTEMPTS is five. It will be locked out for one day, or until a DBA unlocks it manually, since PASSWORD\_LOCK\_TIME is specified in days.
3. **E.** To assign MYPROFILE to an existing user in the database called Henry, you would issue the command ALTER USER HENRY PROFILE MYPROFILE.
4. **D.** The grace period specified by PASSWORD\_GRACE\_TIME begins after the first successful login following the expiry of the password. This means that the first time the user successfully connects to the instance after PASSWORD\_LIFE\_TIME for the user is triggered, the counter for PASSWORD\_GRACE\_TIME starts. It will give the user until the end of the grace period to change the password.

5. **A.** If the user does not change his or her password within the period specified by `PASSWORD_GRACE_TIME`, the account will be locked out and will need to be manually unlocked by a DBA.
6. **E.** The most likely reason that the `IDLE_TIME` resource limit is not being enforced is that the `RESOURCE_LIMIT` initialization parameter is not in the `INIT.ORA` file or is not set to `TRUE`. Henry cannot disable the application of resource limits with an `ALTER SESSION` command.
7. **B, E.** The most likely reason that the `PRIVATE_SGA` resource limit is not being enforced is that `RESOURCE_LIMIT=TRUE` is not configured for the instance, or that Henry is connected to the instance using a dedicated server connection. `PRIVATE_SGA` is only enforced for multi-threaded server (MTS) connections and only when `RESOURCE_LIMIT` is set to `TRUE` for the instance.
8. **C.** The most likely reason that the composite limit is not being enforced in this situation is that, even though the `INIT.ORA` value for `RESOURCE_LIMIT` is `TRUE`, the currently active setting is `FALSE`. This is because any changes made to the `INIT.ORA` file are not automatically applied to the running instance — the instance needs to be restarted for the resource limits to be enforced.
9. **A.** If you do not specify values for all profile settings, any settings not included in the `CREATE PROFILE` will be assigned a value of `DEFAULT`. This means that these resource limits will inherit the values of the `DEFAULT` profile. Any changes made to the same resource limit in the `DEFAULT` profile will also apply to the profile you created.
10. **C.** To manually unlock John's user account before the value of `PASSWORD_LOCK_TIME` is reached, you can issue the command:

```
ALTER USER JOHN ACCOUNT UNLOCK;
```

## Scenarios

1. The easiest way to apply the same changes to all databases is to create a SQL script using any text editor (such as Notepad or VI) to modify the `DEFAULT` profile. Because the limits need to be enforced for all users, the `DEFAULT` profile is the most likely choice as its settings apply to all users, unless they have been assigned another profile. Since you are currently not making use of profiles, the `DEFAULT` profile will work well.

In order to satisfy the requirements provided to you, you would modify the `DEFAULT` profile with these settings:

```
ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 45
PASSWORD_GRACE_TIME 0
PASSWORD_VERIFY_FUNCTION CORP_PASS_FUNC
IDLE_TIME 10
PASSWORD_REUSE_TIME 450
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME UNLIMITED;
```

To ensure that the corporate password requirements are met, you would create a function in the SYS schema called CORP\_PASS\_FUNC (before actually modifying the DEFAULT profile) whose code would be similar to the following:

```
CREATE OR REPLACE FUNCTION CORP_PASS_FUNC
(username varchar2,
 password varchar2,
 old_password varchar2)
RETURN boolean IS
 m integer;
 isnonchar boolean;
 ischar boolean;
 nonchararray varchar2(40);
 chararray varchar2(52);

BEGIN
 nonchararray:= '0123456789!"#$%&()``*+,-/::<=>?_';
 chararray:= 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

 -- Check for the minimum length of the password
 IF length(password) < 5 THEN
 raise_application_error(-20002, 'Password length less than 5');
 END IF;

 -- Check for the maximum length of the password
 IF length(password) > 12 THEN
 raise_application_error(-20002, 'Password length more than 12');
 END IF;

 -- Check if the password contains at least one letter and
 -- one non-alphabetic character -- punctuation mark.
 -- 1. Check for the non-alphabetic
 isnonchar:=FALSE;
 m := length(password);
 FOR i IN 1..length(nonchararray) LOOP
 FOR j IN 1..m LOOP
 IF substr(password,j,1) = substr(nonchararray,i,1) THEN
 isnonchar:=TRUE;
 GOTO findchar;
 END IF;
 END LOOP;
 END LOOP;
 IF isnonchar = FALSE THEN
 raise_application_error(-20003, 'Password should contain at least \
 one non-alphabetic character');
 END IF;

 -- 2. Check for the character
 <<findchar>>
 ischar:=FALSE;
 FOR i IN 1..length(chararray) LOOP
 FOR j IN 1..m LOOP
 IF substr(password,j,1) = substr(chararray,i,1) THEN
 ischar:=TRUE;

```

```

 END IF;
 END LOOP;
END LOOP;
IF ischar = FALSE THEN
 raise_application_error(-20003, 'Password should contain at least one \
 alphabetic character');
END IF;

-- Everything is fine; return TRUE ;
RETURN(TRUE);
END;
/

```

In order for the `IDLE_TIME` resource limit to be enforced, you will need to modify the `INIT.ORA` file in each database to include the parameter `RESOURCE_LIMIT=TRUE`. To ensure that the limits are enforced without restarting each instance, you will also need to issue the following command while connected to each instance as a DBA:

```
ALTER SYSTEM SET RESOURCE_LIMIT=TRUE;
```

## Lab Exercises

### Lab 16-1

1. Using Server Manager line mode or SQL\*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a profile called USER\_PROFILE with the following settings:
  - Passwords expire after 30 days. The user has two days to change the password.
  - No more than three failed login attempts will be allowed. If this limit is exceeded the account should be locked out until a DBA unlocks it.
  - The user cannot use more than five seconds of CPU time for each SQL statement submitted to the database.
  - The user will be automatically disconnected if no database activity takes place for 20 minutes.

```

SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> CREATE PROFILE USER_PROFILE LIMIT
 2 PASSWORD_LIFE_TIME 30
 3 PASSWORD_GRACE_TIME 2
 4 FAILED_LOGIN_ATTEMPTS 3
 5 PASSWORD_LOCK_TIME UNLIMITED
 6 CPU_PER_CALL 500
 7 IDLE_TIME 20;

```

Profile created.

```
SQL>
```

Enable the enforcement of resource limits in the database.

```
SQL> ALTER SYSTEM SET RESOURCE_LIMIT=TRUE;
```

```
System altered.
```

```
SQL>
```

- 3. Assign the profile to the user STUDENT using the following command:**

```
ALTER USER STUDENT PROFILE USER_PROFILE;
```

- 4. Attempt to connect to the CERTDB instance as user STUDENT four times, each time specifying an incorrect password until you receive an error message. After you receive the error message, attempt to connect to the CERTDB instance as user STUDENT with a password of ORACLE. What happens?**

```
SQL> connect student/ora1@certdb.delphi.bradsys.com
ERROR:
ORA-01017: invalid username/password; logon denied
```

```
Warning: You are no longer connected to ORACLE.
```

```
SQL> connect student/ora2@certdb.delphi.bradsys.com
ERROR:
ORA-01017: invalid username/password; logon denied
```

```
SQL> connect student/ora3@certdb.delphi.bradsys.com
ERROR:
ORA-01017: invalid username/password; logon denied
```

```
SQL> connect student/ora4@certdb.delphi.bradsys.com
ERROR:
ORA-28000: the account is locked
```

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
ERROR:
ORA-28000: the account is locked
```

```
SQL>
```

- 5. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER and query the DBA\_USERS view to determine if the user STUDENT is locked out.**

```
SQL> connect system/manager@certdb.delphi.bradsys.com;
Connected.
SQL> col username format a10
SQL> col profile format a15
SQL> col account_status format a15
```

```
SQL> col lock_date format a10
SQL> SELECT USERNAME, PROFILE, ACCOUNT_STATUS, LOCK_DATE
 2 FROM DBA_USERS
 3 WHERE USERNAME='STUDENT';
```

| USERNAME | PROFILE      | ACCOUNT_STATUS | LOCK_DATE |
|----------|--------------|----------------|-----------|
| STUDENT  | USER_PROFILE | LOCKED(TIMED)  | 27-JUN-01 |

```
SQL>
```

**6. Unlock the STUDENT user account by issuing the command:**

```
ALTER USER STUDENT ACCOUNT UNLOCK;
```

**7. Connect to the CERTDB instance as user STUDENT with a password of ORACLE. Were you successful?**

```
SQL> connect student/oracle@certdb.delphi.bradsys.com;
Connected.
SQL>
```

## Lab 16-2

**1. Using Server Manager line mode or SQL\*Plus Connect to the CERTDB instance as a user with SYSDBA privileges.**

```
SQL> connect internal/oracle@certdb.delphi.bradsys.com;
Connected.
SQL>
```

**2. Review and execute the UTLPWDMG.SQL script in the ORACLE\_HOME/RDBMS/ADMIN directory.**

```
SQL> @%ORACLE_HOME%\rdbms\admin\utlpwdmg.sql;
```

```
Function created.
```

```
Profile altered.
```

```
SQL>
```

**3. While connected as user SYSTEM, attempt to change your password to SYSTEM using the following command:**

```
ALTER USER SYSTEM IDENTIFIED BY SYSTEM;
```

**What happens and why?**

```
SQL> ALTER USER SYSTEM IDENTIFIED BY SYSTEM;
ALTER USER SYSTEM IDENTIFIED BY SYSTEM
*
```

```
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
```

```
ORA-20001: Password same as user
```

```
SQL>
```

- 4. Connect to the CERTDB instance as user STUDENT with a password of ORACLE. Attempt to change your password to DATABASE. What happens and why?**

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
```

```
SQL> ALTER USER STUDENT IDENTIFIED BY DATABASE;
```

```
ALTER USER STUDENT IDENTIFIED BY DATABASE
```

```
*
```

```
ERROR at line 1:
```

```
ORA-28003: password verification for the specified password
failed
```

```
SQL>
```

Even though the user STUDENT is using the USER\_PROFILE and not the DEFAULT profile, the value of the PASSWORD\_VERIFY\_FUNCTION for the USER\_PROFILE is DEFAULT. As such, any setting for the verification function of the DEFAULT profile is inherited by the USER\_PROFILE assigned to STUDENT. For this reason, the password change fails as being too simplistic.

- 5. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER. Modify the USER\_PROFILE profile so that no password verification will take place when a user changes his or her password.**

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
```

```
SQL> ALTER PROFILE USER_PROFILE LIMIT
```

```
2 PASSWORD_VERIFY_FUNCTION NULL;
```

```
Profile altered.
```

```
SQL>
```

- 6. Connect to the CERTDB instance as user STUDENT with a password of ORACLE. Attempt to change your password to DATABASE. What happens and why? Change your password back to ORACLE. Were you successful?**

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
```

```
SQL> ALTER USER STUDENT IDENTIFIED BY DATABASE;
```

```
User altered.
```

```
SQL> ALTER USER STUDENT IDENTIFIED BY ORACLE;
```

```
User altered.
```

```
SQL>
```

Both password changes were successful because the USER\_PROFILE assigned to the user STUDENT no longer has a password verification function.

### Lab 16-3

1. Using Server Manager line mode or SQL\*Plus Connect to the CERTDB instance as user SYSTEM with a password of MANAGER. Attempt to drop the USER\_PROFILE. What happens and why?

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> DROP PROFILE USER_PROFILE;
DROP PROFILE USER_PROFILE
*
ERROR at line 1:
ORA-02382: profile USER_PROFILE has users assigned, cannot drop without CASCADE

SQL>
```

2. Drop the USER\_PROFILE and ensure that any user currently assigned the profile will get the DEFAULT profile.

```
SQL> DROP PROFILE USER_PROFILE CASCADE;

Profile dropped.

SQL>
```

3. Query the DBA\_PROFILES and DBA\_USERS view to verify your work.

```
SQL> SELECT USERNAME, PROFILE FROM DBA_USERS;
```

| USERNAME | PROFILE |
|----------|---------|
| SYS      | DEFAULT |
| SYSTEM   | DEFAULT |
| OUTLN    | DEFAULT |
| DBSNMP   | DEFAULT |
| STUDENT  | DEFAULT |

```
SQL>
```



# Managing Users

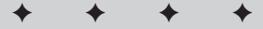
---

## EXAM OBJECTIVES

- ◆ Managing Users
  - Create new database users
  - Alter and drop existing database users
  - Monitor information about existing users

# 17

CHAPTER



## CHAPTER PRE-TEST

1. What user authentication mechanisms are supported by Oracle?
2. What prefix does Oracle recommend you use for operating system authentication and why?
3. How can you change a user's name while retaining all existing privileges granted to the user?
4. How would you ensure that a user had to change his or her password the first time a connection to the database is made?
5. What happens if you set a quota to 0 on a tablespace where objects owned by the user already exist?
6. How should you create a tablespace that you will specify as the TEMPORARY TABLESPACE for several users?
7. If you specify that Bob's DEFAULT TABLESPACE is USER\_DATA, will Bob be allowed to create a table on USER\_DATA? Why or why not?
8. How can you drop a user and all objects that the user owns?
9. How can you change a user's authentication mechanism from database to operating system?
10. When querying DBA\_TS\_QUOTAS, you notice that some users have a value of -1 in the MAX\_BYTES column. What does that mean?
11. What is a schema?

If your company is like the majority of those using Oracle, chances are that your database needs to be accessed by many users at the same time. Furthermore, over time you will need to add new users to the database or remove existing ones as people are brought on or leave the organization. Some users may also need to be given permissions to create objects on tablespaces or have their quotas increased or decreased so that they may be able to perform their duties. Understanding how to create, modify, and remove users from the database is key to the security responsibilities of a database administrator.

## Overview of User Management

In the previous chapter you learned how you could create profiles to implement password management and resource limits in your database. As you may recall from that discussion, profiles are assigned to users and are then enforced for those users to whom they have been assigned. The next logical step in understanding security in an Oracle database has to deal with creating users and the options available to do so.

When you create an Oracle database using the `CREATE DATABASE` command, the `SQL.BSQ` script that is run automatically creates two users — `SYS` and `SYSTEM`. As you run `CATALOG.SQL` and other scripts, other users will also be created. Some of these include the `OUTLN` user, the `DBSNMP` user used by Oracle Intelligent Agent to connect to the instance, and others. If you elected to create a starter database when you installed Oracle8i, you may also have other users for the sample schemas provided by Oracle, including `SCOTT` and others.

Any Oracle database, except those used for simple testing, also needs the DBA to create additional users who will create schema objects or access objects created by other users. It is the responsibility of the DBA to ensure that user accounts are created for those individuals who need access to a particular database, while those who should not have access are prevented from gaining it.

## Users and schemas

One of the terms that you have probably seen throughout this book is *schema*. A schema is a collection of all objects that a user has created. Schemas may contain objects that are segments (that is, occupy physical disk space in a tablespace) such as a table, index, cluster, or objects that a user creates and are stored in the data dictionary such as a sequence, stored procedure, or function.

When you create a user account in an Oracle database, Oracle also allows for the possibility that the user may own objects. When you grant privileges for the user to create an object and the user exercises the privilege, the first object created by the

user also creates the user's schema. Because a schema is so closely tied to a user, the term *schema* is often considered interchangeable with the user.



The assignment of privileges that allow a user to create objects will be covered in Chapter 18.

The objects that may be created in an Oracle8i database and become part of the user's schema, as well as a short description of each, are outlined in Table 17-1.

**Table 17-1**  
**Database Objects Available in Oracle8i**

| <b>Object</b>     | <b>Description</b>                                                                                                                                                                                                                                                                                               |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Table             | A collection of columns and rows representing a single entity (for example, students, courses, instructors, and so on).                                                                                                                                                                                          |
| Constraints       | Database objects that are used to enforce simple business rules and database integrity. Examples of constraints are primary key, foreign key, NOT NULL, CHECK and others.                                                                                                                                        |
| Views             | Views are a logical projection of data from one or more tables as represented by a SQL statement stored in the database. Views are used to simplify complex and repetitive SQL statements by assigning those statements a name in the database.                                                                  |
| Indexes           | Indexes are database objects that help speed up retrieval of data by storing logical pointers to specific key values. By scanning the index, which is organized in either ascending or descending order according to the key value, you are able to retrieve a row quicker than by scanning all rows in a table. |
| Sequences         | Sequences allow you to create and increment a counter that can be used to generate numerical values to be used as primary key values for a table.                                                                                                                                                                |
| Synonyms          | Like in the English language, a synonym is another name for an existing object. They are used in Oracle as shorthand for objects with long names, or to make it easier to remember a specific object.                                                                                                            |
| Stored Procedures | Stored procedures are a collection of SQL and PL/SQL statements that perform a specific task such as insert a row into a table, update data, and so on.                                                                                                                                                          |
| Triggers          | Triggers are a special kind of stored procedure that cannot be invoked manually but rather are automatically invoked whenever an action is performed on a table. Triggers are always associated with a table and a corresponding action such as INSERT, UPDATE, or DELETE.                                       |

| <i>Object</i>          | <i>Description</i>                                                                                                                                                                                                                                                                                     |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Functions              | Functions are stored programs that must return a value. Unlike stored procedures which can have parameters passed to them and do not need to return any value as output, a function must return a value.                                                                                               |
| Packages               | Packages are a collection of stored procedures and functions grouped under a common name, enabling you to group all program elements logically for a particular part of the database under a single name for maintenance and performance reasons.                                                      |
| User-Defined Datatypes | User-defined datatypes are database objects that can be used in any table or another object definition. Using user-defined datatypes allows you to ensure consistency between tables and also lets you apply methods (that is, actions that can be performed by the object) as part of the definition. |

## Guidelines for Creating Users

Although as a DBA you can create as many or as few users in your database as you like, chances are that you will want to follow a set of rules that would ensure that users are created for those individuals who need to access the database, and that the initial settings of the user accounts match the users' needs.

### Authentication mechanisms

One of the things that you need to consider when creating a user is what authentication mechanism will be used to verify the user's password and whether or not the user should access the database. Oracle supports three authentication mechanisms:

- ♦ **Database Authentication** — The most common authentication mechanism is database authentication. When using this method, the DBA creates a user account in the database for each user who needs access. When creating the user, the DBA also specifies a password, although the user may change the password later if needed. When the user attempts to connect to the instance, Oracle verifies the username and password provided at connect time with those stored in the data dictionary. If a match is found and the user has appropriate permissions, the user is allowed to connect to the instance; if not, the connection is denied and an error message returned to the user.
- ♦ **Operating Systems Authentication** — When using operating system authentication, the user account must still be created in the Oracle database, but the management of the password for the account can be handled outside of Oracle. All other restrictions that may be imposed on a user account that is

authenticated by the database can still be applied to the user account authenticated by the operating system (such as quotas, and the granting of privileges), but the DBA does not need to deal with password management.

Furthermore, any profile that is assigned to a user authenticated by the operating system will have the bulk of its password management settings ignored since Oracle no longer manages the user's password. Operating system authentication also brings other benefits and drawbacks, which is covered later in this chapter.

- ♦ **Network Authentication**— Many organizations today make use of directory services and external network authentication mechanisms. Some of these methods include RADIUS servers, Novell Directory Services (NDS), or Active Directory in a Windows 2000 environment. Oracle8i even includes its own directory, called Oracle Internet Directory (OID), that can be used to implement network authentication. The advantage of network authentication is a centralized user repository. However, network authentication is complex to implement and is beyond the scope of this book or the “Oracle8i: Architecture and Administration” exam and is not covered here for those reasons.

An important thing to remember about selecting an authentication mechanism for a user is that you are not stuck with your first choice. If you initially selected operating system authentication, you can always modify the user to use database authentication at a later date, or vice versa.

When selecting an authentication mechanism for a user in your database, you need to balance your security requirements with the administrative overhead of one method versus the other. Using database authentication may require more work but will ensure that only those user accounts that have been explicitly created in the database using the password stored in the data dictionary will be allowed to connect and get access to the data. This prevents someone who happens to know a user's network password from accessing a database, which could occur if you use operating system or network authentication.

## Quotas

As mentioned previously, any user you create in an Oracle database can potentially create objects. One of the ways that you control which users are able to create objects is by assigning privileges to users to perform that task. However, in order for a user to create those objects that are also considered segments and occupy disk space, you need to configure a quota for the user on the tablespaces where you want him or her to create those segments.

Oracle, by default, does not allow a user to create segments and take up space on tablespaces in the database. This is by design and is there to ensure that the DBA is able to allocate disk space as needed. As most DBAs really need to be control freaks and ensure that space allocation is something that they alone are responsible for, a good practice is to have all segments created by DBAs only. In this way the DBA has ultimate control over all storage considerations for the database. While this may be

good in theory, it may not always work in practice so a proper understanding of quotas is necessary.

When you create a user account in the database, you may specify a quota for the user on any tablespace in the database. You can also increase or decrease the user's quota at a later time by using the ALTER USER command. If a quota is specified for a user on a tablespace, the amount of disk space allocated in the quota is the total amount of disk space that all of the user's objects (segments) that are created on that tablespace can occupy. If the user exceeds his or her quota, the objects cannot grow any larger until the quota is reset. Of course, the user must also be granted privileges to create objects (segments) in the first place. If the user is not granted these privileges, he or she cannot make use of the quota.

For example, if you create a user called JohnS and grant him a quota of 10MB on the USER\_DATA tablespace, JohnS will be able to create tables, indexes, and other segments on the USER\_DATA tablespace (assuming he has been granted privileges to do so). If JohnS creates several tables and indexes on the USER\_DATA tablespace, Oracle tracks the total amount of disk space that all of JohnS's objects consume on the USER\_DATA tablespace. If JohnS grants others privileges to insert, or update data in his tables, the data that these users add to the tables is applied against JohnS's quota on the tablespace. This is because quotas are based on the owner of the object and not the user who actually adds data to the object. If the sum total of all tables, indexes, and other segments that JohnS owns on the USER\_DATA tablespace exceeds his quota of 10MB, Oracle will not allow any inserts or updates to take place on these objects until the quota is increased by the DBA or some segments are dropped to decrease the space utilization.

A single user can have a quota on several tablespaces at the same time. In practice this is a good thing to do because it enables users to create tables on one tablespace and indexes on another, as discussed in earlier chapters. You can specify an actual quota limit, such as 10MB or 1GB, for a quota for a user on a tablespace, or you can specify the keyword UNLIMITED when configuring a user's quota on a tablespace. Specifying UNLIMITED allows the user's objects to consume all available disk space on a tablespaces, and, if auto-growth has been enabled on datafiles that make up the tablespace, may cause the hard disk to be filled as well — which may cause other problems. It is generally not recommended that you specify a quota of UNLIMITED for any user on any tablespace.

It should be noted that quotas need to be specified only on those tablespaces where a user will create permanent objects, such as tables, indexes, and so on. You do not need to specify a quota on a temporary tablespace where the user may create global temporary tables or use those tablespaces for sorts that spill to disk. Oracle handles the allocation of space for any temporary objects itself.

Under no circumstances should you configure a quota on the SYSTEM tablespace for any user except SYS and SYSTEM, or those granted the DBA role. These users are the ones who will manipulate the data dictionary and should be the only ones allowed to create segments on the SYSTEM tablespace. The SYSTEM tablespace

should not contain any user-defined segments. Always allocate quotas to users on those tablespaces where you want them to create objects — and SYSTEM should not be one of them.

## Tablespace assignment

When you create a new user, you need to specify a DEFAULT TABLESPACE and a TEMPORARY TABLESPACE for the user. The values you specify for both parameters need to be tablespaces that already exist in the database.

The DEFAULT TABLESPACE is used to store objects that the user creates and for which no tablespace was explicitly specified. The tablespace configured as the user's DEFAULT TABLESPACE must have been created to store permanent objects using the CREATE TABLESPACE command (and not the CREATE TEMPORARY TABLESPACE command). Oracle ensures that the object will still be created by placing it on the DEFAULT TABLESPACE configured for the user. If the user does not have a quota on the tablespace configured as the user's DEFAULT TABLESPACE, the object creation will fail with an error. This means that the allocation of space on the DEFAULT TABLESPACE is no different from other tablespaces — quotas still need to be specified.

The user's TEMPORARY TABLESPACE is used to create temporary objects, such as global temporary table's data and sorts that spill to disk. You should specify a TEMPORARY TABLESPACE for the user that has been created to hold temporary objects using the CREATE TEMPORARY TABLESPACE command. Although Oracle allows you to configure a user to make use of a tablespace that holds permanent objects as the user's TEMPORARY TABLESPACE, this causes performance problems and is not recommended. In fact, the capability to use a regular tablespace as the user's TEMPORARY TABLESPACE is only provided for backward compatibility with previous versions and should be avoided.

## Other user creation guidelines

When you create a user to be authenticated by the database, you need to specify a password. You may choose to use a standard password, or even the username as the password. You can freely choose any password that will be allowed by the password verification function, but make sure that you communicate the password to the user so that he or she can log on to the instance.

A good idea when creating a user in an Oracle database is to also expire the password at the time of creation. This is done with the PASSWORD EXPIRE parameter in the CREATE USER statement. Expiring the password forces the user to change the password the first time a connection to the database is made. The user can then choose a new password that he or she may find easier to remember and also adheres to the password verification function of the profile you assigned to the user at creation time.

One other option of the CREATE USER command that you may also want to consider using is ACCOUNT LOCK. In some situations it may be beneficial to create user accounts in advance of actually needing them. An example of this includes when new employees join the company and you want to set things up before their arrival, or a series of part-time employees are brought on board who will be using generic usernames. In those cases, if you create the user accounts before they are needed, and specify ACCOUNT LOCK at the time of creation, no one will be able to use the accounts until you manually unlock them using the ALTER USER ... ACCOUNT UNLOCK command. In this way you can perform all other configuration tasks for the user accounts that you need to, and then turn them loose when things are ready and the individuals using those accounts are ready.

A final point deals with education. It has been said many times that if you see someone doing something that you do not think is right, it may not be because he or she does not have the mental faculties to perform the task well—it just may be that that user does not know any better. Users need to be shown how to connect to the instance and properly enter their username and password in the tool that they will be using. They also need to be shown how to change the password of their user account using the same tool or the ALTER USER statement. Education goes a long way toward reducing the amount of simple problem solving that you as a DBA may need to do in this area.

## Managing Users

Managing users is the simple process of creating, altering, or dropping users. These tasks can be accomplished off the command line using the CREATE USER, ALTER USER, and DROP USER command, respectively, or by using Oracle Enterprise Manager or DBA Studio.

### Creating users

#### Objective

Create new database users

The easiest way to create a user is to connect to the instance as a DBA using Server Manager line mode, SQL\*Plus or SQL\*Worksheet, and issue the CREATE USER command. The syntax of the command is as follows:

```
CREATE USER username
 IDENTIFIED [BY password | EXTERNALLY | GLOBALLY AS extname]
 [DEFAULT TABLESPACE tablespacename]
 [TEMPORARY TABLESPACE tablespacename]
 [ACCOUNT LOCK | UNLOCK]
 [PROFILE profilename | DEFAULT]
 [PASSWORD EXPIRE]
 [QUOTA num [K|M] | UNLIMITED ON tablespace]
 [QUOTA num [K|M] | UNLIMITED ON tablespace] ...]
```

The meaning of each of the parameters of the CREATE USER command is outlined in Table 17-2.

Table 17-2  
CREATE USER Parameter Descriptions

| <i>Parameter</i> | <i>Description</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Username         | The name of the user to be created. The name must be specified and must be unique within the database. Oracle automatically converts the name to uppercase. This parameter can be up to 30 characters long and must adhere to Oracle naming conventions (starts with a letter, may contain numbers, letters, or the symbols #, _, or \$).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| IDENTIFIED       | <p>Specifies how the user will have their authenticity validated. The three methods include:</p> <p><b>BY password</b>, where <b>password</b> represents a DBA-specified password of up to 30 characters. The database checks the credentials provided by the user against the username and passwords in the data dictionary and denies or grants access based upon whether or not a match is found.</p> <p><b>EXTERNALLY</b>, where the username is authenticated by the operating system of the computer on which the database is running and Oracle allows access if the operating system authenticates the user. In order to configure this type of authentication, some additional processes are involved.</p> <p><b>GLOBALLY AS extname</b>, where the username and password are passed to the <b>extname</b> service for logon validation. This type of authentication requires external authentication mechanisms, such as a RADIUS server.</p> |

| <i>Parameter</i>      | <i>Description</i>                                                                                                                                                                                                                                                                                                                                               |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DEFAULT TABLESPACE    | The name of a tablespace that is used to store segments created by the user if no tablespace is specified at creation time. The value for DEFAULT TABLESPACE defaults to SYSTEM if this parameter is not specified. Always specify a DEFAULT TABLESPACE for the user and assign the user a quota on this tablespace if you expect him or her to create segments. |
| TEMPORARY TABLESPACE  | The name of a tablespace that is used to store temporary segments, such as sort segments. The value for TEMPORARY TABLESPACE defaults to SYSTEM if this parameter is not specified. Always specify a TEMPORARY TABLESPACE for the user.                                                                                                                          |
| ACCOUNT LOCK   UNLOCK | Allows you to explicitly lock or unlock a user account at creation time. The default for this parameter is UNLOCK, which means that a user can connect to the instance as soon as the account is created and the appropriate privileges granted.                                                                                                                 |
| PROFILE               | Specifies the profile that will be assigned to the user. The profile determines password management and, optionally, resource limits that apply to the user. If this parameter is not specified, or the keyword DEFAULT is used, the user is assigned the DEFAULT profile.                                                                                       |
| PASSWORD EXPIRE       | This parameter enables you to automatically expire the user's password at creation time, forcing him or her to change the password the first time a successful connection to the instance is established. The default behavior is not to expire the user's password, though use of this parameter is recommended.                                                |
| QUOTA                 | Lets you configure a quota for the user on tablespaces in the database. The quota is specified in bytes, kilobytes (K), or megabytes (M). You should specify a quota on the user's DEFAULT TABLESPACE if you expect the user to create segments. You should not specify an UNLIMITED quota on a tablespace for a regular user.                                   |

An example of creating a user is shown in the following code:

```
SQL> CREATE USER JohnS
 2 IDENTIFIED BY Pass10ra$
 3 DEFAULT TABLESPACE USERS
 4 TEMPORARY TABLESPACE TEMP
 5 QUOTA 20M ON USERS
 6 QUOTA 20M ON INDX
 7 PASSWORD EXPIRE
 8* PROFILE DEVELOPER_PROFILE
SQL> /
```

User created.

```
SQL>
```

Once you create the user using the `CREATE USER` command, even if you did not specify `ACCOUNT LOCK`, the user will not be able to connect to the instance as a privilege to do so must first be granted. Any attempt to connect to the instance will generate an error (though the user will be prompted to change their password if `PASSWORD EXPIRE` was specified), as follows:

```
SQL> connect johns/Pass10ra$@certdb.delphi.bradsys.com;
Changing password for johns
New password: *****
Retype new password: *****
ERROR:
ORA-01045: user JOHNS lacks CREATE SESSION privilege; logon
denied
```

Password unchanged  
SQL>

If you use Oracle Enterprise Manager to create a user, OEM automatically grants the user the `CREATE SESSION` privilege.



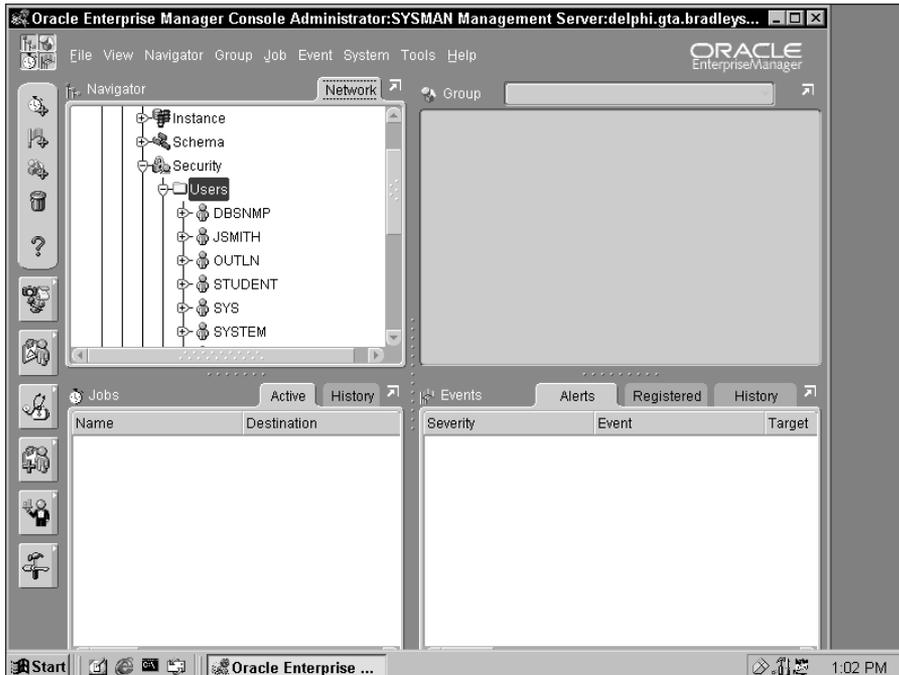
System privileges and the `CREATE SESSION` privilege will be discussed in Chapter 18.

## Creating users using Oracle Enterprise Manager

You can also create users with Oracle Enterprise Manager using the following steps:

### STEP BY STEP: Creating a User Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to create the user.
2. Expand the database and then Security, and then Users, as shown in Figure 17-1.



**Figure 17-1:** Expanding the Security node of the database where you want to create the user will display the Users node.

3. Right-click on Users and select “Create,” as shown in Figure 17-2, to create a new user.



**Figure 17-2:** Right-click Users and select “Create” to create a new user.

4. In the dialog box that comes up, enter a name for the user, authentication mechanism, profile, password expire option, account lock or unlock, and default and temporary tablespace information on the General tab, as shown in Figure 17-3.

A screenshot of the 'Create User' dialog box. The title bar reads 'Create User - SYSTEM@CERTDB.bradsys.com'. The 'General' tab is selected. Fields include: Name: DAVID; Profile: DEVELOPER\_PROFILE; Authentication: Password; Enter Password: [masked]; Confirm Password: [masked];  Expire Password Now; Tablespaces: Default: USERS; Temporary: TEMP; Status:  Locked,  Unlocked. Buttons at the bottom: Create, Cancel, Show SQL, Help.

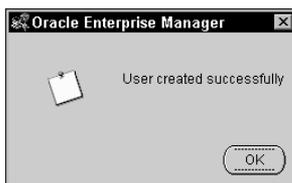
**Figure 17-3:** Enter a name for the user and other key information on the General tab.

5. To enter quota limits for tablespace, select the Quota tab and enter appropriate values, as shown in Figure 17-4.



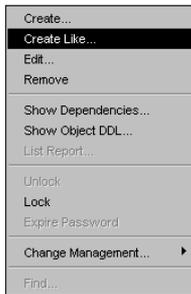
**Figure 17-4:** Click the Quota tab to enter quota limits for the user.

6. When you have made all your selections, click on “Create” to create the user. If there are no errors, you will be shown a dialog box indicating success, as presented in Figure 17-5.



**Figure 17-5:** You have successfully created a new user.

7. If you want to create another user with the same quota settings and permissions as an existing user in the database, you can right-click on the user whose settings you want to duplicate and select “Create Like...” as shown in Figure 17-6.



**Figure 17-6:** You can use OEM to create users with similar settings to those already in the database.

## Configuring operating system authentication

When you create a user in your database and specify the IDENTIFIED EXTERNALLY clause in the CREATE USER command, the user is authenticated by the operating system when an attempt to connect to the Oracle instance is made. Operating system authentication makes the most sense when the user connects to the Oracle instance on the same machine that is running Oracle and where the database resides. It provides a way to have administrators that manage both the server and Oracle database to only have to login to the server and then have automatic access to the database.

When connecting to an Oracle instance using operation system authentication, the user does not have to specify a username and password. The only requirements are that the environment variable ORACLE\_SID be set to the SID of the instance that you want to connect to and that the user has been created in the database using the IDENTIFIED EXTERNALLY clause. If these are true, the user can connect to the instance when invoking a tool, such as SQL\*Plus as follows:

```
C:\> sqlplus /
```

Using this method instructs Oracle to use the individual’s operating system username and check it against a list of users in the database that have been created using the IDENTIFIED EXTERNALLY clause and see if a match exists. If one is found, the connection is allowed; if not, the user will be shown an error and not be allowed to connect to the instance.

### Parameters for operating system authentication

In order to provide some additional level of security and to prevent anyone with an operating system user account from attempting to gain access to the database, Oracle uses a couple of Oracle initialization (INIT.ORA) parameters that determine how the username is formulated in the database when using operating system authentication and where the authentication request is sent by Oracle. These are OS\_AUTHENT\_PREFIX and REMOTE\_OS\_AUTHENT, and their meaning is as follows:

- ♦ **OS\_AUTHENT\_PREFIX**— This parameter, whose default value is OPS\$, specifies what is added to the user's operating system user ID to map it to an Oracle username. When specifying this parameter, you need to add its value to the username you create in Oracle. This enables Oracle to map the individual's operating system user ID to an Oracle username and determine if access should be granted to the database.



Tip

If you use the Oracle Database Assistant to create a database in Oracle8i, the INIT.ORA file created by the Assistant explicitly sets the value of OS\_AUTHENT\_PREFIX to "" (blank). If you want to use the default value of OPS\$ instead, you need to comment out or remove the OS\_AUTHENT\_PREFIX line in the INIT.ORA file and shutdown and restart your instance. This parameter cannot be changed using the ALTER SYSTEM command.

Table 17-3 shows some examples of using the prefix value, and corresponding operating system user ID and Oracle username that will be required. As is evident from reviewing Table 17-3, the parameter specifies exactly what its name infers—the set of characters that will be added to the username in Oracle when creating a user in order to map it to the appropriate operating system user ID.

Table 17-3  
**OS\_AUTHENT\_PREFIX Mappings**

| <i>OS_AUTHENT_PREFIX</i> | <i>O/S User ID</i> | <i>Oracle Username</i> |
|--------------------------|--------------------|------------------------|
| OS\$                     | Bob                | OS\$Bob                |
| OPS\$ (the default)      | Bob                | OPS\$Bob               |
| "" (Blank)               | Bob                | Bob                    |
| NTUSER_                  | Bob                | NTUSER_Bob             |
| Unix\$                   | Bob                | Unix\$Bob              |

One thing to consider is that if you leave the `OS_AUTHENT_PREFIX` parameter at its default value of `OPS$`, you will be able to create the user account to be identified by a password, which can be used to connect to the instance when not logged in to the same computer that is running the instance, and use operating system authentication when you are on the same computer. For example, if you create an Oracle username for the operating system user “Bob” using the following command:

```
CREATE USER OPS$Bob IDENTIFIED BY password
```

and the value of `OS_AUTHENT_PREFIX` is `OPS$`, if Bob has logged in to the computer where the instance resides, he can issue the following command to be connected to the instance:

```
sqlplus /
```

In other words, when logging on locally, Bob will be able to use operating system authentication. If Bob needs to connect to the instance from another computer, he can issue the following command within `SQL*Plus`:

```
connect OPS$Bob/password@myinstance
```

where “myinstance” is the name of the instance Bob wants to connect to.

An important point to remember is that if you change the value of `OS_AUTHENT_PREFIX` in the `INIT.ORA` file after you have already created users with the prefix you used previously, any existing user needs to be dropped and re-created. The `OS_AUTHENT_PREFIX` does not dynamically change the names of users in the database when it is modified. The net effect of this is that users may no longer be able to connect to the instance if `OS_AUTHENT_PREFIX` is changed in the Oracle initialization file.

If you need to change the `OS_AUTHENT_PREFIX` value, make use of Oracle Enterprise Manager and the “Create Like ...” option outlined previously to create duplicate accounts for users with the new prefix. Using this method retains all of the original user’s quotas and privileges and then enables you to delete the old user, assuming the user does not own any objects that need to be re-created. If the user owns objects, you can use the Export and Import utilities to move the objects to another user’s schema using the `FROMUSER` and `TOUSER` options of the Import command.

- ♦ **REMOTE\_OS\_AUTHENT**— This parameter, which is set to `FALSE` by default, enables you to configure Oracle to request authentication by a remote server of any user created with the `IDENTIFIED EXTERNALLY` clause. Normally, Oracle asks the local computer on which the instance is running to authenticate a user created with the `IDENTIFIED EXTERNALLY` clause. In certain environments—such as when operating in Windows NT/2000 domains—the computer where Oracle is running may not be the one where the user’s network account resides. In order for these users to be properly authenticated, the local computer may need to contact another machine (for example, a domain controller in Windows NT/2000 domains) to authenticate the user.

Oracle does not recommend that you include the line `REMOTE_OS_AUTHENT=TRUE` in your `INIT.ORA` file as it may leave the database more vulnerable than if the local machine contained all the user accounts for access to the database. If you are operating in an environment where user accounts are resident on a machine other than the Oracle database server, you may want to enable it, or, better still, use database authentication for all users and thereby keep closer control over who has access to your databases.

## Modifying users

### Objective

Alter and drop existing database users

Having to modify users is a relatively common practice in those Oracle environments that have implemented password management. This is because a DBA needs to periodically change the user's password or unlock a user account that was locked because the number of `FAILED_LOGIN_ATTEMPTS` was exceeded, or a password was not changed within `PASSWORD_GRACE_TIME`.

Reasons for modifying users can be several and depend on the type of database and its configuration. Some of the more common reasons include changing the user's `DEFAULT TABLESPACE` or `TEMPORARY TABLESPACE` values, increasing, decreasing, or adding a new quota on a tablespace for the user, changing a password (for which using the `PASSWORD EXPIRE` clause is also recommended since this forces the user to pick a new password the first time a connection to the database is made).

To modify user accounts you can make use of the `ALTER USER` command while connected to the instance as a DBA using `SQL*Plus` or `Server Manager` line mode. You can change the majority of the user's attributes, including the authentication method and password. However, you cannot change the username—you need to create a new user and drop the old one in order to do that. The syntax of the `ALTER USER` command is as follows:

```
ALTER USER username
 IDENTIFIED [BY password | EXTERNALLY | GLOBALLY AS extname]
 [DEFAULT TABLESPACE tablespacename]
 [TEMPORARY TABLESPACE tablespacename]
 [ACCOUNT LOCK | UNLOCK]
 [PROFILE profilename]
 [PASSWORD EXPIRE]
 [QUOTA num [K|M] | UNLIMITED ON tablespace
 [QUOTA num [K|M] | UNLIMITED ON tablespace] ...]
```

For example, if you wanted to change JohnS's default tablespace to `CERTDB` and assign him a quota on that tablespace, you would issue the following command:

```
SQL> ALTER USER JOHNS
2 DEFAULT TABLESPACE CERTDB
3 QUOTA 20M ON CERTDB;
```

```
User altered.
```

```
SQL>
```

If you also wanted to remove JohnS's quota on the USERS tablespace, you can issue the following command:

```
SQL> ALTER USER JOHNS
2 QUOTA 0 ON USERS;
```

```
User altered.
```

```
SQL>
```

Setting a quota to 0 on a tablespace for a user allows any segments that are currently on the tablespace and owned by the user to remain; however, these segments will not be allowed to grow — that is, no new extents can be allocated to them. In other words, Oracle allows the user to keep any data on the tablespace, since it may be needed by others, but prevents any further use of disk space by the user's objects.

When you issue the ALTER USER command, any parameters that you have not specified remain intact. This means that all quotas, and other settings are not modified unless you explicitly do so with the ALTER USER command.

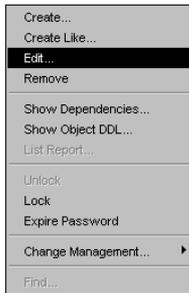
The ALTER USER command can be used by any user that is authenticated by the database to change his or her password. Oracle does not restrict who can issue the ALTER USER command for this purpose, although only the DBA, by default, can change anyone's password, where regular users can only change their own password.

## Modifying users Using Oracle Enterprise Manager

You can also modify users with Oracle Enterprise Manager using the following steps:

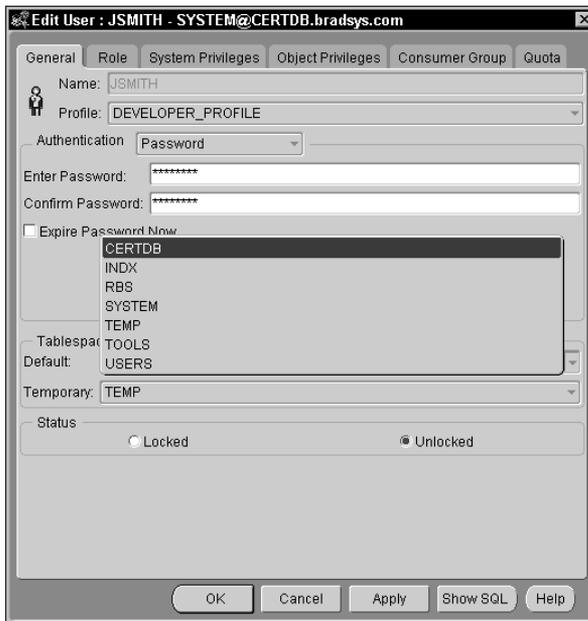
### STEP BY STEP: Modifying a User Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database where the user to be modified exists.
2. Expand the database, Security, and Users, and then select the user whose settings you want to modify.
3. Right-click on the username of the user and then select "Edit," as shown in Figure 17-7.



**Figure 17-7:** Right-click on the user you want to modify and select “Edit.”

4. In the dialog box that comes up make necessary changes selecting the appropriate tabs as needed, such as changing the user’s default tablespace as shown in Figure 17-8.



**Figure 17-8:** Make necessary changes in the Edit User dialog box by selecting the appropriate tabs.

5. When you have made all your changes, click on “Apply” to save your changes, or “Cancel” to abort the process. To save your changes and exit you can also click “OK.”

## Dropping users

**Objective**

Alter and drop existing database users

If a user should no longer have access to the database, you can issue the `DROP USER` command to remove the user from the database. The syntax of the `DROP USER` command is as follows:

```
DROP USER username [CASCADE]
```

Oracle prevents you from dropping a user from the database whose schema contains objects. This is to ensure that objects created by one user and depended upon by other users or their objects are not inadvertently removed from the database. Since the user and the schema are linked, dropping to user also drops the schema. Oracle does not allow you to drop both the user and schema, unless you specify the `CASCADE` option on the `DROP USER` command.

Using the `CASCADE` option drops all objects, as well as any data contained in tables, that the user owns (that is, are in the user's schema). This can have drastic side effects in the database if not planned properly. It is always recommended that before dropping a user, you determine if the user owns any objects in the database, and, if so, drop the objects after verifying that the objects are not depended upon by other users.

To find out which objects are owned by a user you wish to drop, you can query the `DBA_OBJECTS` view. If you get zero rows returned, as in the following example, you can safely drop the user:

```
SQL> SELECT OBJECT_NAME, OBJECT_TYPE FROM DBA_OBJECTS
2 WHERE OWNER='JOHNS';
```

```
no rows selected
```

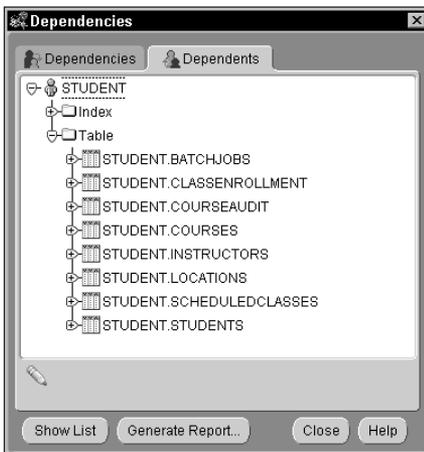
```
SQL>
```

On the other hand, if you do get a list of objects back, you should also want to query the `DBA_DEPENDENCIES` view to determine which objects are dependent on those owned by the user you wish to drop. If you prefer, you can also use Oracle Enterprise Manager to check objects dependencies.

When using Oracle Enterprise Manager to check which objects depend upon objects in the schema of the user you are about to drop, expand the database, and then Security, and then Users. Right-click on the user you want to remove and select "Show Dependencies," as presented in Figure 17-9. In the Dependencies dialog box that comes up, click the Dependents tab to get a list of objects on the user's schema, sorted by type, as shown in Figure 17-10. Expand the type of object and the object name itself to determine other depended objects and their owners. Continue this process until you have a list of objects that may be affected by dropping the user. You will then need to break the dependencies manually, or create the objects in other schemas and then re-establish the dependencies.



**Figure 17-9:** To display dependencies for a user, right-click on the user in question and select Show Dependencies.



**Figure 17-10:** Click the Dependents tab in the Dependencies dialog box to determine which objects are in the user's schema and which other objects depend on them.

Once you have resolved any dependencies, you can drop the user without the CASCADE option, as shown here:

```
SQL> DROP USER JOHNS;
```

```
User dropped.
```

```
SQL>
```

The DROP USER command is immediate and there is no backing out of the operation—that is, rollback is not possible. For this reason, you should perform a full backup of the database before dropping users.

## Dropping users using Oracle Enterprise Manager

You can also drop users with Oracle Enterprise Manager using the following steps:

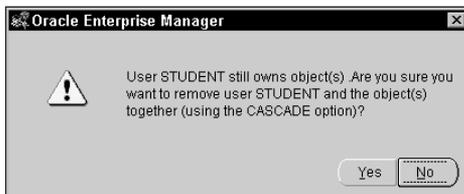
### STEP BY STEP: Dropping a User Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to drop the user.
2. Expand the database and then Security, and then Users, and then select the user that you want to drop.
3. Right-click on the username of the user and then select “Remove,” as shown in Figure 17-11.

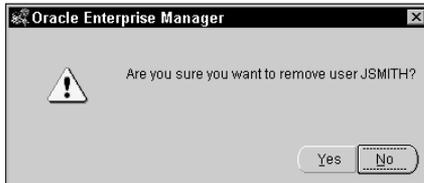


**Figure 17-11:** Right-click on the user you want to drop and select “Remove”.

4. If the user owns objects, a dialog box stating so will appear, as shown in Figure 17-12. If the user does not own objects you will be presented with a dialog box similar to Figure 17-13. Make the appropriate choice by selecting Yes or No.



**Figure 17-12:** If the user owns objects, you can drop both the user and the schema at the same time. Select Yes to do so, or No to cancel the operation.



**Figure 17-13:** If the user does not own objects, you are prompted to verify that you want to drop the user.

## Getting Information about Users

### Objective

Monitor information about existing users

Oracle provides a few data dictionary views that can be used to get information on users in the database. The first of these is `DBA_USERS` whose column's structure is as shown in the following described command:

```
SQL> DESC DBA_USERS;
Name Null? Type

USERNAME NOT NULL VARCHAR2(30)
USER_ID NOT NULL NUMBER
PASSWORD NOT NULL VARCHAR2(30)
ACCOUNT_STATUS NOT NULL VARCHAR2(32)
LOCK_DATE DATE
EXPIRY_DATE DATE
DEFAULT_TABLESPACE NOT NULL VARCHAR2(30)
TEMPORARY_TABLESPACE NOT NULL VARCHAR2(30)
CREATED NOT NULL DATE
PROFILE NOT NULL VARCHAR2(30)
INITIAL_RSRC_CONSUMER_GROUP VARCHAR2(30)
EXTERNAL_NAME VARCHAR2(4000)

SQL>
```

The majority of the columns in the view are self-explanatory, although a few do merit further explanation. The `USER_ID` column is one that is automatically populated by Oracle when the user is created. It is the internal identifier for the user that Oracle attaches to objects that the user creates and privileges that the user has been granted. The `PASSWORD` column is the user's password in an internally encrypted format. This means that not even the DBA knows what a user's password

is, although the DBA could always change the password. The column `INITIAL_RSRC_CONSUMER_GROUP` deals with consumer groups, which is a method available to a DBA to limit the CPU usage and parallel query utilization by a given set of users, but is not covered in the “Oracle8i: Architecture and Administration” exam. `EXTERNAL_NAME` deals with network authentication, which was discussed earlier.

Some of the uses of the data dictionary views include the ability to get a list of users in the database, the account status, and when their passwords expire, you could issue the following command:

```
SQL> col account_status format a15
SQL> col username format a15
SQL> SELECT USERNAME, ACCOUNT_STATUS, EXPIRY_DATE
2 FROM DBA_USERS;
```

| USERNAME | ACCOUNT_STATUS | EXPIRY_DA |
|----------|----------------|-----------|
| SYS      | OPEN           | 26-JUL-01 |
| SYSTEM   | OPEN           | 26-JUL-01 |
| OUTLN    | OPEN           | 26-JUL-01 |
| DBSNMP   | OPEN           | 06-AUG-01 |
| STUDENT  | OPEN           | 26-AUG-01 |
| JOHNS    | OPEN           | 13-AUG-01 |

6 rows selected.

```
SQL>
```

The `DBA_TS_QUOTAS` view can be used to get a list of tablespace quotas for a particular user or group of users, as in the following example:

```
SQL> col tablespace_name format a15
SQL> SELECT USERNAME, TABLESPACE_NAME, BLOCKS, BYTES, MAX_BLOCKS, MAX_BYTES
2 FROM DBA_TS_QUOTAS
3 WHERE USERNAME IN ('STUDENT','JOHNS')
4* ORDER BY USERNAME
SQL> /
```

| USERNAME | TABLESPACE_NAME | BLOCKS | BYTES  | MAX_BLOCKS | MAX_BYTES |
|----------|-----------------|--------|--------|------------|-----------|
| JOHNS    | USERS           | 0      | 0      | 2560       | 20971520  |
| JOHNS    | INDX            | 0      | 0      | 1920       | 15728640  |
| JOHNS    | CERTDB          | 0      | 0      | 2560       | 20971520  |
| STUDENT  | CERTDB          | 80     | 655360 | -1         | -1        |

```
SQL>
```

The `BLOCKS` and `BYTES` columns of the `DBA_TS_QUOTAS` view display the amount of space, in database blocks and bytes, are currently allocated to the user in each

tablespace where the user has been given a quota. The `MAX_BLOCKS` and `MAX_BYTES` columns display the maximum amount of disk space that the user is allowed in each tablespace (that is, the user's quota). A value of `-1` for either `MAX_BYTES` or `MAX_BLOCKS` indicates that the user has an `UNLIMITED` quota on the tablespace, as is the case with user `STUDENT` on the `CERTDB` tablespace.



Other data dictionary views are available to display user's privileges and roles assigned. These are covered in chapters 18 and 19.

## Key Point Summary

In preparing for the “Oracle8i DBA: Architecture and Administration” exam, please keep these points regarding managing users in mind:

- ♦ A schema is a collection of all objects that a user has created and, therefore, owns.
- ♦ Each individual requiring access to the database must provide a username and password that will be authenticated and determine whether or not access will be permitted.
- ♦ Oracle supports three user authentication mechanisms: database authentication, operating system authentication, and network (or global) authentication.
- ♦ Database authentication requires that you specify a password for the user at creation time, or later when using the `ALTER USER` command.
- ♦ In order to make use of operating system authentication, you need to properly configure the `OS_AUTHENT_PREFIX` Oracle initialization parameter and properly create user accounts with the prefix. If you change the prefix, you need to re-create user accounts with the new prefix.
- ♦ The default for `OS_AUTHENT_PREFIX` is `OPS$` and allows users to have a password specified and use database authentication when connecting from a remote computer, and also use operating system authentication when connecting from the machine on which the database resides.
- ♦ The `CREATE USER` command or Oracle Enterprise Manager can be used to create a new database user. If you use the `CREATE USER` command, you also need to grant the user the `CREATE SESSION` privilege so that the user can connect to the instance. This is done automatically when creating a user with Oracle Enterprise Manager.
- ♦ The `ALTER USER` command can be used by a database-authenticated user to change his or her password. The DBA can use the `ALTER USER` command to

make changes to any of the user's parameter, with the exception of the user name. In order to change a user's name, you need to create a new user with the desired name and drop the old user. Oracle Enterprise Manager's "Create Like ..." menu option is recommended if you want to preserve privileges and quotas.

- ♦ The DROP USER command can be used to drop a user. If the user to be dropped owns objects (that is, has a schema), you can use the CASCADE option of the DROP USER command to drop the user and the schema.
- ♦ The DBA\_USERS view can be used to get information on a user account's status, password expiry date, default and temporary tablespace, and lockout information.
- ♦ The DBA\_TS\_QUOTAS view provides information on quotas assigned to users, and current space usage by segments that the user has created.



# STUDY GUIDE

---

The assessment questions, scenarios and labs for this chapter will help you to become more familiar with creating, modifying and querying information about users. Work through the labs and test your knowledge of users in Oracle8i by answering the assessment questions and scenarios.

## Chapter Assessment

1. You have an operating system user called BobW. You want to create a user account for BobW in an Oracle database. You are running with the default setting for the OS\_AUTHENT\_PREFIX parameter. What will be the username you create in Oracle? (Choose the best answer.)
  - A. OS\_BOBW
  - B. OPS\$BOBW
  - C. OS\$BOBW
  - D. BOBW
  - E. BOBW\$OPS
2. If you do not specify a TEMPORARY TABLESPACE when creating a new user account, what will be the value of this parameter when the user is created? (Choose the best answer.)
  - A. SYSTEM
  - B. TEMP
  - C. NULL
  - D. "" (empty string)
  - E. You must specify a value for TEMPORARY TABLESPACE.
3. Which of the following commands can a new user called BobW issue after successfully connecting to the instance and establishing a user session? (Choose all correct answers.)
  - A. ALTER USER BobW PASSWORD EXPIRE;
  - B. ALTER USER BobW QUOTA 2M ON SYSTEM;
  - C. ALTER USER BobW ACCOUNT LOCK;
  - D. ALTER USER BobW TEMPORARY TABLESPACE TEMP;
  - E. ALTER USER BobW IDENTIFIED BY NEWPASS;

4. If you wanted to get a list of users with their quotas in bytes, and password expiry dates, which SQL statement would you issue? (Choose the best answer.)
- A. SELECT USERNAME, TABLESPACE\_NAME, QUOTA\_BYTES, EXPIRY\_DATE FROM DBA\_TS\_QUOTAS;
  - B. SELECT U.USERNAME, TABLESPACE\_NAME, MAX\_BYTES, EXPIRY\_DATE FROM DBA\_TS\_QUOTAS T, DBA\_USERS U WHERE U.USERNAME=T.USERNAME
  - C. SELECT U.USERNAME, TABLESPACE\_NAME, QUOTA\_BYTES, EXPIRY\_DATE FROM DBA\_TS\_QUOTAS T, DBA\_USERS U WHERE U.USERNAME=T.USERNAME
  - D. SELECT USERNAME, TABLESPACE\_NAME, QUOTA\_BYTES, EXPIRY\_DATE FROM DBA\_USERS;
  - E. SELECT USERNAME, TABLESPACE\_NAME, MAX\_BYTES, EXPIRY\_DATE FROM DBA\_USERS;
  - F. SELECT USERNAME, TABLESPACE\_NAME, MAX\_BYTES, EXPIRY\_DATE FROM DBA\_TS\_QUOTAS;
5. You want to prevent BobS from creating new objects on the USERS tablespace. Which command must you issue? (Choose the best answer.)
- A. ALTER TABLESPACE USERS QUOTA 0 FOR BobS;
  - B. ALTER USER BobS NOQUOTA ON USERS;
  - C. ALTER TABLESPACE USER NOQUOTA FOR BobS;
  - D. ALTER USER BobS QUOTA 0 ON TABLESPACE USERS;
  - E. ALTER USER BobS QUOTA 0 ON USERS;

6. JohnS attempts to create a table on the USERS tablespace and receives an error that there is insufficient space. When querying DBA\_TS\_QUOTAS, you receive the following results:

```
SQL> SELECT TABLESPACE_NAME, BYTES, MAX_BYTES FROM
 2 DBA_TS_QUOTAS WHERE USERNAME='JOHNS'
```

| TABLESPACE_NAME | BYTES   | MAX_BYTES |
|-----------------|---------|-----------|
| USERS           | 6553600 | -1        |

What is the most likely reason that JohnS is not able to create the table? (Choose the best answer.)

- A. JohnS does not have permissions to create the table.
  - B. The tablespace has no more free disk space.
  - C. The user's quota has been exceeded for JohnS.
  - D. JohnS is attempting to create a temporary table on a permanent tablespace.
  - E. The database is corrupt.
7. What affect does setting REMOTE\_OS\_AUTHENT to TRUE have on how user's are authenticated when attempting to connect to the instance? (Choose the best answer.)
- A. Database authenticated users can now be authenticated by the operating system.
  - B. A remote server can authenticate a user IDENTIFIED EXTERNALLY.
  - C. You can use a RADIUS server to authenticate database users.
  - D. Users can be automatically authenticated when connecting to other Oracle instances without needing a user account in the database.
  - E. The parameter has no affect and is only provided for backward compatibility.
8. If your database has modified the DEFAULT profile as per the contents of the Oracle-provided *utlpwdmg.sql* script, which line of the following CREATE USER statement will cause the operation to fail? (Choose the best answer.)
- ```
1 CREATE USER JamieL
2 IDENTIFIED EXTERNALLY
3 PASSWORD EXPIRE
4 ACCOUNT LOCK;
```
- A. 1
 - B. 2
 - C. 3
 - D. 4
 - E. The command will succeed.

9. If your database has modified the DEFAULT profile as per the contents of the Oracle-provided *utlpwdmg.sql* script, which line of the following CREATE USER statement will cause the operation to fail? (Choose the best answer.)

```
1 CREATE USER JamieL
2 IDENTIFIED BY oralpass_
3 PASSWORD EXPIRE
4 ACCOUNT LOCK;
```

- A. 1
 - B. 2
 - C. 3
 - D. 4
 - E. The command will succeed.
10. What privilege is granted automatically by Oracle Enterprise Manager and not by the CREATE USER command when you create a new user account? (Choose the best answer.)
- A. ALTER USER
 - B. ALLOW CONNECT
 - C. CREATE SESSION
 - D. CREATE CONNECTION

Scenarios

1. You are the DBA of a small company that is using Oracle to provide data to a Web-based application running on the same computer as the Oracle database. The database will need to be accessed by 35 users, of which three (BobW, MarkS, and MaryB) will need to create objects. The rest of the objects in the database are owned by the user SYSTEM and will only need to be accessed for query and update activity by the remaining users.

As you are responsible for both the server and the database, you want to ensure that you provide access to the Web site and the database in the most efficient way possible. Each of the 35 users will need to connect to the Web site with a distinct username and be authenticated by the server operating system. These users will also need to have access to the database and should not be required to re-enter their authentication information.

BobW, MarkS, and MaryB will also need to access the database through the Web site or from a client-side application resident on their computer. The user account you create for them in the database should support this functionality.

Furthermore, BobW, MarkS, and MaryB will also need to create objects on the USERS, INDX, DATA01, DATA02, and DATA03 tablespaces. The database also contains two temporary tablespaces (TEMP1 and TEMP2) as well as the SYSTEM tablespace.

BobW and MaryB should not be restricted in the amount of disk space their objects use on DATA01, DATA02, or DATA03 tablespaces, but should only be allowed to use a maximum of 100MB on each of the USERS and INDX tablespaces. MarkS should only be allowed to make use of 50MB on the USERS tablespace and 25MB on INDX.

What authentication mechanism should you use to support the outlined requirements? Identify any INIT.ORA parameters that are affected and what their settings will be.

What will be the value of the DEFAULT TABLESPACE and TEMPORARY TABLESPACE clauses for BobW, MaryB, and MarkS? The remaining 32 users?

What commands would you issue to ensure that quotas are properly configured for all users?

Do you need to do anything else to allow all users to connect to the instance?

Lab Exercises

Lab 17-1 Creating Users

1. Using Server Manager line mode or SQL*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a user called MarkS with a password of password. Ensure that MarkS's objects will be created on the CERTDB tablespace and that sorts will spill to disk on tablespace TEMP. Assign MarkS a quota of 10MB on the CERTDB tablespace. Make sure that MarkS changes his password the first time he connects to the instance.

Were you successful? Why not?

3. Modify the DEFAULT profile to remove the password verification function.
4. Try creating MarkS again. Were you successful this time?
5. Create a user called MaryB with a password of oracle. Ensure that MaryB's sorts will use the TEMP tablespace and that all of her objects will be created on the CERTDB tablespace.
6. Issue the following command to ensure that MarkS can connect to the instance:

```
GRANT CREATE SESSION TO MARKS;
```

7. Invoke another SQL*Plus session and connect to the CERTDB instance as MarkS. What happens?
8. Invoke another SQL*Plus session and attempt to connect to the CERTDB instance as user MaryB. What happens and why? Correct the problem and try again?
9. As MaryB, attempt to create a table using the following command:


```
CREATE TABLE T1 (C1 NUMBER(4));
```

Were you successful? Why or why not?
10. Issue the same CREATE TABLE command in step 9 while connected to the instance as MarkS. Were you successful? Why or why not?

Lab 17-2 Modifying and Querying Users

1. If you have not already done so, using Server Manager line mode or SQL*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER. Also invoke SQL*Plus two more times to connect to the CERTDB instance as MarkS and MaryB.
2. As user MarkS, issue the command to change your password to *database*. Were you successful? Why or why not? Exit SQL*Plus.
3. As user MaryB, issue the command to assign yourself a quota of 10MB on the CERTDB tablespace. Were you successful? Why or why not?
4. As SYSTEM, lock MarkS's user account, and grant MaryB a 2MB quota on the CERTDB tablespace.
5. Using SQL*Plus attempt to connect to the instance as MarkS. What happens?
6. As SYSTEM issue the following command to allow MaryB to create a table:


```
GRANT CREATE TABLE TO MaryB;
```
7. As MaryB issue the following create table command:


```
CREATE TABLE T1 (C1 NUMBER(4));
```

Were you successful? Why or why not?
8. As SYSTEM, query the data dictionary to determine when the passwords for users Student, MaryB, and MarkS will expire. Also, determine if any accounts are locked out. Unlock any accounts that are locked.
9. As SYSTEM, query the data dictionary to determine how much space is currently used by MarkS and MaryB on each tablespace where they have objects. Determine the number of blocks and bytes that may be used for these user's objects and the tablespaces where each user may create objects.

Lab 17-3 Dropping Users

1. If you have not already done so, using Server Manager line mode or SQL*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Drop MarkS's user account from the database. Were you successful? Why or why not? Make any changes necessary to ensure that the user account is dropped.
3. Drop MaryB's user account from the database. Were you successful? Why or why not? Make any changes necessary to ensure that the user account is dropped.
4. Query the data dictionary to ensure that the user accounts are removed.

Answers to Chapter Questions

Chapter Pre-Test

1. Oracle supports three user authentication mechanisms. Database authentication requires you to specify a password for the user, which will be checked by Oracle each time the user attempts to connect to the instance. Operating system authentication requires you to map an operating system user ID to an Oracle username. When the user attempts to connect to the instance, Oracle will check with the operating system to ensure that the user is valid and then allow the connection. Finally, Oracle also supports network, or global, authentication where the user is verified by a third-party such as a RADIUS server.
2. When making use of operating system authentication, Oracle recommends that you use the default OS_AUTHENT_PREFIX of OPS\$. This will allow you to create users that can use operating system authentication when connecting to the instance on the local machine, and also make use of a password when connecting remotely. This provides the most flexibility.
3. In order to change a user's name, you will need to create a new user account. This is because Oracle does not support a rename functionality for user accounts. The easiest way to create a new user with the exact same set of privileges as the old user account is to use Oracle Enterprise Manager's "Create Like ..." option when right-clicking on the user account you want to duplicate. If the user owns objects, you will need to use the Export and Import utilities to move the objects to the new user by using the FROMUSER and TOUSER options of the Import utility.
4. If you want to force the user to change their password the first time they connect to the instance, specify the PASSWORD EXPIRE clause when creating the user.

5. If you set a quota to 0 on a tablespace where the user already has created objects, those objects will be allowed to remain on the tablespace but will not be able to grow. Oracle freezes any space allocation made to the user on the affected tablespace.
6. Any tablespace that you will specify as the TEMPORARY TABLESPACE when you create or alter a user account should be created with the CREATE TEMPORARY TABLESPACE syntax. This will ensure that Oracle performs the most efficient allocation and de-allocation of space and that management of temporary objects does not unnecessarily slow down the database.
7. If you specify that Bob's DEFAULT TABLESPACE is USER_DATA, Bob will not be able to create objects on the tablespace unless you assign Bob a quota on the tablespace and the necessary privileges to create the objects, such as CREATE TABLE.
8. If you wanted to drop a user and all the objects that the user owns, you can issue the DROP USER command with the CASCADE option. This will drop all user objects, and then the user account. Doing this can cause problems if other users' objects depend on the one's dropped, so make sure that no negative side effects will result by issuing the command.
9. If you want to change a user's authentication mechanism from database to operating system, you can issue the following command:

```
ALTER USER username
IDENTIFIED EXTERNALLY;
```

Prior to performing this action, ensure that the username corresponds to the setting of the OS_AUTHENT_PREFIX initialization parameter and that the user account exists at the operating system level. If both of these are not true, the user will not be able to connect to the instance.

10. A value of -1 in the MAX_BYTES and MAX_BLOCKS columns of the DBA_TS_QUOTAS view indicates that the user has been assigned a quota of UNLIMITED on the tablespace.
11. A schema is the collection of all objects owned by a user. This includes segments — that is, objects requiring storage space such as tables and indexes — and other objects such as stored procedures and views.

Assessment Questions

1. **B.** The default OS_AUTHENT_PREFIX is OPS\$, so you would need to create an Oracle username of OPS\$BOBW to allow the operating system user BobW to gain access to the database.
2. **A.** The default value for both DEFAULT TABLESPACE and TEMPORARY TABLESPACE is SYSTEM. This is because Oracle can only be certain that the SYSTEM tablespace exists in the database. If you fail to specify a value for TEMPORARY TABLESPACE when creating a user, Oracle will set the value to SYSTEM.

3. **E.** The only ALTER USER command that a new user can issue is one to change his or her password. Users cannot lock their account, expire their password, assign quotas to themselves, or modify the DEFAULT TABLESPACE or TEMPORARY TABLESPACE values.
4. **B.** In order to get both the tablespace quotas and expiry dates, you would need to join the DBA_USERS and DBA_TS_QUOTAS data dictionary views on the USERNAME column. Of the two possible options (B or C) that performed a join between these views, only B had the right set of columns (MAX_BYTES, EXPIRY_DATE, TABLESPACE_NAME, and USERNAME).
5. **E.** Setting the quota on the USERS tablespace to 0 for BobS will prevent him from creating any new objects on the tablespace. Existing objects owned by BobS on the USERS tablespace will be allowed to remain there, but no new extents will be able to allocate to them.
6. **B.** The most likely reason that JohnS is not able to create a new object on the tablespace is that the tablespace has no more space available, as the error message appeared to indicate.
7. **B.** Setting REMOTE_US_AUTHENT to TRUE allows Oracle to have users that were configured for operating system authentication (IDENTIFIED EXTERNALLY) to be authenticated by a computer other than the one on which the instance is running.
8. **C.** If you create a user to be IDENTIFIED EXTERNALLY, you cannot expire the user's password. This is because Oracle is not managing the user's password but the operating system is. Oracle cannot dictate to the OS that the password should be expired, but is only able to ask the OS to authenticate the user.
9. **E.** The command will succeed and not generate an error. The *utlpwdmg.sql* script ensures that passwords contain at least two of three types of characters (letters, numbers, symbols) and the password specified (*ora1pass_*) satisfies this requirement. The other parameters specified in the CREATE USER statement are valid, so the command will succeed, assuming another user with the same name does not already exist in the database.
10. **C.** Oracle Enterprise Manager automatically grants a new user the CONNECT role, which includes the CREATE SESSION privilege. Since the CONNECT role was not listed, the only valid option is CREATE SESSION.



For more information on roles, refer to Chapter 19.

Scenarios

1. In order to satisfy the requirements of this scenario you will need to ensure that each user account is authenticated by the operating system. For this, you will need to create all 35 users to be IDENTIFIED EXTERNALLY. By doing so, you will ensure that users will only need to login to the computer when accessing the Web site (which appears to require that a user provide

credentials), and the Web site application can then use those same user credentials to connect to the instance. If you use operating system authentication, the user will not be prompted to enter a username and password for database access.

BobW, MaryB, and MarkS will also need to access the database from remote computers, so operating system authentication will not work in this case. However, if you ensure that the `OS_AUTHENT_PREFIX` is set to the default value of `OPS$`, you can allow these users to be authenticated by the operating system when accessing the database through the Web site, while also allowing them to specify a password when connecting remotely. In order to accomplish this, you would not create their user accounts to be `IDENTIFIED EXTERNALLY`, but rather specify a password for each of them that they can use to connect to the instance remotely. You will also need to educate them to use the `OPS$BobW`, `OPS$MaryB`, and `OPS$MarkS` user accounts when connecting to the instance remotely.

To allow BobW, MarkS, and MaryB to create objects on the tablespaces outlined, you will need to assign appropriate quotas and ensure that they have the proper privileges to create the objects. The quota assignment can be accomplished by issuing the following `ALTER USER` commands:

```
ALTER USER BOBW
    QUOTA UNLIMITED ON DATA01
    QUOTA UNLIMITED ON DATA02
    QUOTA UNLIMITED ON DATA03
    QUOTA 100M ON USERS
    QUOTA 100M ON INDX;
```

```
ALTER USER MARYB
    QUOTA UNLIMITED ON DATA01
    QUOTA UNLIMITED ON DATA02
    QUOTA UNLIMITED ON DATA03
    QUOTA 100M ON USERS
    QUOTA 100M ON INDX;
```

```
ALTER USER MARKS
    QUOTA 50M ON USERS
    QUOTA 25M ON INDX;
```

The `DEFAULT TABLESPACE` clause for all users should be set to `USERS`, because this is where the quota limits are properly set for those users (MaryB, MarkS, and BobW) that can create objects. As other users do not have any quotas, setting it to `USERS` reduces work in the future by not requiring you to change this value if someone else needs to create objects on that tablespace later and is assigned a quota.

The `TEMPORARY TABLESPACE` value for users should be either `TEMP1` or `TEMP2`. Users should be divided between these two tablespaces.

After creating user accounts, make sure that you grant all users the `CREATE SESSION` privilege so that they can connect to the instance.

Lab Exercises

Lab 17-1

1. Using Server Manager line mode or SQL*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a user called MarkS with a password of password. Ensure that MarkS's objects will be created on the CERTDB tablespace and that sorts will spill to disk on tablespace TEMP. Assign MarkS a quota of 10MB on the CERTDB tablespace. Make sure that MarkS changes his password the first time he connects to the instance.

Were you successful? Why not?

```
SQL> CREATE USER MarkS
  2  IDENTIFIED BY password
  3  DEFAULT TABLESPACE CERTDB
  4  TEMPORARY TABLESPACE TEMP
  5  QUOTA 10M ON CERTDB
  6  PASSWORD EXPIRE;
CREATE USER MarkS
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20002: Password too simple

SQL>
```

The user creation failed because the DEFAULT profile assigned to the user has a password verification function active. The password specified for MarkS does not satisfy the requirements of the password verification function.

3. Modify the DEFAULT profile to remove the password verification function.

```
SQL> ALTER PROFILE DEFAULT LIMIT
  2  PASSWORD_VERIFY_FUNCTION NULL;

Profile altered.

SQL>
```

4. Try creating MarkS again. Were you successful this time?

```
SQL> CREATE USER MarkS
  2  IDENTIFIED BY password
  3  DEFAULT TABLESPACE CERTDB
  4  TEMPORARY TABLESPACE TEMP
  5  QUOTA 10M ON CERTDB
  6  PASSWORD EXPIRE;
```

User created.

SQL>

Removing the password verification function allowed the user creation to succeed.

5. Create a user called MaryB with a password of oracle. Ensure that MaryB's sorts will use the TEMP tablespace and that all of her objects will be created on the CERTDB tablespace.

```
SQL> CREATE USER MaryB
      2 IDENTIFIED BY oracle
      3 TEMPORARY TABLESPACE TEMP
      4 DEFAULT TABLESPACE CERTDB;
```

User created.

SQL>

6. Issue the following command to ensure that MarkS can connect to the instance:

```
GRANT CREATE SESSION TO MARKS;
```

7. Invoke another SQL*Plus session and connect to the CERTDB instance as MarkS. What happens?

```
SQL> connect marks/password@certdb.delphi.bradsys.com
Changing password for marks
New password: *****
Retype new password: *****
Connected.
SQL>
```

MarkS is prompted to change his password because the user was created with the PASSWORD EXPIRE clause. You can enter any password, but until the password change is complete MarkS will not be able to connect to the instance.

8. Invoke another SQL*Plus session and attempt to connect to the CERTDB instance as user MaryB. What happens and why? Correct the problem and try again.

```
SQL> connect MaryB/oracle@certdb.delphi.bradsys.com
ERROR:
ORA-01045: user MARYB lacks CREATE SESSION privilege; logon
denied
```

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> GRANT CREATE SESSION TO MaryB
2 /

Grant succeeded.

SQL> connect MaryB/oracle@certdb.delphi.bradsys.com
Connected.
SQL>
```

- 9. As MaryB, attempt to create a table using the following command:**

```
CREATE TABLE T1 (C1 NUMBER(4));
```

Were you successful? Why or why not?

```
SQL> CREATE TABLE T1 (C1 NUMBER(4));
CREATE TABLE T1 (C1 NUMBER(4))
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

```
SQL>
```

MaryB does not have the privilege to create a table. Even if she did, the statement would fail because she has not been assigned a quota on the CERTDB tablespace, which is her DEFAULT TABLESPACE.

- 10. Issue the same CREATE TABLE command in step 9 while connected to the instance as MarkS. Were you successful? Why or why not?**

```
SQL> connect marks/oracle@certdb.delphi.bradsys.com
Connected.
SQL> CREATE TABLE T1 (C1 NUMBER(4));
CREATE TABLE T1 (C1 NUMBER(4))
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

```
SQL>
```

Even though MarkS has a quota on the CERTDB tablespace, the statement fails because he lacks the necessary privileges to perform the action.

Lab 17-2

1. If you have not already done so, using Server Manager line mode or SQL*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER. Also invoke SQL*Plus two more times to connect to the CERTDB instance as MarkS and MaryB.
2. As user MarkS, issue the command to change your password to *database*. Were you successful? Why or why not? Exit SQL*Plus.

```
SQL> connect Marks/oracle@certdb.delphi.bradsys.com
Connected.
SQL> ALTER USER MarkS
  2 IDENTIFIED BY database;

User altered.

SQL>
```

Any user may change his or her own password using the ALTER USER syntax.

3. As user MaryB, issue the command to assign yourself a quota of 10MB on the CERTDB tablespace. Were you successful? Why or why not?

```
SQL> connect MaryB/oracle@certdb.delphi.bradsys.com
Connected.
SQL> ALTER USER MaryB
  2 QUOTA 10M ON CERTDB;
ALTER USER MaryB
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

User cannot assign themselves quotas on tablespaces. In order for a user to have a quota, the DBA must assign the quota to the user.

4. As SYSTEM, lock MarkS's user account, and grant MaryB a 2MB quota on the CERTDB tablespace.

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> ALTER USER MarkS ACCOUNT LOCK;

User altered.

SQL> ALTER USER MaryB
  2 QUOTA 2M ON CERTDB;

User altered.

SQL>
```

5. Using SQL*Plus attempt to connect to the instance as MarkS. What happens?

```
SQL> connect MarkS/database@certdb.delphi.bradsys.com
ERROR:
ORA-28000: the account is locked
```

```
Warning: You are no longer connected to ORACLE.
SQL>
```

6. As SYSTEM issue the following command to allow MaryB to create a table:

```
GRANT CREATE TABLE TO MaryB;
```

7. As MaryB issue the following create table command:

```
CREATE TABLE T1 (C1 NUMBER(4));
```

Were you successful? Why or why not?

```
SQL> connect maryb/oracle@certdb.delphi.bradsys.com
Connected.
SQL> CREATE TABLE T1 (C1 NUMBER(4));
```

```
Table created.
```

```
SQL>
```

The CREATE TABLE statement succeeded because MaryB has been granted the CREATE TABLE privilege and has also been assigned a 2MB quota on the CERTDB tablespace, which is her DEFAULT TABLESPACE. Since the CREATE TABLE command did not specify a tablespace, the table was created on the CERTDB tablespace.

8. As SYSTEM, query the data dictionary to determine when the passwords for users Student, MaryB, and MarkS will expire. Also, determine if any accounts are locked out. Unlock any accounts that are locked.

```
SQL> col username format a15
SQL> col account_status format a15
SQL> SELECT USERNAME, ACCOUNT_STATUS, LOCK_DATE, EXPIRY_DATE
       2 FROM DBA_USERS;
```

USERNAME	ACCOUNT_STATUS	LOCK_DATE	EXPIRY_DA
SYS	OPEN		26-JUL-01
SYSTEM	OPEN		26-JUL-01
OUTLN	OPEN		26-JUL-01
DBSNMP	OPEN		06-AUG-01
MARKS	LOCKED	29-JUN-01	28-AUG-01
MARYB	OPEN		28-AUG-01
STUDENT	OPEN		26-AUG-01
JOHNS	OPEN		13-AUG-01

8 rows selected.

```
SQL> ALTER USER MARKS ACCOUNT UNLOCK;
```

User altered.

```
SQL>
```

- 9. As SYSTEM, query the data dictionary to determine how much space is currently used by MarkS and MaryB on each tablespace where they have objects. Determine the number of blocks and bytes that may be used for these user's objects and the tablespaces where each user may create objects.**

```
SQL> col tablespace_name format a15
```

```
SQL> col username format a15
```

```
SQL> SELECT USERNAME, TABLESPACE_NAME, BYTES, BLOCKS, MAX_BYTES, MAX_BLOCKS
 2 FROM DBA_TS_QUOTAS
 3 WHERE USERNAME IN ('MARKS', 'MARYB')
 4 ORDER BY USERNAME;
```

USERNAME	TABLESPACE_NAME	BYTES	BLOCKS	MAX_BYTES	MAX_BLOCKS
MARKS	CERTDB	0	0	10485760	1280
MARYB	CERTDB	40960	5	2097152	256

```
SQL>
```

Lab 17-3

1. If you have not already done so, using Server Manager line mode or SQL*Plus connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Drop MarkS's user account from the database. Were you successful? Why or why not? Make any changes necessary to ensure that the user account is dropped.

```
SQL> DROP USER MarkS;
```

User dropped.

```
SQL>
```

- 3. Drop MaryB's user account from the database. Were you successful? Why or why not? Make any changes necessary to ensure that the user account is dropped.**

```
SQL> DROP USER MaryB;
DROP USER MaryB
*
ERROR at line 1:
ORA-01922: CASCADE must be specified to drop 'MARYB'
```

```
SQL>
```

MaryB owns objects in the database and cannot be dropped using the regular DROP USER syntax. The CASCADE option must be specified.

```
SQL> DROP USER MaryB CASCADE;
```

```
User dropped.
```

```
SQL>
```

- 4. Query the data dictionary to ensure that the user accounts are removed.**

```
SQL> SELECT USERNAME FROM DBA_USERS
       2 WHERE USERNAME IN ('MARKS', 'MARYB');
```

```
no rows selected
```

```
SQL>
```

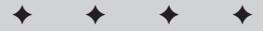

Managing Privileges

EXAM OBJECTIVES

- ◆ Managing Privileges
 - Identify system and object privileges
 - Grant and revoke privileges
 - Control operating system or password file authentication
 - Identify auditing capabilities

18

CHAPTER



CHAPTER PRE-TEST

1. Who can configure database auditing and what is needed to ensure that it is working properly?
2. If you grant a user the CREATE TABLE privilege, do you have to do anything else to allow the user to create a table in the database?
3. If you want a user to be able to create a package in any schema, what privilege must you grant?
4. Can the user SYSTEM grant SELECT privileges to a user on any table in the database? Why or why not?
5. If you grant BobW all privileges on your Orders table and you want him to be able to grant the SELECT privilege on the table to others, what commands would you need to issue?
6. What happens when you revoke a system privilege from a user to whom it was granted WITH ADMIN OPTION?
7. What are three types of auditing that you can perform in an Oracle database and how do you configure each?
8. How can a user who creates a table configure auditing of who inserts data into the table and when the action was performed?
9. If a user deletes data from a table on which auditing of DELETE statements is configured but decides to roll back the transaction, will an audit trail of the attempted DELETE exist?
10. What are the possible locations of the audit trail in Oracle8i?
11. What commands are used to configure system and object privileges? What commands are used to configure auditing of statements, system privileges, and object access?

After you configure profiles to deal with password management and resource limits, and after you have created users so that individuals can connect to your database, you need to grant privileges so that users can perform system tasks, or access database objects. Oracle divides privileges into two groups: system privileges that are typically granted by the DBA and enable users to perform tasks, such as create objects or alter session settings; and, object privileges that enable a user to query or modify data in the database, which are granted by the owner of the object. In some environments where security is paramount, there may also be a requirement to track which users made use of what privileges, or accessed what objects. All of these can be accomplished in Oracle8i and fall under the category of managing privileges.

Overview of Managing Privileges

The task of managing privileges involves two distinct sets of operations:

- ♦ The assignment and management of system and object privileges by the database administrator or object owner
- ♦ The tracking of privilege usage in the database through the built-in auditing facility provided by Oracle8i

Most DBAs will spend the vast majority of their time in this area with the granting and revoking of system and object privileges. In very few environments is the auditing facility that is available in Oracle8i effectively used. The reasons for this are varied but usually deal with the fact that applications that make use of Oracle databases typically include their own logging or auditing functionality through the use of triggers, and that the performance overhead of auditing is quite high. That notwithstanding, a combination of the two approaches is recommended.



Even though the auditing capability of Oracle is not widely used, the Oracle8i: Architecture and Administration exam will test your thorough understanding of how to configure, enable, disable, and monitor auditing. Do not skip this part of the chapter in your preparation for the exam.

In the management of privileges and ensuring that user can perform the tasks they need to do their job or access database objects as needed, two commands are used extensively: GRANT and REVOKE. The GRANT command is used to grant privileges to a user to perform a specific task, such as CREATE TABLE or DROP INDEX, or to perform operations on database objects, such as GRANT SELECT ON a table or GRANT INSERT ON a table. The REVOKE command performs the opposite operation of GRANT — it takes away privileges from the user, such as REVOKE CREATE TABLE or REVOKE SELECT ON a table.

To track a user's actions in the database, two auditing commands are available: `AUDIT` and `NOAUDIT`. The `AUDIT` command will enable auditing of a particular statement, such as `AUDIT CREATE TABLE`, or `AUDIT DROP INDEX`, or a user's access to a database objects, as in `AUDIT SELECT ON` a view or `AUDIT INSERT ON` a table. Auditing is disabled by default for all database actions, but once it is turned on can also be disabled by issuing the `NOAUDIT` command, such as `NOAUDIT CREATE TABLE` or `NOAUDIT DROP INDEX`. You can also disable auditing on object access as in `NOAUDIT SELECT ON` a view or `NOAUDIT UPDATE ON` a table.

Types of Privileges

Objective

Identify system and object privileges

Grant and revoke privileges

Control operating system or password file authentication

If you define privileges as the ability to perform a specific task or manipulate a database object, you can break down the privileges available in Oracle into two distinct types: system privileges, which let you perform a specific task such as create a view or alter the database, and object privileges, which enable you to manipulate an object such as execute a stored procedure or insert data into a table. System privileges are typically granted to users by the DBA, and even then the DBA will not and should not grant a user more privileges than are required to perform their designated tasks in the database. Object privileges are granted by the owner of the object and determine which other users, besides the owner and the DBA, are allowed to manipulate objects.

System privileges

Oracle8i contains some 126 privileges that can be assigned to users. The exact number of system privileges that can be assigned varies with the options selected when you create the database, and/or run additional scripts to add functionality. As the Oracle feature set develops and new options are added, the number of privileges is expected to increase as well. When categorizing the system privileges in Oracle8i, they can be broadly classified into privileges that deal with the following areas:

- ♦ **System-wide operations**— This category of privileges allows the holder to make changes to the settings of the database or instance as a whole. This includes privileges such as `ALTER SYSTEM`, which enables you to modify the currently running settings for the instance, `ALTER DATABASE`, which enables you to move files in the database and change other database settings, `CREATE SESSION`, which enables you to connect to the instance, `CREATE TABLESPACE`, which enables you to create a tablespace, and many others.

The granting of system-wide privileges is something that is done by the DBA, and, with the exception of the `CREATE SESSION` privilege required by every user in order to connect to the instance, is generally not performed too often. In other words, the DBA keeps control of privileges that affect the entire database because allowing too many users to hold these privileges could lead to the potential of a single change by a user causing problems affecting everyone else.

- ♦ **Management of own objects**—In order for a user to be able to create, alter, or drop tables, views, and other objects, the privilege to do so needs to be granted. Once a user has been assigned the privilege, they are able to manage objects in their own schema. The privileges that can be granted include `CREATE TABLE`, `CREATE VIEW`, and so on.

Once a user has been granted the privilege to create objects in his or her schema, other restrictions, such as quotas on tablespaces, still apply. The combination of privileges granted and other systems settings configured for the user will determine whether or not the action will succeed. However, once a user has created an object, the modification of the object as well as its removal from the database are automatically inherited. This means that if you have been granted the `CREATE TABLE` privilege, once you create a table in your schema, you can also issue the `ALTER TABLE` and `DROP TABLE` commands on the table you have created. This is because object owners have full privileges on objects that they create.

- ♦ **Management of objects in any schema**—A certain group of privileges, when granted, allows users holding them to manage any objects of the same type that exist in the database. These privileges, such as `CREATE ANY TABLE`, `ALTER ANY INDEX`, or `DROP ANY PROCEDURE`, are typically not granted to any user except the DBA.

Since holding any of these special system privileges enables a user to modify objects in other schemas, this can be considered a security breach and should be avoided. However, it can be useful when objects, such as indexes, need to be created to support the way the database is accessed, and you want to ensure that the index is in the same schema as the table. Again, even this last example is something that a DBA would probably do, and the `CREATE ANY INDEX` privilege would not be granted to others.

If you want a complete list of system privileges that are available in Oracle, issue the following query, which retrieves the list of privileges that can be granted to users. The majority of these are granted to the DBA role, which is held by those users who are to administer the database, as well as `SYSTEM` and `SYS`. The output of the query has been truncated to save space.



For information on the DBA role and roles in general, please refer to Chapter 19.

```
SQL> SELECT NAME FROM SYSTEM_PRIVILEGE_MAP;
```

```
NAME
```

```
-----  
ALTER SYSTEM  
AUDIT SYSTEM  
CREATE SESSION  
ALTER SESSION  
RESTRICTED SESSION  
CREATE TABLESPACE  
ALTER TABLESPACE  
MANAGE TABLESPACE  
DROP TABLESPACE  
UNLIMITED TABLESPACE  
CREATE USER  
BECOME USER  
ALTER USER  
DROP USER  
CREATE ROLLBACK SEGMENT  
ALTER ROLLBACK SEGMENT  
DROP ROLLBACK SEGMENT  
CREATE TABLE  
CREATE ANY TABLE  
ALTER ANY TABLE  
BACKUP ANY TABLE  
DROP ANY TABLE  
LOCK ANY TABLE  
COMMENT ANY TABLE  
SELECT ANY TABLE  
INSERT ANY TABLE  
UPDATE ANY TABLE  
DELETE ANY TABLE  
CREATE CLUSTER  
CREATE ANY CLUSTER  
ALTER ANY CLUSTER  
DROP ANY CLUSTER  
CREATE ANY INDEX  
ALTER ANY INDEX  
DROP ANY INDEX  
CREATE SYNONYM  
CREATE ANY SYNONYM  
DROP ANY SYNONYM  
SYSDBA  
SYSOPER
```

```
...
```

```
CREATE ANY OUTLINE
```

```
ALTER ANY OUTLINE  
DROP ANY OUTLINE  
ADMINISTER RESOURCE MANAGER  
ADMINISTER DATABASE TRIGGER
```

```
126 rows selected.
```

```
SQL>
```

A number of details about system privileges are not immediately clear or obvious. One of these is that if you issue the previous command in your own database and scroll through the list of privileges returned, you will notice that there is no CREATE INDEX privilege. There is a CREATE ANY INDEX privilege, which enables you to create an index in any schema, including your own, but no CREATE INDEX privilege to create indexes on your schema only. This is not an oversight, but rather by design.

If you want to be able to create an index on your own table, you are allowed to do so because you already own the table. As mentioned previously, object owners have full privileges on the objects that they create and, for a table, this also means that you can create indexes on your own tables. However, if you want to create an index on someone else's table, you need to be granted the CREATE ANY INDEX privilege because the owner of the index and the owner of the table will be two different users. An interesting side effect of this is that once granted the CREATE ANY INDEX privilege, the user holding it can also create the index in the same schema as the table to which the index will belong.

A side effect of the capability to create indexes on tables you own is that if you are granted the CREATE TABLE privilege, it not only includes the CREATE INDEX privilege but also the ANALYZE privilege. This enables users to perform an analysis and tweak the cost-based optimizer to ensure that the best execution plan is selected for a particular query. Once you create an index in your schema, the ability to analyze the index is also inherited. As you can see, granting CREATE TABLE to a user does a number of things that should be remembered.

In order for users to be able to create segments in the database (tables, indexes, clusters, and so on) a quota needs to be assigned on the tablespaces where the segments will be created. As you saw in Chapter 17, quotas can be assigned using the CREATE USER or ALTER USER command. Oracle also includes a special privilege that grants the holder an unlimited quota on all tablespaces in the database. This is the UNLIMITED TABLESPACE system privilege. Any user that has been granted the UNLIMITED TABLESPACE system privilege has a QUOTA UNLIMITED on every tablespace—including SYSTEM. While normal users would not have this privilege, it can be useful to grant it to a user whose schema will be where most objects in the database are created. In this way, you do not need to grant explicit quotas to the

user, and, furthermore, the user's objects will never run out of space because of quota restrictions. Granting of the UNLIMITED TABLESPACE privilege to that type of user is quite common for application developers of commercial packages using Oracle. By default, the UNLIMITED TABLESPACE privilege is assigned to the RESOURCE role, which itself is assigned to the DBA.

SYSDBA and SYSOPER special system privileges

Objective

Control operating system or password file authentication

Oracle8i also includes two special system privileges called SYSOPER and SYSDBA. These privileges were introduced in Chapter 2, and later used to stop and start an instance, as discussed in Chapter 3. These privileges are used when password file authentication is configured to enable privileged users (that is, those that have been granted SYSOPER or SYSDBA) to perform certain special database operations such as stop and start the instance, perform database backup or recovery, or even create the database. If you configured operating system authentication, the OSOPER and OSDBA privileges provide the same level of functionality.



While the Oracle documentation refers to SYSDBA and SYSOPER, as well as OSDBA and OSOPER, as system privileges, most database administrators working with Oracle actually call them *roles*. As you will see in Chapter 19, a role is a container for a number of privileges that can be assigned as a group to a user. SYSDBA and SYSOPER, when granted, confer a number of privileges to the user and are therefore closer to the definition of a role (that is, a collection of privileges) than a system privilege, which usually refers to the ability to perform a specific individual task. Whether you use the terms *system privilege* or the term *role* to refer to SYSDBA and SYSOPER is up to you. The exam may actually use either when asking a question relating to SYSDBA and SYSOPER.

Cross-Reference

For more information on password file and operating system authentication for privileged users, please consult Chapter 2.

The specific tasks that can be performed when a user is granted the SYSDBA or SYSOPER privileges are outlined in Table 18-1. When a user connects to the instance as a user with one of these privileges, Oracle will actually connect the individual issuing the command that follows as the user SYS and grant him/her all of the privileges that are granted to the user SYS, plus the additional special privileges that the SYSOPER or SYSDBA privilege. Because the user SYS is the owner of the database and the data dictionary, and has full privileges throughout the database, granting the SYSDBA or SYSOPER privileges should be carefully planned.

Table 18-1
Commands Available to SYSOPER and SYSDBA

<i>Privilege</i>	<i>Commands Available (including all variations)</i>
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE MOUNT ALTER DATABASE OPEN ALTER DATABASE BACKUP CONTROLFILE ALTER TABLESPACE BEGIN BACKUP ALTER TABLESPACE END BACKUP ALTER DATABASE RECOVER DATABASE ALTER DATABASE ENABLE RESTRICTED SESSION ALTER DATABASE DISABLE RESTRICTED SESSION ALTER DATABASE ARCHIVELOG
SYSDBA	SYSOPER privileges WITH ADMIN OPTION REATE DATABASE RECOVER DATABASE UNTIL

SYSDBA and SYSOPER privileges can be granted or revoked from users using the normal GRANT and REVOKE commands, but can only be done by a user who also holds the same privilege (that is, you must have been granted SYSDBA in order to grant SYSDBA). However, the granting of these privileges requires that a password file be configured and that the Oracle initialization parameter REMOTE_LOGIN_PASSWORDFILE be set to EXCLUSIVE in the INIT.ORA file. Failure to configure password file authentication properly will result in the following error when you attempt to grant the privilege:

```
SQL> connect internal/oracle@orcl.mars.bradsys.com
Connected.
SQL> GRANT SYSDBA TO SYSTEM;
GRANT SYSDBA TO SYSTEM
*
ERROR at line 1:
ORA-01994: GRANT failed: cannot add users to public password
file
```

```
SQL>
```

If you want to determine which users currently hold the SYSDBA and/or SYSOPER privileges, query the V\$PFILE_USERS view as follows:

```
SQL> SELECT * FROM V$PFILE_USERS;

USERNAME                                SYSDB SYSOP
-----
INTERNAL                                TRUE  TRUE
SYS                                       TRUE  TRUE

SQL>
```

A value of TRUE in the column under the heading SYSDB indicates that the user has been granted the SYSDBA privilege, while a value of TRUE in the column with the SYSOP heading indicates that the user has been granted the SYSOPER privilege. As you can see, both SYS and INTERNAL are granted both privileges. If you get a “no rows select” message when querying V\$PFILE_USERS, this indicates that the REMOTE_LOGIN_PASSWORDFILE parameter is not set to EXCLUSIVE and you are not using password file authentication.

Granting system privileges

Objective

Grant and revoke privileges

You can grant system privileges in Oracle using the GRANT command or Oracle Enterprise Manager (OEM). When granting system privileges, you need to ensure that you are connected as a DBA or as a user that has the privilege to grant the system privilege in question to other users.

When using the GRANT command, the syntax for granting system privileges is as follows:

```
GRANT system_priv [, system_priv, ...]
TO user | role | PUBLIC
[, user | role | PUBLIC, ...]
[WITH ADMIN OPTION]
```

As you can see from the syntax, it is possible to grant more than one system privilege to the same user or role, as well as grant the same system privilege to more than one user or role, or any combination thereof. Oracle allows the DBA to perform the granting of a number of privileges to a number of users simultaneously. Furthermore, if you would like all users to have the same privilege, you can grant it to the special role PUBLIC, which all users are assigned and cannot be removed from.



For information on roles, please refer to Chapter 19.

For example, if you want all users to have the CREATE SESSION privilege, issue the following command:

```
SQL> GRANT CREATE SESSION TO PUBLIC;
```

```
Grant succeeded.
```

```
SQL>
```

If you want the user JohnS to be able to create indexes and views in any schema, issue the following command:

```
SQL> GRANT CREATE ANY INDEX, CREATE ANY VIEW TO JOHNS;
```

```
Grant succeeded.
```

```
SQL>
```

An option when granting system privileges is to use the WITH ADMIN OPTION clause. The use of this clause needs to be planned carefully as it can be difficult to completely reverse the process, as you will find out later in this chapter when discussing the removal of system privileges. When you specify the WITH ADMIN OPTION during the granting of privileges to a user, role or PUBLIC, you are basically saying that the user to whom the privilege is being granted WITH ADMIN OPTION is also able to grant the same privilege to other users or roles in the database.

The ability to allow other users to be granted privileges, which they in turn can grant to others, can be handy when you need to delegate some administrative authority over the database. An example of this could be the ability for someone else, besides the DBA, to be able to create database objects and grant the same privilege to others. In a software development shop, this could be useful as the DBA may be responsible for several databases whereas certain users can then maintain who will be able to create objects.

For example, if you want BobW to be able to create several database objects, and also be able to grant the same privileges to others, issue the following command:

```
SQL> GRANT CREATE TABLE, CREATE VIEW, CREATE PROCEDURE,  
2 CREATE TYPE TO BOBW  
3 WITH ADMIN OPTION;
```

```
Grant succeeded.
```

```
SQL>
```

After this is done, BobW could connect to the instance and grant the same privileges (or a portion thereof) to MarkS, as in the following example:

```
SQL> connect bobw/oracle@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> GRANT CREATE TABLE, CREATE VIEW  
2 TO MARKS;
```

```
Grant succeeded.
```

```
SQL>
```

You should note that once a user has been granted a system privilege WITH ADMIN OPTION, that same user can also grant the same privilege to others — also WITH ADMIN OPTION. In other words, once the DBA has granted a privilege WITH ADMIN OPTION to a user in the database, he or she has effectively lost control over whom that privilege can be granted to. This can be a dangerous thing to do and is not normal practice in most organizations.

Granting system privileges using Oracle Enterprise Manager

You can also grant system privileges with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Granting System Privileges Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to grant system privileges as a user with appropriate permissions to do so.
2. Expand the database and then the Security node, as shown in Figure 18-1.

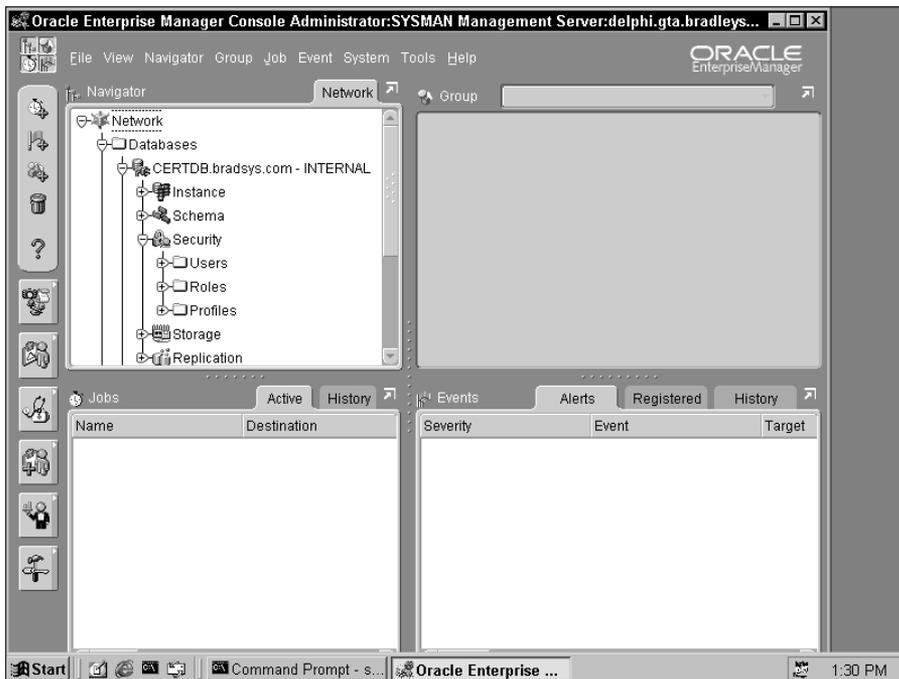


Figure 18-1: Expand the Security node of the database to locate the Users node.

- Expand the Users node and right-click the user to whom you want to grant the system privilege and select Edit, as shown in Figure 18-2.



Figure 18-2: Expand Users and select the user to grant system privileges to.

- On the Edit User dialog box, click System Privileges to view a list of system privileges granted, as well as a list of all available system privileges that may be granted to the user, as shown in Figure 18-3.

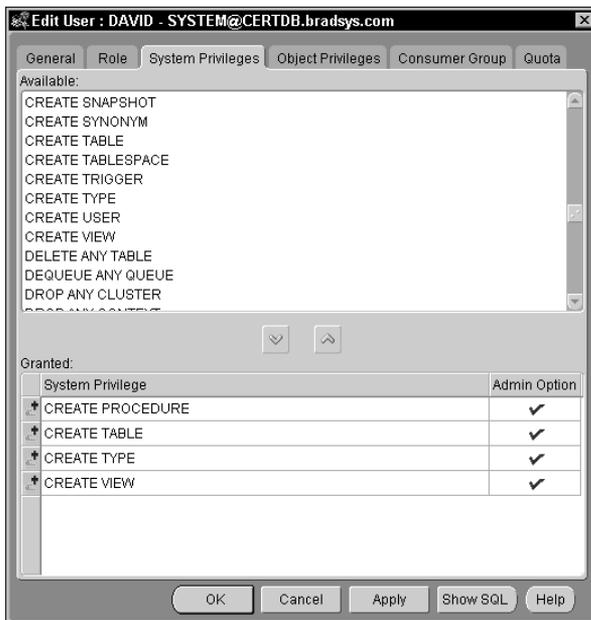


Figure 18-3: Click the System Privileges tab to see a list of system privileges granted and available.

5. Scroll down the list of available system privileges and select the ones you want to grant, and then click on the down arrow to assign them to the user, as shown in Figure 18-4.

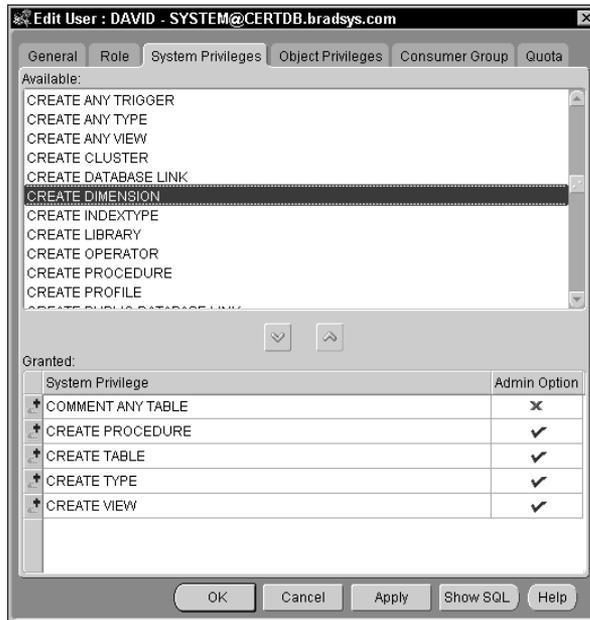


Figure 18-4: Select the system privileges to grant. Click on the System Privileges tab to see a list of system privileges granted and available.

6. If you want to grant the system privilege WITH ADMIN OPTION, click on the Admin Option box next to the privilege until a checkmark appears, as shown in Figure 18-5. You can toggle the checkmark on and off until all privileges have the proper setting for WITH ADMIN OPTION.

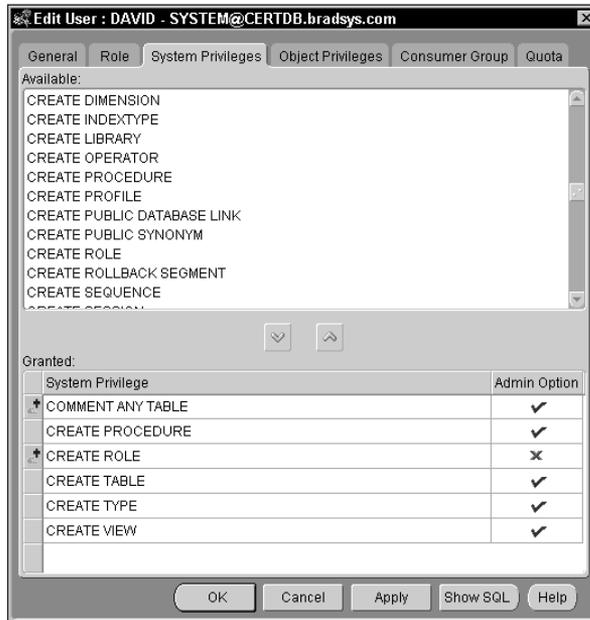


Figure 18-5: Click on the box next to the privilege to grant it WITH ADMIN OPTION.

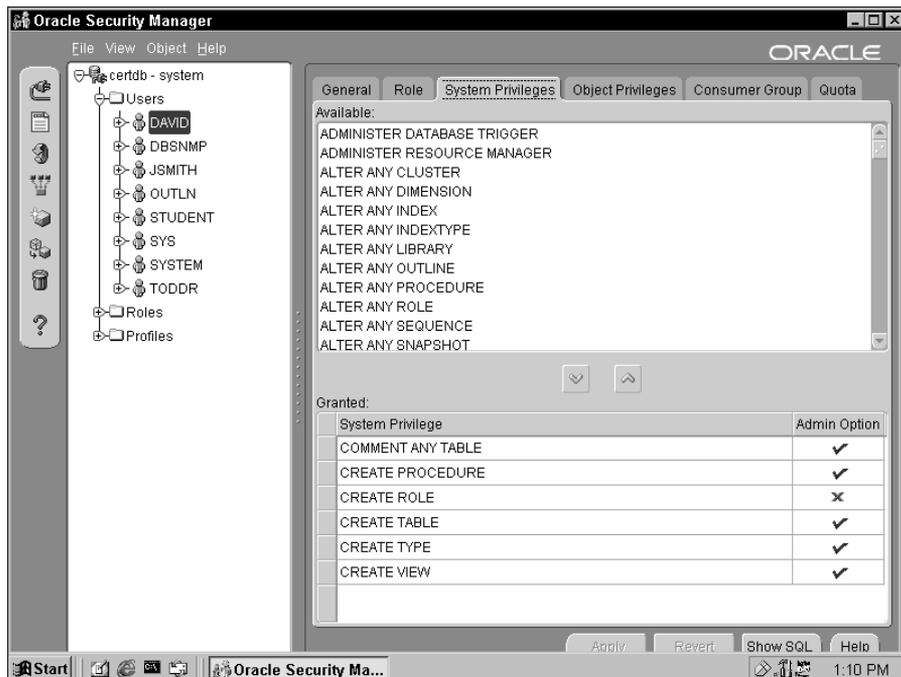


Figure 18-6: You can also use Security Manager to assign system privileges.

7. When you have granted the system privileges desired, click Apply to save your changes and OK to exit the dialog box.

If you are using Security Manager from the Database Administration menu and not the Enterprise Manager console to grant system privileges, the interface will resemble the one shown in Figure 18-6, although the functionality will be exactly the same. Either Security Manager or the Oracle Enterprise Manager console can be used to grant system privileges to users, provided you have the necessary privileges yourself to perform the task.

Revoking system privileges

To revoke system privileges, use Oracle Enterprise Manager or the REVOKE command. The syntax of the REVOKE command is as follows:

```
REVOKE system_priv [, system_priv, ...]
      FROM user | role | PUBLIC
      [, user | role | PUBLIC];
```

As with the GRANT command, you can revoke more than one privilege from a user or role at the same time, or revoke the same privilege from multiple users or roles, or the PUBLIC role, or any combination thereof.

If you are revoking privileges from a role, user, or PUBLIC, the privileges must have previously been granted to the same user, role, or PUBLIC. This means that if you decide to revoke the CREATE TABLE privilege from PUBLIC, the privilege will not be revoked from all users, but only from the PUBLIC role itself. If the privilege was never granted to PUBLIC, the REVOKE command will have no effect—that is, you can't revoke something you never granted.

In understanding how the REVOKE command and the GRANT command interrelate, an analogy might be in order. If you have a football and a group of people on a football field that want to play, and if you give them the ball to play with, it is similar to granting a system privilege to PUBLIC—you have one ball that everyone will have access to. If you do not want David, one of the players on the field, to have access to the ball, you cannot simply take the ball away since this will take it away from everyone—that is, you cannot revoke the privilege from him since it was granted to PUBLIC (all players collectively). In order to not allow David to gain access to the ball, you need to remove him from the field (that is, drop the user from the database).

The scenario presented previously changes if you have a number of balls and give each player on the field a separate football. In this case, if you did not want David to have a ball, you simply remove it from him—that is, REVOKE the system privilege from the user. All other players will still have their own football to play with, but David will not be able to play since he has no football. Unlike the real world where David could potentially muscle his way around and get a ball from another player, Oracle acts as a very tough referee and ensures that a user cannot access privileges granted to others.

Another element to consider when you revoke privileges is that doing so may cause side effects that may not have been intended. For example, if you grant a user the SELECT ANY TABLE privilege and the CREATE VIEW privilege, if the user's SELECT ANY TABLE privilege is then revoked, it may invalidate any views that he or she has created. This is because once the privilege is removed, objects that depended on the privilege's existence may no longer be able to access other database objects that they depend on. To ensure that revoking a system privilege does not cause any negative side effects, you should review object dependencies in Oracle Enterprise Manager or use the DBA_DEPENDENCIES view prior to revoking the privilege.

Finally, privileges that were granted WITH ADMIN OPTION to a user will not cascade if the privilege is revoked from the user that the privilege was initially granted to. For example, previously the CREATE TABLE, CREATE VIEW, and CREATE PROCEDURE privileges were granted to BobW WITH ADMIN OPTION. BobW then granted the CREATE TABLE and CREATE VIEW privileges to MarkS. If you revoke the CREATE VIEW privilege from BobW, this will not revoke it from MarkS, as shown in this example:

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> GRANT SELECT ANY TABLE TO BOBW, MARKS;

Grant succeeded.

SQL> REVOKE CREATE VIEW FROM BOBW;

Revoke succeeded.

SQL> connect bobw/password@certdb.delphi.bradsys.com
```

Connected.

```
SQL> SELECT LASTNAME, FIRSTNAME FROM
  2 STUDENT.INSTRUCTORS;
```

LASTNAME	FIRSTNAME
Harrison	Michael
Keele	Susan
Ungar	David
Jamieson	Kyle
Cross	Lisa

```
SQL> CREATE VIEW STUD_INSTRUCT AS
  2 SELECT LASTNAME, FIRSTNAME FROM
  3 STUDENT.INSTRUCTORS;
STUDENT.INSTRUCTORS
  *
```

ERROR at line 3:

ORA-00942: table or view does not exist

```
SQL> connect marks/oracle@certdb.delphi.bradsys.com
```

Connected.

```
SQL> SELECT LASTNAME, FIRSTNAME FROM
  2 STUDENT.INSTRUCTORS;
```

LASTNAME	FIRSTNAME
Harrison	Michael
Keele	Susan
Ungar	David
Jamieson	Kyle
Cross	Lisa

```
SQL> CREATE VIEW STUD_INSTRUCT AS
  2 SELECT LASTNAME, FIRSTNAME FROM
  3 STUDENT.INSTRUCTORS;
```

View created.

```
SQL> SELECT * FROM STUD_INSTRUCT;
```

LASTNAME	FIRSTNAME
Harrison	Michael
Keele	Susan
Ungar	David
Jamieson	Kyle
Cross	Lisa

```
SQL>
```

In this series of commands, both MarkS and BobW were granted the SELECT ANY TABLE privilege. BobW, who previously was granted the CREATE VIEW privilege WITH ADMIN OPTION had the privilege revoked. BobW can still query the Student.Instructors table, but any attempt to create a view on the table results in an error as BobW no longer holds that privilege. However, MarkS, who was granted the CREATE VIEW privilege by BobW prior to the DBA taking it away from BobW, is able to both query the Student.Instructors table and create a view. This is because revoking the CREATE VIEW privilege from BobW does not automatically revoke it from any other user or role that BobW granted the privilege to. In other words, the REVOKE command for system privilege does not cascade any privileges that were granted by users who were previously assigned privileges WITH ADMIN OPTION.

Revoking system privileges using Oracle Enterprise Manager

You can also revoke system privileges with Oracle Enterprise Manager using the following steps.

STEP BY STEP: Revoking System Privileges Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager console and connect to the database in which you want to revoke system privileges as a user with appropriate permissions to do so.
2. Expand the database and then Security.
3. Expand Users and right-click the user from whom you want to revoke the system privilege and select Edit.
4. On the Edit User dialog box, click System Privileges to view a list of system privileges granted to the user.
5. On the bottom, review the privileges granted and select the one you want to revoke, and then click the up arrow to remove it from the list of granted privileges, as shown in Figure 18-7.
6. When you have revoked all of the system privileges desired, click Apply to save your changes and OK to exit the dialog box.

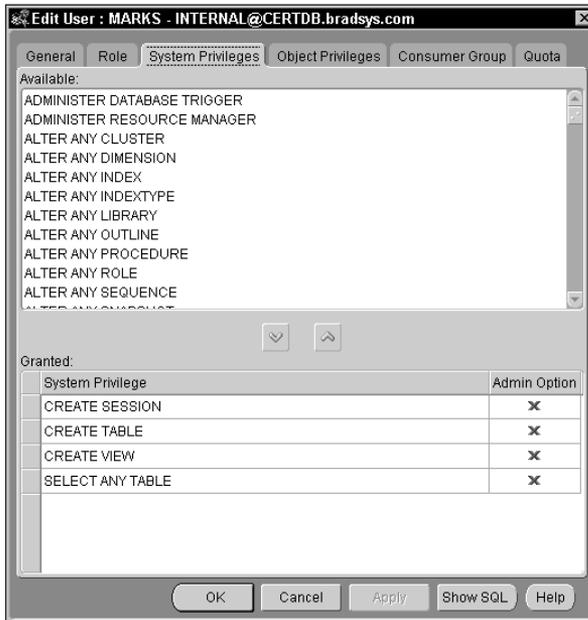


Figure 18-7: Select the system privileges you want to revoke on the System Privileges tab for the user and click the up arrow to revoke it.

Getting information on system privileges

Oracle provides a number of methods to determine which system privileges have been granted to which users. One that you have already seen (in the previous discussion about revoking system privileges using Oracle Enterprise Manager) is the ability to view system privileges assigned to users through OEM. You can also query two data dictionary views to determine which system privileges have been assigned to which users, or what a user's currently running system privileges are. These are the `DBA_SYS_PRIVS` and `SESSION_PRIVS` data dictionary views.

The `DBA_SYS_PRIVS` data dictionary view can be used by the DBA to determine which privileges have been granted to which users, or what a user's set of system privileges are. The structure of the `DBA_SYS_PRIVS` view is as follows:

```
SQL> DESC DBA_SYS_PRIVS;
Name                               Null?    Type
-----
GRANTEE                             NOT NULL VARCHAR2(30)
PRIVILEGE                            NOT NULL VARCHAR2(40)
```

```
ADMIN_OPTION                                VARCHAR2(3)
```

```
SQL>
```

If you want to find out which system privileges are granted to BobW, issue the following query:

```
SQL> col grantee format a15
SQL> col privilege format a25
SQL> SELECT * FROM DBA_SYS_PRIVS
  2  WHERE GRANTEE='BOBW';
```

GRANTEE	PRIVILEGE	ADM
BOBW	COMMENT ANY TABLE	YES
BOBW	CREATE PROCEDURE	YES
BOBW	CREATE ROLE	NO
BOBW	CREATE TABLE	YES
BOBW	CREATE TYPE	YES
BOBW	SELECT ANY TABLE	NO

```
6 rows selected.
```

```
SQL>
```

The result would indicate the name of the user the privilege was granted to (GRANTEE, the privilege, and whether or not the privilege was granted to the user WITH ADMIN OPTION (the ADM column heading).

If you want to find out which database users or roles have the CREATE TABLE privilege, issue the following query:

```
SQL> col grantee format a30
SQL> SELECT GRANTEE FROM DBA_SYS_PRIVS
  2  WHERE PRIVILEGE='CREATE TABLE';
```

```
GRANTEE
-----
BOBW
CONNECT
DBA
JOHNS
MARKS
OEM_MONITOR
RECOVERY_CATALOG_OWNER
RESOURCE
```

```
8 rows selected.
```

```
SQL>
```

As you can see, the result returns those users you would have suspected, such as BobW, JohnS, and MarkS, but it also returns a number of other grantees, such as DBA, CONNECT, RESOURCE, and so on. These are roles that have been granted the CREATE TABLE privilege.

A related view to DBA_SYS_PRIVS is the USER_SYS_PRIVS view (the ALL_SYS_PRIVS view does not exist). This view will return the system privileges that have been granted directly to the user, or to PUBLIC, and are not inherited from any other role that the user has been granted. For example, when connected as BobW, querying USER_SYS_PRIVS returns the following:

```
SQL> connect bobw/oracle@certdb.delphi.bradsys.com
Connected.
SQL> SELECT * FROM USER_SYS_PRIVS;
```

USERNAME	PRIVILEGE	ADM
BOBW	COMMENT ANY TABLE	YES
BOBW	CREATE PROCEDURE	YES
BOBW	CREATE ROLE	NO
BOBW	CREATE TABLE	YES
BOBW	CREATE TYPE	YES
BOBW	SELECT ANY TABLE	NO
PUBLIC	CREATE SESSION	NO

```
7 rows selected.

SQL>
```

A user can always query the USER_SYS_PRIVS view just like any other USER_ data dictionary view.

To get a complete list of all currently running system privileges for the user's session, the user can also query the SESSION_PRIVS data dictionary view. It will return all of the system privileges that were granted to the user directly, as well as any other system privileges that the user has received due to having a role granted to him or her. For example, if you compare the previous USER_SYS_PRIVS output to the output from SESSION_PRIVS for BobW, you will note that several additional privileges (ALTER SESSION, CREATE CLUSTER, CREATE SYNONYM, and so forth) are available because BobW has been assigned other roles and therefore inherited some more system privileges.

```
SQL> connect bobw/oracle@certdb.delphi.bradsys.com
Connected.
SQL> SELECT * FROM SESSION_PRIVS;
```

PRIVILEGE
CREATE SESSION
ALTER SESSION
CREATE TABLE

```
COMMENT ANY TABLE  
SELECT ANY TABLE  
CREATE CLUSTER  
CREATE SYNONYM  
CREATE VIEW  
CREATE SEQUENCE  
CREATE DATABASE LINK  
CREATE ROLE  
CREATE PROCEDURE  
CREATE TYPE
```

```
13 rows selected.
```

```
SQL>
```

Data dictionary accessibility and system privileges in Oracle8i

In Oracle8i it is possible to restrict who has access to view data dictionary objects. This is done by setting the Oracle initialization parameter `O7_DICTIONARY_ACCESSIBILITY` to `FALSE` in the `INIT.ORA` file. The default value for this parameter is `TRUE`, which means that all users can freely query the data dictionary and its views. If this parameter is set to `FALSE`, only users that have been assigned the `SYSDBA` or `SYSOPER` system privileges (roles) are able to query the data dictionary. What the parameter actually does is determine whether users will be able to query objects that exist in the `SYS` schema. When set to `TRUE`; the `SYS` schema is available to be viewed by everyone; if set to `FALSE`, Oracle verifies that the user attempting to view the `SYS` schema has been granted the `SYSDBA` or `SYSOPER` system privilege and is connected as a privileged user.

Use of this parameter is one way to ensure that users do not find out more about the database than they are permitted to know. It also has an impact on the way certain system privileges behave. When the `O7_DICTIONARY_ACCESSIBILITY` parameter is set to `TRUE`, the `SELECT ANY TABLE` privilege will grant users the ability to query data in any table in the database—that is, all schemas including the `SYS` schema. However, when the parameter is set to `FALSE`, the `SELECT ANY TABLE` privilege will enable users to query data in any schema *except* `SYS`.

The side effects of this are not actually as great as one would imagine. Setting this parameter to `FALSE` will not prevent users from querying data dictionary views such as `USER_` data dictionary views. Users will still be able to make use of system functions and other elements of the database that are required for proper operation. However, users will not be able to query tables in the `SYS` schema directly or incorporate `SYS` schema objects in their own views or procedures. The main reason for this parameter is to ensure that users do not attempt to query parts of the data dictionary that they should not (like the `X$` base tables) and that they do not have the ability to drop data dictionary objects when granted something like the `DROP ANY VIEW` system privilege.

To change the value of the `O7_DICTIONARY_ACCESSIBILITY` parameter, you need to modify the `INIT.ORA` file and stop and restart your instance. The parameter cannot be changed using the `ALTER SYSTEM` command.

Object privileges

Object privileges enable you to manipulate objects in your Oracle database. They are assigned by the owner of the object (that is, the user who created them) and can be assigned to others with the ability to further grant those privileges. Because different objects in Oracle have different characteristics, the types of privileges that can be granted differ as well. Table 18-2 lists the object privileges available and what objects they can be assigned to. When reviewing the table, keep in mind that a Program Unit in Oracle refers to a stored procedure, function, package, or type.

Table 18-2
Object Privileges Available and the Objects Affected

<i>Privilege</i>	<i>TABLE</i>	<i>VIEW</i>	<i>SEQUENCE</i>	<i>PROGRAM UNIT</i>
SELECT	X	X	X	
INSERT	X	X		
UPDATE	X	X		
DELETE	X	X		
ALTER	X		X	
EXECUTE				X
INDEX	X			
REFERENCES	X			

Granting object privileges

Like system privileges, object privileges can be granted using the GRANT command in SQL*Plus or by using Oracle Enterprise Manager. However, unlike system privileges, which are granted by the DBA, object privileges can only be granted by the owner of the object or a user that has been granted privileges to the object by the owner WITH GRANT OPTION.

The syntax of the GRANT command for object privileges is

```
GRANT ALL [PRIVILEGES] | object_priv [(column, column, ...)]
    [, object_priv [(column, column, ...()) , ...]
ON [schema_name.]object_name
TO user | role | PUBLIC
    [, user | role | PUBLIC, ...]
[WITH GRANT OPTION]
```

A number of elements of the syntax need further scrutiny:

- ♦ The WITH GRANT OPTION enables you to permit the person to whom the privilege is being granted to grant the same privilege on the same object to other users or roles in the database. This is similar to the WITH ADMIN OPTION for system privileges and should be used sparingly and with proper forethought of the consequences — don't get yourself in trouble by being nice to other users.
- ♦ It is possible to grant all privileges on an object by using the ALL keyword (PRIVILEGES is optional for ALL). When using the GRANT ALL syntax, the privileges that will be granted depend on the object, as outlined in Table 18-2, as well what objects are currently available to the person issuing the command.

For example, if the owner of a table issues the command GRANT ALL, the privileges that will be granted are those that are available for a table, namely SELECT, INSERT, UPDATE, DELETE, ALTER, INDEX, and REFERENCES. However, if the GRANT ALL syntax is issued by someone that has been granted the SELECT, INSERT, and UPDATE privileges on a table WITH GRANT OPTION, then the only privileges that will be granted are those which the grantor (that is, the user doing the granting) has, namely SELECT, INSERT, and UPDATE.

- ♦ It is possible to grant privileges for users to perform INSERT, UPDATE, or REFERENCES operations on individual columns of a table or view by including a column list in the GRANT command. While this may seem like a good idea in that it gives you precise control over the columns that users have access to, it can become a nightmare to maintain and should be avoided. If you need to limit the columns a user should have privileges to, create a view with only those columns and then grant the user the appropriate privileges on the view.

For example, if you want user MarkS to modify the Comments and Email address of instructors that work for you, but not the instructor's rates or other sensitive information, you could grant the UPDATE privilege on only the columns you want MarkS to modify, as follows:

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
SQL> GRANT UPDATE (EMail, Comments)
      2 ON Instructors
      3 TO MarkS;

Grant succeeded.

SQL>
```

The best way to succeed in granting object privileges to users is to be connected to the instance as the owner of the object and then grant the privileges as required, as in this example:

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
```

```
SQL> GRANT SELECT ON Courses TO PUBLIC;

Grant succeeded.
SQL> GRANT ALL ON Students
  2 TO SYSTEM
  3 WITH GRANT OPTION;

Grant succeeded.

SQL>
```

If you attempt to grant a privilege to others that you yourself have not been granted, even if you are the DBA, you will receive an error, as in this example:

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> GRANT SELECT ON Student.Courses
  2 TO Marks;
GRANT SELECT ON Student.Courses
                               *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> SELECT COUNT(*) FROM Student.Courses;

  COUNT(*)
-----
         9

SQL> GRANT SELECT ON Student.Students
  2 TO Marks WITH GRANT OPTION;

Grant succeeded.

SQL>
```

In the previous example the user `SYSTEM`, who has full privileges throughout the database, is not able to grant the `SELECT` privilege to the `Student.Courses` table even though he or she can retrieve data from the table. This is because `SYSTEM` was not granted the `SELECT` privilege to the `Courses` table by `Student`. However, note that `SYSTEM` is able to grant the `SELECT` privilege to the `Student.Students` table to `Marks` WITH GRANT OPTION. This is because `SYSTEM` was assigned `ALL` on `Students` by the owner of the object, the user `Student`, previously and the `WITH GRANT OPTION` was specified at that time.

Granting object privileges using Oracle Enterprise Manager

You can also grant object privileges with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Granting Object Privileges Using Oracle Enterprise Manager (Security Manager)

1. Start the Oracle Enterprise Manager console and connect to the database in which you want to grant object privileges as a user with appropriate permissions to do so.
2. Expand the database and then Security.
3. Expand Users and right-click the user to whom you want to grant object privileges and select Edit.
4. On the Edit User dialog box, click Object Privileges to view a list of schemas in the database, as well as any existing object privileges granted, as shown in Figure 18-8.



Figure 18-8: The Object Privileges tab shows a list of schemas and object privileges granted the user.

5. Expand the schema and then the object type to display a list of objects of the type you want to grant object privileges on, as shown in Figure 18-9.

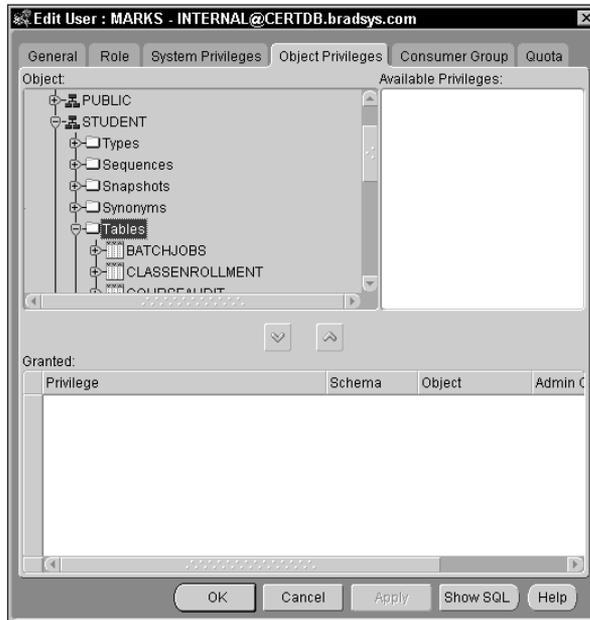


Figure 18-9: Expand the schema and the object type to display a list of objects to which privileges can be assigned.

6. Click on the object to which you want to grant privileges to display a list of privileges that can be granted. Select the appropriate privileges and then click the down arrow to assign the privilege to the user, as shown in Figure 18-10.

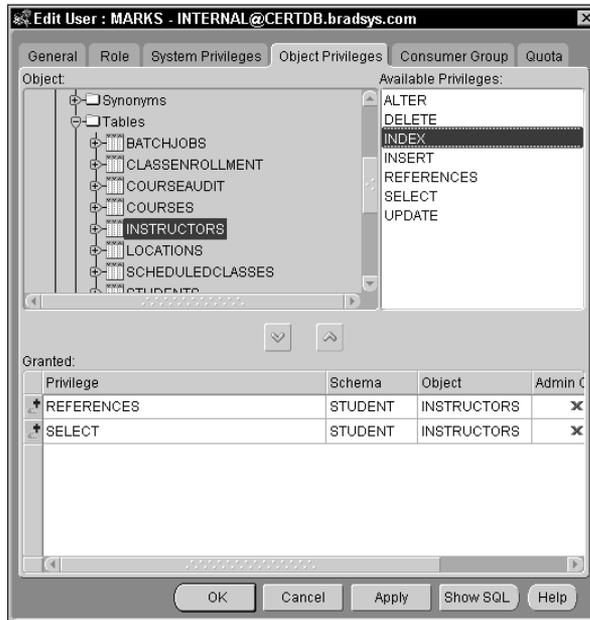


Figure 18-10: Click on the object and select the privileges to assign.

7. If you want to assign the privilege WITH GRANT OPTION, click on the Admin Option column and ensure that the checkmark appears, as shown in Figure 18-11.

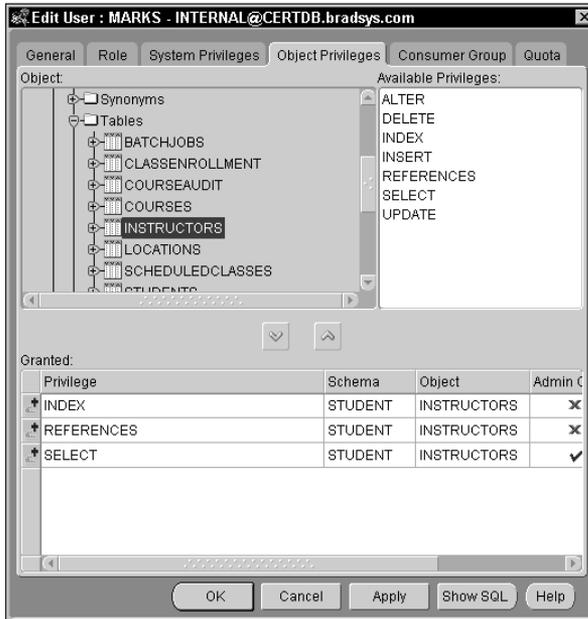


Figure 18-11: A checkmark in the Admin Option column indicates that the privilege will be assigned WITH GRANT OPTION to the user.

- When you have granted all the object privileges to the user, click Apply to save your changes and OK to exit the dialog box. If you get an error similar to the one shown in Figure 18-12, it means that you are connected to the instance as a user other than the owner of the object, or have not been granted the privileges you want to assign WITH GRANT OPTION. Connect to the instance as a user with the appropriate privileges and try again.

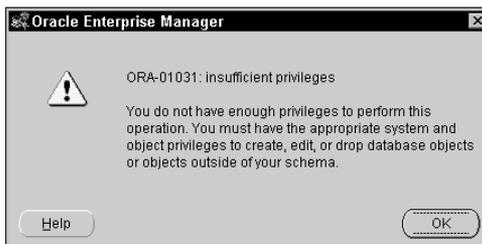


Figure 18-12: This error indicates that you are not connected to the instance as the owner of the object or don't have the ability to grant these privileges to others.

The previous steps showed you how to assign object privileges to a user through the Oracle Enterprise Manager Security Manager component. You can navigate to the object in the Oracle Enterprise Manager Schema Manager component in order to assign privileges to multiple users on the same object. To do so, follow these steps.

STEP BY STEP: Granting Object Privileges Using Oracle Enterprise Manager (Schema Manager)

1. Start the Oracle Enterprise Manager console and connect to the database in which you want to grant object privileges as a user with appropriate permissions to do so.
2. Expand the database and then the Schema node, as shown in Figure 18-13.

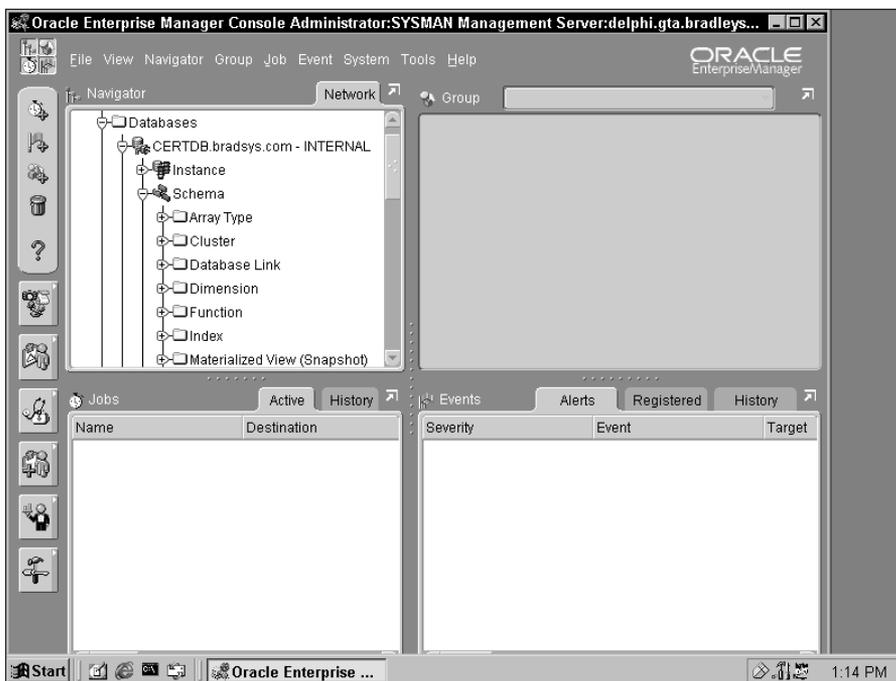


Figure 18-13: Expand the Schema node in Oracle Enterprise Manager to get a list of objects in the database.

3. Expand the object type and then schema that you want to assign object privileges to, as shown in Figure 18-14.

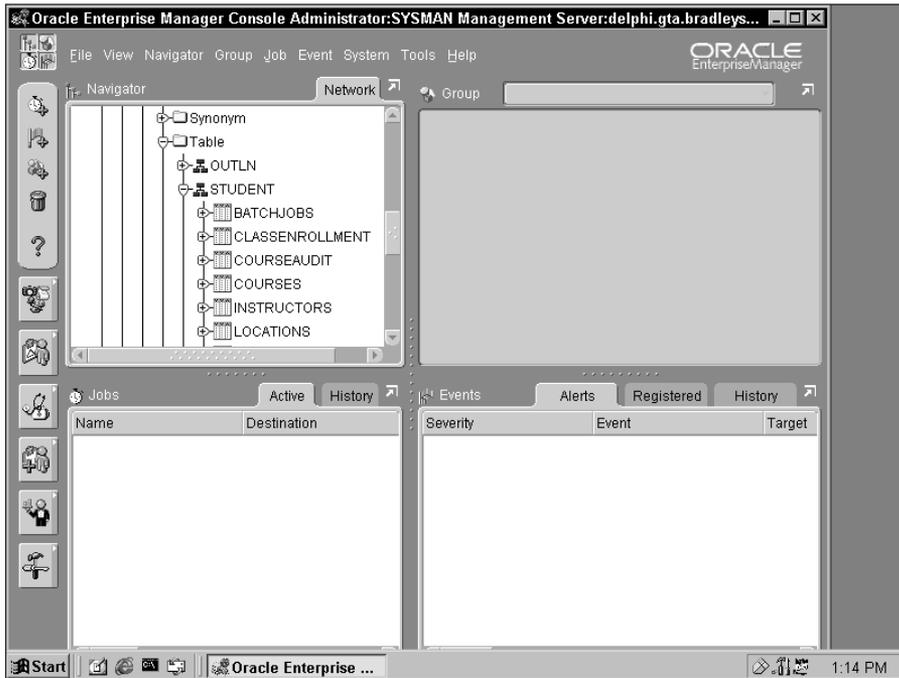


Figure 18-14: Locate the object type and schema that you want to assign object privileges to by expanding the appropriate nodes.

4. Once you have located the object to assign privileges on, right-click the object and select Grant Privileges On as shown in Figure 18-15.

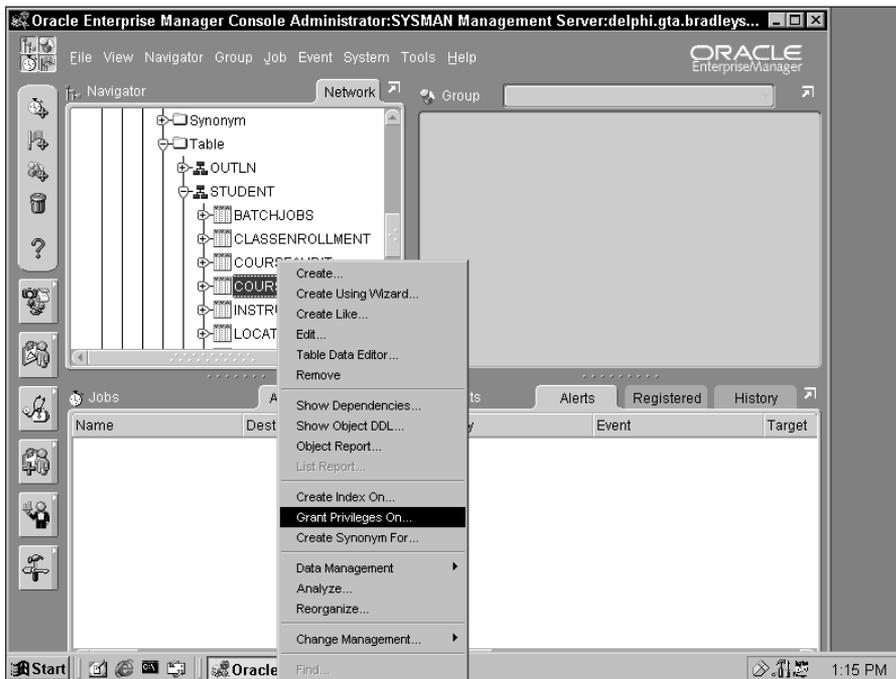


Figure 18-15: Right-click the object and select Grant Privileges On to assign privileges to the object.

5. On the Grant Object Privilege dialog box, select the privileges you want to grant in the top portion, and then the users or roles you want to grant those privileges to on the bottom, as shown in Figure 18-16. If you want to grant these privileges to the grantees selected WITH GRANT OPTION, check the WITH GRANT OPTION button on the bottom left of the dialog box.

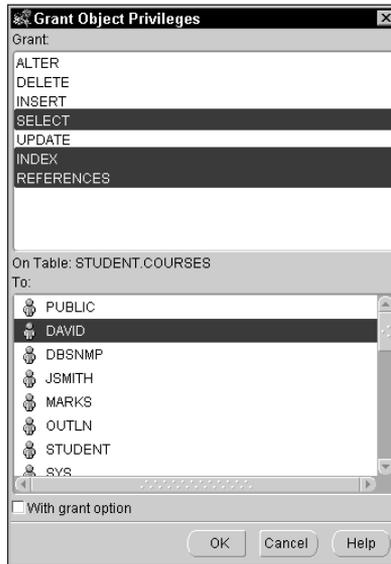


Figure 18-16: Select the privileges to grant and the grantees in the Grant Object privilege dialog box.

6. Click OK to save your changes. If you receive an error similar to that shown in Figure 18-17, ensure that you are connected to the instance as a user with the appropriate privileges — for example, the owner of the object.

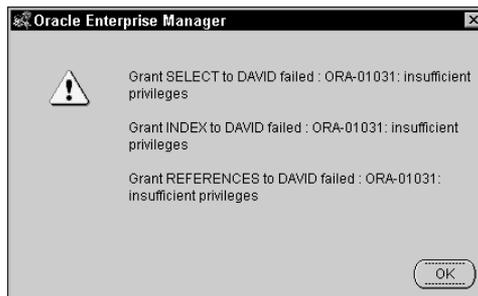


Figure 18-17: If you receive an error similar to this, you do not have sufficient privileges to grant the object privileges to others.

Revoking object privileges

Revoking object privileges also can be performed using Oracle Enterprise Manager or by issuing the REVOKE command. The syntax of the REVOKE command, when dealing with object privileges, is as follows:

```
REVOKE ALL [PRIVILEGES] | object_priv
      [, object_priv, ...]
ON [schema_name.]object_name
FROM user | role | PUBLIC
      [, user | role | PUBLIC, ...]
[CASCADE CONSTRAINTS]
```

As you can see, the syntax is very similar to the GRANT command used to grant object privileges to users. As with the GRANT command, you can revoke more than one privilege, or every privilege that you granted to the user, in a single REVOKE statement. You can also revoke the same set of privileges from more than one user or role at the same time. However, only those individuals that initially granted the privilege can revoke it. Not even the DBA is able to revoke object privileges from users if the DBA did not initially grant them, as shown in this example:

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> REVOKE REFERENCES ON STUDENT.STUDENTS
      2 FROM JOHNS;
REVOKE REFERENCES ON STUDENT.STUDENTS
*
ERROR at line 1:
ORA-01927: cannot REVOKE privileges you did not grant

SQL>
```

The CASCADE CONSTRAINTS clause of the REVOKE command is used to drop any referential integrity constraints that the grantee has defined because of the REFERENCES or ALL privileges being granted to him or her. If you do not include the CASCADE CONSTRAINTS clause and referential integrity constraints exist that would depend on the privileges being revoked, you will get an error and the REVOKE command will fail. Oracle includes the CASCADE CONSTRAINTS clause in the REVOKE command to enable you to remove the permission and any dependent constraints at the same time. However, before doing so you should verify that revoking the privilege and dropping the constraints will not cause other problems in the database—that is, ensure that the privilege can be revoked before doing so.

To revoke a privilege from a user, role, or PUBLIC, connect to the instance as the user that initially granted the privilege and issue the REVOKE command, as in the following examples:

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
SQL> REVOKE SELECT ON Courses FROM PUBLIC;

Revoke succeeded.

SQL> REVOKE ALL ON Students FROM JohnS;

Revoke succeeded.

SQL> REVOKE REFERENCES ON Students FROM MarkS;
REVOKE REFERENCES ON Students FROM MarkS
*
ERROR at line 1:
ORA-01927: cannot REVOKE privileges you did not grant

SQL> REVOKE ALL ON Students FROM MarkS;

Revoke succeeded.

SQL> REVOKE ALL ON Instructors FROM MarkS;

Revoke succeeded.

SQL>
```

As the previous example clearly indicates, if you attempt to remove a privilege that you did not grant the user or the user does not have, Oracle will return the ORA-01927 error. One of the ways to ensure that you are removing all privileges on an object is to use the ALL keyword, as shown in the examples.

An interesting difference in revoking object privileges that were granted WITH GRANT OPTION, as opposed to system privileges granted WITH ADMIN OPTION is that revoking object privileges from users to whom they were granted WITH GRANT OPTION will also cascade to anyone that the user has granted the privilege to. For example, if the user Student were to grant the SELECT privilege on the Instructors table to SYSTEM, and SYSTEM were to grant it to JohnS, revoking it from SYSTEM will also revoke it from JohnS, as shown here:

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
SQL> GRANT SELECT ON Instructors
  2 TO SYSTEM WITH GRANT OPTION;

Grant succeeded.

SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> GRANT SELECT ON Student.Instructors
```

```
2 TO JohnS;
```

Grant succeeded.

```
SQL> connect johns/password@certdb.delphi.bradsys.com
Connected.
SQL> SELECT FirstName, LastName FROM Student.Instructors;
```

FIRSTNAME	LASTNAME
Michael	Harrison
Susan	Keele
David	Ungar
Kyle	Jamieson
Lisa	Cross

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
SQL> REVOKE SELECT ON Instructors
2 FROM SYSTEM;
```

Revoke succeeded.

```
SQL> connect johns/password@certdb.delphi.bradsys.com
Connected.
SQL> SELECT FirstName, LastName FROM Student.Instructors;
SELECT FirstName, LastName FROM Student.Instructors
*

```

```
ERROR at line 1:
ORA-01031: insufficient privileges
```

```
SQL>
```

Revoking object privileges using Oracle Enterprise Manager

You can also revoke object privileges with Oracle Enterprise Manager using the following steps.

STEP BY STEP: Revoking Object Privileges Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager console and connect to the database in which you want to revoke object privileges as a user with appropriate permissions to do so.
2. Expand the database and then Security.

- Expand Users and right-click the user from whom you want to revoke object privileges and select Edit.
- On the Edit User dialog box, click Object Privileges to view a list of schemas in the database, as well as any existing object privileges granted.
- Click the object privilege you want to revoke and then click on the up arrow to remove the privilege, as shown in Figure 18-18.

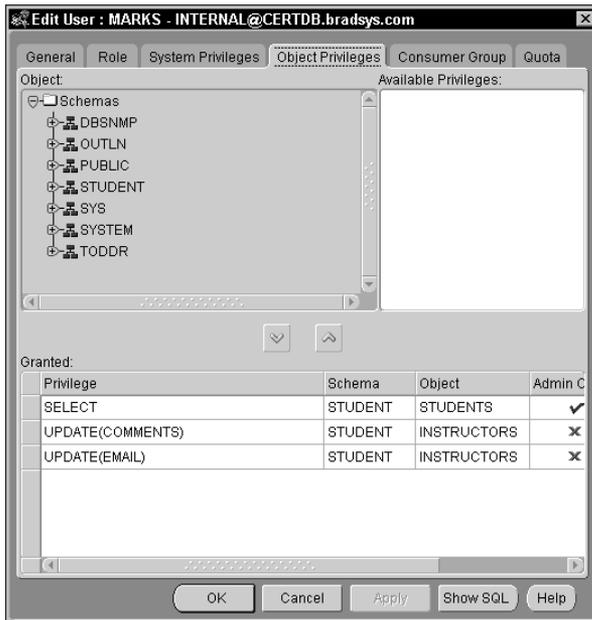


Figure 18-18: Click the privilege and then the up arrow to revoke the privilege.

- When you have revoked all the object privileges from the user, click Apply to save your changes and OK to exit the dialog box.

Getting information on object privileges

If you happen to like graphical user interfaces, Oracle Enterprise Manager provides a nice interface to see which privileges have been granted to a user, or role, or to PUBLIC. You can click and point to each user or role and see what object privileges have been granted, as you have already seen in the previous Step By Step.

Another way to view information on object privileges is to query several data dictionary views, which are outlined in Table 18-3. As a DBA, two of the key views will be `DBA_TAB_PRIVS` and `DBA_COL_PRIVS`, which provide information on privileges granted to users on tables and views, as well as columns in the database. Individual users will make use of the `ALL_` or `USER_` views outlined in Table 18-3, while the DBA should use the `DBA_` views.

Table 18-3
Object Privilege Data Dictionary Views

<i>Data Dictionary View</i>	<i>Description</i>
<code>DBA_TAB_PRIVS</code>	<p>Provides information on all privileges granted to all users, roles, or PUBLIC on all objects in the database. The name may be somewhat misleading, but all object privileges granted are included in the <code>DBA_TAB_PRIVS</code> view, as this query clearly indicates:</p> <pre>SQL> SELECT DISTINCT OBJECT_TYPE FROM 2 DBA_OBJECTS, DBA_TAB_PRIVS 3 WHERE DBA_OBJECTS.OBJECT_NAME= 4* DBA_TAB_PRIVS.TABLE_NAME; OBJECT_TYPE ----- CONSUMER GROUP FUNCTION PACKAGE PACKAGE BODY PROCEDURE SYNONYM TABLE TYPE VIEW 9 rows selected. SQL></pre> <p>This view is only available to users that have been granted the DBA role.</p>

Continued

Table 18-3 (continued)

<i>Data Dictionary View</i>	<i>Description</i>
DBA_COL_PRIVS	<p>Provides information on all privileges granted to users on columns of a table or a view. Only available to users that have been granted the DBA role.</p> <p>Ideally, this view should return zero rows since you should not grant privileges on columns, but create views with only the relevant columns included and then grant privileges on the view.</p>
USER_TAB_PRIVS	<p>Lists all the privileges granted to or granted by the current user on all objects in the database. This provides a complete picture of all privileges that the user has available or has made available to others. It contains information found in both USER_TAB_PRIVS_MADE and USER_TAB_PRIVS_RECD.</p> <p>This view is available to all users in the database and provides session-specific information.</p>
USER_TAB_PRIVS_MADE	<p>Lists all privilege granted to others on object in the current user's schema. This view presents a way for the user to determine which privileges were granted to other users, roles, and PUBLIC on objects in the user's schema.</p>
USER_TAB_PRIVS_RECD	<p>Lists all privileges that have been granted to the current user by other users. It is a way to determine what privileges on objects in other schemas you have.</p>
USER_COL_PRIVS	<p>Lists all privileges granted or received by the current user on columns of tables or views. This view combines the information found by querying the USER_COL_PRIVS_MADE and USER_COL_PRIVS_RECD views.</p>
USER_COL_PRIVS_MADE	<p>Displays information on privileges granted to others by the current user on columns of tables and views in the user's schema.</p>
USER_COL_PRIVS_RECD	<p>Displays information on privileges granted to the current user by others on tables and views in other users' schemas.</p>
ALL_TAB_PRIVS	<p>Lists all privileges granted to or granted by the current user in all schemas that the user has access to.</p> <p>This view is a superset of the USER_TAB_PRIVS view and a subset of the DBA_TAB_PRIVS view. The information presented through this view only applies to the current user.</p>
ALL_TAB_PRIVS_MADE	<p>Displays privileges granted by the current user or by other users that were granted privileges WITH GRANT OPTION to objects in the current user's schema.</p>

<i>Data Dictionary View</i>	<i>Description</i>
	This view is a superset of the USER_TAB_PRIVS_MADE view. The information presented through this view only applies to the current user.
ALL_TAB_PRIVS_RECD	Lists all privileges granted to the current user or PUBLIC. This view is a superset of the USER_TAB_PRIVS_RECD view. The information presented through this view only applies to the current user.

When connected as user Student, to retrieve a list of privileges granted to others on objects in your schema, issue the following query:

```
SQL> connect student/oracle@certdb.delphi.bradsys.com;
Connected.
SQL> col grantee format a10
SQL> col grantor format a10
SQL> col owner format a10
SQL> col table_name format a15
SQL> col privilege format a20
SQL> SELECT * FROM ALL_TAB_PRIVS_MADE;
```

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRA
PUBLIC	STUDENT	COURSES	STUDENT	SELECT	NO
SYSTEM	STUDENT	STUDENTS	STUDENT	ALTER	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	DELETE	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	INDEX	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	INSERT	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	SELECT	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	UPDATE	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	REFERENCES	YES
MARKS	STUDENT	STUDENTS	SYSTEM	SELECT	YES

9 rows selected.

SQL>

If you connect as Marks and want to determine the privileges you have on objects in the database, including columns, issue the following query:

```
SQL> connect marks/oracle@certdb.delphi.bradsys.com
Connected.
SQL> SELECT * FROM USER_TAB_PRIVS;
```

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRA
MARKS	STUDENT	STUDENTS	SYSTEM	SELECT	YES

```
SQL> col column_name format a10
SQL> col privilege format a10
SQL> SELECT * FROM USER_COL_PRIVS;
```

GRANTEE	OWNER	TABLE_NAME	COLUMN_NAM	GRANTOR	PRIVILEGE	GRA
MARKS	STUDENT	INSTRUCTORS	EMAIL	STUDENT	UPDATE	NO
MARKS	STUDENT	INSTRUCTORS	COMMENTS	STUDENT	UPDATE	NO

```
SQL>
```

As the DBA, if you want to determine which privileges were granted by user Student to others for objects in the database, issue the following command:

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> SELECT * FROM DBA_TAB_PRIVS
2 WHERE GRANTOR='STUDENT';
```

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRA
PUBLIC	STUDENT	COURSES	STUDENT	SELECT	NO
SYSTEM	STUDENT	STUDENTS	STUDENT	ALTER	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	DELETE	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	INDEX	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	INSERT	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	SELECT	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	UPDATE	YES
SYSTEM	STUDENT	STUDENTS	STUDENT	REFERENCES	YES

```
8 rows selected.
```

```
SQL>
```

Auditing Privilege Usage

Objective

Identify auditing capabilities

One of the things that organizations may be concerned about is the use (or misuse) of privileges assigned to users. Oracle8i provides a built-in capability to audit the use of system privileges, the execution of classes of SQL commands, and access of database objects by users. While this facility may not provide everything that's needed to create a complete audit trail of user activity, it can be useful in determining if unauthorized access is being attempted or how frequently certain objects are being accessed.

Types of auditing

When auditing operations on your database, it is possible to break down the type of auditing you may want to perform into three categories:

- ♦ **Database auditing**—Database auditing is the monitoring of certain database operations by selected users. With this type of auditing you can track a user's execution of certain statements, such as ALTER USER or CREATE TABLE, or the usage of certain system privileges such as SELECT ANY TABLE, or even the use of object privileges such as EXECUTE for a procedure. This type of auditing is available with the built-in auditing capability of Oracle8i.
- ♦ **Auditing of privileged operations**—Certain operations that are of a sensitive nature should always be recorded. The Oracle server automatically audits the startup and shutdown of an instance, or connections to the instance with SYSDBA, SYSOPER, OSDBA, or OSOPER system privileges, or as the INTERNAL user account. The information that is recorded includes the date and time the operation was performed, the operating system user performing the action, and the name of the machine or a terminal ID (for Unix systems) from which the user performing the action is accessing the instance. The audit mechanism cannot be turned off.

On UNIX systems (Solaris, HP-UX, Linux, and so forth) the privileged operation audit trail for all instances on the server is recorded in a text file with the same name as the operating system process ID of the server process performing the action. These files are located in the path pointed to by \$ORACLE_HOME/rdbms/audit. Oracle will continue to add files to this location as more privileged operations are performed and will not remove any files. It is a good idea to periodically review the files in this location and then delete the ones that you no longer require, or at least back them up and then delete the files from the directory. If you do not do so, Oracle may cause the hard disk where this path is located to become full and may crash the system or have other negative side effects.

On a WindowsNT/2000 computer, Oracle will record privileged operation audit information in the application log of Event Viewer. To prevent the log from filling up, ensure that it is of the appropriate size for your installation and that it is backed up and periodically cleared, or configured to be automatically overwritten.

- ♦ **Application of value-based auditing**—If you need to keep an audit trail of changes made to the database by users, including the values of the data being modified, you need to implement triggers or other methods to track this information. While Oracle's database auditing capability will record the fact that a user modified the value of data in a table, it will not record what the data was changed to or even what row was changed.

The specific requirements for application-level auditing vary by the type of information being kept in the database. For example, an account application may need to keep a complete audit trail of all activity so that any change can be properly tracked by who performed it, what change was made, and what

the data looked like before and after. A different type of application may not have these requirements and would therefore not be well served by database auditing in Oracle. For this reason, and also because of the potential sheer overhead of doing value-based auditing at the database level, Oracle relies upon the application developer and designer to implement application auditing through triggers.

Database auditing guidelines

While the ability to be able to keep track of user actions in the database is a good idea, using it to track everything that users do is not. Like everything else, just because it's there does not mean you have to use it. When determining what level of auditing to implement in your databases, you should follow a number of simple, commonsense guidelines by asking these questions:

- ♦ **Why are you auditing?** The first thing that you should be certain of is the purpose of your auditing. Is it to detect any suspicious database activity? Are you trying to determine what level of activity is taking place in the database? Do you need an historical record of who did what when? Do you want to track the execution of certain statements by a user or users? The first step in ensuring that auditing is necessary is to determine why you are doing it. The reason for auditing has to be clear — doing something for no reason is usually not a good idea.
- ♦ **What are you auditing?** After you have determined why you need to audit database activity, determine how you will gather that information and which elements of database auditing you are going to employ. Will you audit the use of a system privilege by a user, or will you audit the use of a certain class of SQL statements? Will you employ object auditing to track which users access which objects? In answering these questions, always keep your initial purpose in mind and audit only what is needed to satisfy your reason for auditing. Auditing too much activity will potentially cause performance issues, which may result in user dissatisfaction with the database itself.
- ♦ **Will you audit success or failure, or both?** Oracle's database auditing facility enables you to audit the successful execution of a statement, use of a system privilege, or access to an object, as well as failure to perform any of these actions. Is it important to know who succeeded in performing a task, or who failed in doing so? Do you need to know both pieces of information? Again, to answer this question, always keep in mind your purpose for auditing. If you are keeping an historical archive, you may want to audit both success and failure, while if you are tracking unauthorized attempts to access the database, you may only want to audit failure to perform an action.
- ♦ **How much detail do you want in your audit trail?** Oracle enables you to audit by session or by access. *By session* means that if a user performs an audited action once during the session, the audit trail reflects the fact that the action took place the first time it happened. If the user performs the same audited action a second, third, or fourth time (or more), no additional audit entries are created in the audit trail.

Auditing *by access* records an entry in the audit trail each time a user performs the action or accesses the object. This provides the greatest level of detail on user activity but also generates many more records in the audit trail. If you are considering auditing by access, you should also ensure that the audit trail is backed up frequently and its size is monitored to ensure that it does not fill up. If the audit trail fills up, it causes the instance to disallow the execution of audited statements and generates errors.

- ♦ **Who will monitor the audit trail and how often?** The audit trail is generated according to the configuration that you have established. The whole point of auditing is to review database activity, either from a security perspective or from an historical perspective. For this reason, a user, or group of users, will need to review the information in the audit trail on a regular basis to determine if the requirements in the original purpose are being met. The DBA needs to ensure that the information in the audit trail is provided to these individuals to perform their function.
- ♦ **Where will the audit trail be stored?** Oracle enables you to store the audit trail in the database in the SYS.AUD\$ table on all platforms. This table is located on the SYSTEM tablespace, by default, but can be moved by the DBA to a tablespace that contains more disk space and will provide less overhead during database operation. If moving the audit trail, ensure that the tablespace it is being moved to has sufficient disk space to allow for all audit trail records between backups and table truncation. Furthermore, configure a second tablespace to hold the audit trail index. For both tablespaces, it is best to ensure that only the audit trail information is kept on them and that, if configuring automatic growth on the datafiles, a maximum file size is specified so that the audit trail does not consume all disk space.

In order to move the audit trail from the SYSTEM tablespace, follow these steps:

1. If auditing is currently enabled, turn it off before moving the SYS.AUD\$ table.
2. Connect to the instance as the DBA and issue this command to move the table to another tablespace that you have already created:

```
ALTER TABLE SYS.AUD$ MOVE TABLESPACE AUDIT_TBS;
```

3. Re-create the index on the SYS.AUD\$ table that was marked UNUSABLE during the move. Create this index on a tablespace other than the one on which the SYS.AUD\$ table resides by issuing the following command:

```
CREATE INDEX i_aud1 ON SYS.AUD$(SessionID, Serial#)  
TABLESPACE AUDIT_IDX_TBS;
```

4. Reenable auditing for the instance.

Keeping the audit trail in the database provides the greatest level of security as it ensures that anyone that wants to view it will need to connect to the instance before being able to do so. However, Oracle also enables you to store the audit trail at the operating system level. The exact location of the audit trail, when

stored at the OS level, is determined by the operating system being used (for example, on WindowsNT/2000 audit trail records are sent to the event log, whereas on UNIX systems audit records are stored in an OS file on disk).

- ♦ **Who will be responsible for maintaining the audit trail?** The audit trail generated by Oracle — either within the database in the SYS.AUD\$ table, or in the operating system (a disk file on UNIX or the Event Viewer in WindowsNT/2000) — will need to be maintained to ensure that it is properly backed up and will not fill up and thereby prevent users from performing database tasks. The maintenance of the audit trail needs to be performed by either a system administrator (in the case where the audit trail is stored at the OS level) or a DBA, if the audit trail is stored in the database.

If storing the audit trail in the database, back up and then TRUNCATE the audit trail to ensure that it does not grow too large; if storing the audit trail at the operating system, back up the file or event log and then delete the file or clear the log. Oracle will re-create the operating system file of WindowsNT/2000 event log the next time it needs to audit a database operation. Proper maintenance ensures that the system will continue to function and that a proper historical record exists, if needed.

- ♦ **How will you protect the audit trail?** If you are auditing to the database, how will you ensure that users that have access to the SYS.AUD\$ table (that is, DBAs and those who have been granted the DELETE_CATALOG_ROLE) do not manually delete rows from the table to eliminate any trace of unauthorized activity that they may have performed? If performing OS auditing, how will you ensure that the operating system audit trail is only visible to those users who should have access to it? These questions are crucial to maintain the validity of your auditing operation. Some organizations choose someone other than the DBA to configure and monitor auditing in order to keep the DBA honest.

If you are auditing to the database, ensure that only the DBA has the DELETE_CATALOG_ROLE as the first step in preventing other users from being able to manipulate the audit table. The next step is to ensure that any manual activity on the audit table is itself audited. Issue the following command to audit DELETE operations by all users on the audit trail itself:

```
AUDIT DELETE ON SYS.AUD$ BY ACCESS;
```

If you are storing your audit trail at the operating system level, work with your system administrators to come up with a viable strategy to protect the audit trail.

Implementing database auditing

Now that you have decided upon your reasons for auditing, determined what to audit and how often, and drawn up a plan to monitor and maintain the audit trail, you can configure auditing for your database. In order to do so, you need to both configure the instance to support auditing and enable auditing within the database.

Enabling auditing for the instance

When an Oracle instance starts up, it determines whether to audit database activity based upon the value of the `AUDIT_TRAIL` Oracle initialization parameter. This parameter can have several possible values, including `FALSE`, `NONE`, `TRUE`, `DB`, or `OS`.

The default value for `AUDIT_TRAIL` is `NONE` (or `FALSE`). When set to `NONE`, or `FALSE`, Oracle doesn't perform any auditing of database activity even if it is configured. This ensures that databases aren't burdened with the overhead of auditing if it is not required (a good thing since the vast majority of Oracle databases do not have any auditing configured).

Leaving the value of `AUDIT_TRAIL` set to `NONE` or `FALSE` also enables you to configure auditing for the database but not enable it if not required. This benefits third-party application developers that use the Oracle database in that their software can now support database auditing but not require it to be enabled unless the end-user organization deems it necessary.

Setting the value of the `AUDIT_TRAIL` `INIT.ORA` parameter to `DB` or `TRUE` will enable auditing for the instance and store the audit trail records in the `SYS.AUD$` table. Prior to enabling auditing, ensure that you have moved the audit trail table to a tablespace that has sufficient disk space for the amount of activity you have configured and anticipate. Also ensure that the `i_aud1` index has been created.

The last value that you can assign to the `AUDIT_TRAIL` initialization parameter is `OS`. This enables auditing for the instance and sends the audit trail records to the appropriate operating system location. Because the location differs for each operating system that Oracle runs on, you should always refer to your operating system-specific documentation for information on where the audit trail will be located. On a WindowsNT/2000 platform, it is stored in the event log, while on a UNIX system the audit trail is stored in the path pointed to by the `AUDIT_FILE_DEST` Oracle initialization parameter.

To enable auditing for the instance and store the audit trail in the database, ensure that this line appears in your `INIT.ORA` file:

```
AUDIT_TRAIL = DB
```

Once you have enabled auditing for the instance, Oracle examines each statement executed to determine if it should be audited for the user performing the action. If so, it writes a record to the audit trail configured; if not, it processes the statement normally. If you submit a PL/SQL block or execute a stored procedure, Oracle examines each line of code of the PL/SQL block or stored procedure as it is being executed to determine if it should be audited and then performs the appropriate action. If a user performs actions that you have configured for auditing and then rolls back the transaction, the rollback will not affect the audit information. Oracle

records the audit trail independent of the transaction that caused the auditing to occur. However, if the statement to be executed produces a syntax error, Oracle will not generate an audit record. This is because auditing takes place during the execute phase of statement processing, whereas syntax checking takes place prior to that in the parse phase.

Enabling auditing of database operations

Once you have determined that you will perform database auditing, you need to determine what you will audit. Oracle enables you to audit three types of operations: statement execution, system privilege use, and object access. No matter what type of auditing you use, the command to enable it is the AUDIT command. You cannot, at this time, configure auditing using Oracle Enterprise Manager or any other Oracle-supplied graphical tool. You need to issue the AUDIT command from SQL*Plus, or another interactive tool, to do so.

Auditing statement execution

Statement auditing enables you to audit the execution of classes or statements. For example, you can audit all statements that deal with the creation, modification, or dropping of users by auditing all USER statements, or the creation, modification, truncation, or dropping of a table by auditing all TABLE statements, and so on. Changes to statement auditing options do not take effect until the user disconnects and reconnects to the instance. In other words, enabling statement auditing for a user that is connected to the database will not generate audit records until the next time the user connects to the instance.

The statement classes that can be audited and the actual SQL commands that will be tracked when statement auditing is enabled are outlined in Table 18-4.



While the list of statements that can be audited with statement auditing is quite extensive, you should keep in mind that generally statement auditing will deal with DDL statements for a particular class of database object, such as TABLE, VIEW, or USER. Knowing this will make your preparation for the exam easier.

Table 18-4
Statement Auditing Command Mapping

<i>Statement Audit Option</i>	<i>SQL Command Execution Audited</i>
CLUSTER	CREATE CLUSTER
	ALTER CLUSTER
	DROP CLUSTER
	TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT
	DROP CONTEXT

Statement Audit Option	SQL Command Execution Audited
DATABASE LINK	CREATE DATABASE LINK DROP DATABASE LINK
DIMENSION	CREATE DIMENSION ALTER DIMENSION DROP DIMENSION
DIRECTORY	CREATE DIRECTORY DROP DIRECTORY
INDEX	CREATE INDEX ALTER INDEX DROP INDEX
NOT EXISTS	Any SQL statement that does not return a result because the object being queried or manipulated does not exist.
PROCEDURE (This applies to both PL/SQL and Java types of the associated object.)	CREATE FUNCTION DROP FUNCTION CREATE PROCEDURE DROP PROCEDURE CREATE PACKAGE DROP PACKAGE CREATE PACKAGE BODY DROP PACKAGE BODY CREATE LIBRARY DROP LIBRARY
PROFILE	CREATE PROFILE ALTER PROFILE DROP PROFILE
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM

Continued

Table 18-4 (continued)

<i>Statement Audit Option</i>	<i>SQL Command Execution Audited</i>
ROLE	CREATE ROLE ALTER ROLE DROP ROLE
ROLLBACK SEGMENT	CREATE ROLLBACK SEGMENT ALTER ROLLBACK SEGMENT DROP ROLLBACK SEGMENT
SEQUENCE	CREATE SEQUENCE DROP SEQUENCE
SESSION	Connections to the instance by users (CONNECT command)
SYNONYM	CREATE SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT (of system privileges only) NOAUDIT (of system privileges only)
SYSTEM GRANT	GRANT (of system privileges only) REVOKE (of system privileges only)
TABLE	CREATE TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE
TRIGGER	CREATE TRIGGER ALTER TRIGGER (ENABLE or DISABLE) DROP TRIGGER ALTER TABLE (ENABLE or DISABLE ALL TRIGGERS)
TYPE	CREATE TYPE ALTER TYPE DROP TYPE CREATE TYPE BODY DROP TYPE BODY

<i>Statement Audit Option</i>	<i>SQL Command Execution Audited</i>
USER	CREATE USER
	ALTER USER
	DROP USER
VIEW	CREATE VIEW
	DROP VIEW

The syntax of the AUDIT command for statement auditing is

```
AUDIT statement [, statement, ...]
[BY username [,username, ...]]
[BY SESSION | ACCESS]
[WHENEVER [NOT] SUCCESSFUL]
```

To audit success or failure of all DDL statements associated with creating tables, views, or procedures by JohnS and MarkS, issue the following command:

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> AUDIT TABLE, VIEW, PROCEDURE
2 BY MarkS, JohnS;
```

Audit succeeded.

SQL>

To audit all user operations, including users changing their passwords, issue the following command:

```
SQL> AUDIT USER;
```

Audit succeeded.

SQL>

System privilege usage auditing

Privilege auditing enables you to track the use of system privileges assigned to users or attempted by users who do not have the privilege. Any system privilege that you can assign to a user can also be audited, including CREATE TABLE, SELECT ANY TABLE, ALTER USER, CREATE SESSION, and so on. Like statement auditing, changes to system privilege auditing options do not take effect until the user disconnects and reconnects to the instance. In other words, enabling system privilege auditing for a user that is connected to the database will not generate audit records until the next time the user connects to the instance.

The syntax of the AUDIT command for system privilege auditing is

```
AUDIT system_priv [,system_priv, ...]
  [BY username [,username, ...]]
  [BY SESSION | ACCESS]
  [WHENEVER [NOT] SUCCESSFUL]
```

In the auditing of system privileges, Oracle will first ensure that the privilege being used (for example, SELECT ANY TABLE) is actually being used and that the user has not been given explicit privileges by the owner of any object to perform the action. In this way, Oracle is able to determine whether an audit record should be added to the audit trail. For example, if BobW queries the Students table in the Student schema, Oracle ensures that BobW has not been granted the SELECT privilege on the table directly and is in fact making use of the SELECT ANY TABLE privilege, which should be audited. In this way, Oracle does not generate unnecessary audit trail records for use of privileges granted explicitly to the user or a role the user has been granted.

To record an entry in the audit trail whenever any user that has been granted the SELECT ANY TABLE privilege uses it, issue the following command:

```
SQL> AUDIT SELECT ANY TABLE BY ACCESS;

Audit succeeded.

SQL>
```

If you want to audit any unsuccessful attempts by users to create tables in their schemas, configure auditing as follows:

```
SQL> AUDIT CREATE TABLE
  2  WHENEVER NOT SUCCESSFUL;

Audit succeeded.

SQL>
```

If you also want to know about the first time a user modified their session settings, whether or not they were successful, issue the following command to configure auditing of the system privilege:

```
SQL> AUDIT ALTER SESSION
  2  BY SESSION;

Audit succeeded.

SQL>
```

Auditing object access

Object auditing enables the DBA to track access to database objects by users. This can be used to determine which users are accessing which objects legitimately, or

which users are attempting to access database objects they do not have permission to. This type of auditing can potentially generate the most activity and should be carefully planned.

The syntax of the `AUDIT` command for object auditing is as follows:

```
AUDIT ALL | command [, command, ...]
      ON [schema.]object | DEFAULT
      [BY SESSION | ACCESS]
      [WHENEVER [NOT] SUCCESSFUL]
```

If you want to enable auditing for all objects in the database from this point forward, you can enable auditing `ON DEFAULT`. When you specify `ON DEFAULT`, Oracle will apply the auditing options you have specified to all objects that are created in the database from this point forward. Existing objects will not have their auditing configuration changed, but new database objects, created by any user with the appropriate privilege, will automatically have any auditing options specified `ON DEFAULT` applied to them, assuming the auditing option makes sense (that is, auditing of `EXECUTE` statements on tables will not take place since it does not make sense).

Unlike statement and privilege auditing, which is audited `BY ACCESS` where the statement or system privilege action would create an audit trail record each time the audited event took place, object auditing defaults to `BY SESSION`, which means that an audit trail record is only created the first time the user's session executes the audited command on the object. The reason for this change is primarily to reduce the amount of information in the audit trail as auditing `UPDATE` commands on a table could generate a lot of information if each time the `UPDATE` was issued an audit trail record was generated. The thinking here is that knowing that JohnS performed an `UPDATE` on the `Courses` table is sufficient information, especially if he should not have been doing it in the first place.

Unlike statement and privilege auditing where you can specify a user whose actions to audit, auditing will take place for all users accessing the object. By default, both success and failure of object access will be audited, though you can specify one or the other, depending on the purpose of auditing in your database. Object auditing, also unlike statement and privilege auditing, takes effect immediately after auditing is configured. This means that once you enable auditing on a table or a view, the next time a user accesses that table or view and triggers the auditing action, the audit trail will be updated. There is no need for users to disconnect and reconnect to the instance in order to begin the audit operation.

The types of commands (operations) that can be audited depend on the object for which you are configuring auditing. In general, the actions you can audit on an object are similar to the permissions you can grant. This means that it does not make sense to audit a `SELECT` statement on a procedure or an `EXECUTE` operation on a table. The operations that can be audited on the various objects in the database are outlined in Table 18-5.

Table 18-5
Object Operations That Can Be Audited

<i>Operation</i>	<i>TABLE</i>	<i>VIEW</i>	<i>SEQUENCE</i>	<i>PROGRAM UNIT</i>
SELECT	X	X	X	
INSERT	X	X		
UPDATE	X	X		
DELETE	X	X		
ALTER	X		X	
EXECUTE				X
INDEX	X			
GRANT	X	X	X	X
LOCK	X	X		
RENAME	X	X		X
AUDIT	X	X	X	X
COMMENT	X	X		

It is important to note that, unlike the granting of object privileges, the DBA is the only one that can configure auditing for all objects in the database. The owner of an object cannot, unless explicitly granted privileges to do so, configure auditing on his or her own object. This is because auditing deals with tracking usage and security breach attempts on the database as a whole and, as such, becomes the responsibility of the DBA.

To track all DELETE operations on the ClassEnrollment table, issue the following command:

```
SQL> AUDIT DELETE ON Student.ClassEnrollment
      2 BY ACCESS;
```

Audit succeeded.

```
SQL>
```

To audit any unsuccessful attempts to update the Students table, issue the following command:

```
SQL> AUDIT UPDATE ON Student.Students
      2 BY ACCESS
```

```
3 WHENEVER NOT SUCCESSFUL;
```

```
Audit succeeded.
```

```
SQL>
```

To audit all operations on the `Instructors` table in the `Student` schema whenever they occur, issue the following command:

```
SQL> AUDIT ALL ON Student.Instructors
2 BY ACCESS;
```

```
Audit succeeded.
```

```
SQL>
```

To ensure that any `LOCK`, `ALTER`, `AUDIT`, or `RENAME` operations that take place on any objects created from this point on are audited for each user session, configure auditing as follows:

```
SQL> AUDIT LOCK, AUDIT, ALTER, RENAME
2 ON DEFAULT
3 BY SESSION;
```

```
Audit succeeded.
```

```
SQL>
```

To ensure that any `DELETE` or `GRANT` operations that take place on any objects created from this point on are audited each time they occur, configure auditing as follows:

```
SQL> AUDIT DELETE, GRANT ON DEFAULT
2 BY ACCESS;
```

```
Audit succeeded.
```

```
SQL>
```

Disabling database auditing

If the `AUDIT` command is used to enable auditing, then — you guessed it — the `NOAUDIT` command is used to disable auditing. The syntax of the `NOAUDIT` command corresponds exactly to the syntax of the `AUDIT` command for the exact same type of auditing.

To disable auditing of any `PROCEDURE` statements by `MarkS` and `JohnS`, issue the following command:

```
SQL> NOAUDIT PROCEDURE
      2 BY MarkS, JohnS;
```

Noaudit succeeded.

```
SQL>
```

To disable auditing of the CREATE TABLE or SELECT ANY TABLE system privilege by the user SYSTEM, execute the following command:

```
SQL> NOAUDIT CREATE TABLE, SELECT ANY TABLE
      2 BY SYSTEM;
```

Noaudit succeeded.

```
SQL>
```

In this last example, note that prior to this point an explicit AUDIT command directed at the SYSTEM user was never issued. However, auditing of the CREATE TABLE and SELECT ANY TABLE system privileges was configured to apply to all users in the database. Disabling auditing for the user SYSTEM leaves auditing intact for everyone else, but turns it off for this one user.

To disable auditing of RENAME operations on all objects created from this point on, issue the following command:

```
SQL> NOAUDIT RENAME ON DEFAULT;
```

Noaudit succeeded.

```
SQL>
```

Getting information about auditing

When retrieving information about auditing, you can retrieve two kinds of information — the auditing configuration for the database or the audit trail information, when the audit trail is stored in the database. To accomplish either, you will need to query data dictionary views.

Viewing database audit options

In order to determine what auditing is currently configured in the database, you can query one of several data dictionary views. In order to query these views, you must be connected to the instance as a user who has been assigned the DBA role.

The views that are available include the ALL_DEF_AUDIT_OPTS view, which returns all auditing that has been configured ON DEFAULT for the database; the DBA_STMT_AUDIT_OPTS view, which returns all statement auditing options

configured for the database; the `DBA_PRIV_AUDIT_OPTS` view, for all system privilege auditing options currently enabled; and `DBA_OBJ_AUDIT_OPTS`, for the object auditing enabled for the database.

In order to determine which actions will be audited by default for any object created in the database from this point on, query `ALL_DEF_AUDIT_OPTS` as follows:

```
SQL> SELECT * FROM ALL_DEF_AUDIT_OPTS;

ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE
--- --- --- --- --- --- --- --- --- --- --- ---
S/S S/S -/- A/A A/A -/- -/- S/S -/- -/- -/- -/- -/-
SQL>
```

The output from this view includes a kind of spreadsheet layout with the different operations as column headings and a row for each column indicating whether auditing will be by session or access, and whether to audit success or failure, or both. The headings of the columns, from left to right, indicate the operations that can be performed and are as follows: ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE, REFERENCE, and EXECUTE.

Under each column heading, the “-/-” also has a special meaning. The left side of the forward slash (/) indicates whether or not success will be audited, while the right side indicates whether failure will be audited. If the left or right side contains the letter “S,” then auditing for that column’s operation will be done BY SESSION whenever the operation is successful or not successful, respectively. If the left or right side contains the letter “A,” then auditing for the column’s operation will be done BY ACCESS whenever the operation is successful or not successful, respectively. If a dash “-” appears on either side of the forward slash, auditing for success or failure for that operation is not configured.

To bring this into perspective, reading the output presented previously you can deduce that the default audit options for this database are to audit ALTER, AUDIT, and LOCK statements by session whenever successful or not successful, as well as to audit DELETE and GRANT statements by access whenever successful or not successful. If you go back and review the examples of default auditing presented previously, you will note that the output correctly reflects what was configured.

To determine what statement auditing is configured for the database, query the `DBA_STMT_AUDIT_OPTS` view, as follows:

```
SQL> col user_name format a10
SQL> col audit_option format a20
SQL> SELECT USER_NAME, AUDIT_OPTION, SUCCESS, FAILURE
       2 FROM DBA_STMT_AUDIT_OPTS;

USER_NAME  AUDIT_OPTION          SUCCESS  FAILURE
-----
```

JOHNS	TABLE	BY ACCESS	BY ACCESS
JOHNS	VIEW	BY ACCESS	BY ACCESS
MARKS	TABLE	BY ACCESS	BY ACCESS
MARKS	VIEW	BY ACCESS	BY ACCESS
	USER	BY ACCESS	BY ACCESS
	SELECT ANY TABLE	BY ACCESS	BY ACCESS
	CREATE TABLE	NOT SET	BY ACCESS
	ALTER SESSION	BY ACCESS	BY ACCESS

8 rows selected.

SQL>

You will note that the above-presented output is a little easier to read than the default audit options query results, but some explanation is still warranted. First, if the `USER_NAME` column does not have a value, it means that the statements being audited will be audited for all users. Second, the output for this query will also return system privilege auditing information in the `AUDIT_OPTION` column that can be queried using the `DBA_PRIV_AUDIT_OPTS`. The reason for this is that both statement auditing and system privilege auditing deal with the use of privileges on the database and are considered two sides of the same coin by Oracle. Third, the `SUCCESS` and `FAILURE` columns will indicate whether auditing of the statement will occur `BY ACCESS` or `BY SESSION`, or not at all (`NOT SET`).

To determine which system privilege usage will trigger auditing in the database, execute the following query:

```
SQL> col privilege format a25
SQL> SELECT USER_NAME, PRIVILEGE, SUCCESS, FAILURE
       2 FROM DBA_PRIV_AUDIT_OPTS;
```

USER_NAME	PRIVILEGE	SUCCESS	FAILURE
	ALTER SESSION	BY ACCESS	BY ACCESS
	CREATE TABLE	NOT SET	BY ACCESS
	SELECT ANY TABLE	BY ACCESS	BY ACCESS

SQL>

The output from this view is similar to that of `DBA_STMT_AUDIT_OPTS`, except that the `PRIVILEGE` column indicates which system privileges will be audited, and statement auditing options are not included, as in the previous output. The meaning of values in the `USER_NAME` column is the same— if no username is specified, the auditing will apply to all users; otherwise, it will apply just to the user with the specified name. The `SUCCESS` and `FAILURE` columns work the same way as with `DBA_STMT_AUDIT_OPTS`.

To determine which schema object auditing options are configured in the database, issue the following query:

```

SQL> col object_type noprint
SQL> col owner noprint
SQL> col object_name format a16
SQL> SELECT * FROM DBA_OBJ_AUDIT_OPTS
      2 WHERE OWNER='STUDENT' AND OBJECT_TYPE='TABLE';

```

OBJECT_NAME	ALT	AUD	COM	DEL	GRA	IND	INS	LOC	REN	SEL	UPD	REF	EXE	CRE	REA	WRI
BATCHJOBS	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
CLASSENROLLMENT	-/-	-/-	-/-	A/A	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
COURSEAUDIT	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
COURSES	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
INSTRUCTORS	A/A	-/-	-/-	-/-	-/-	-/-										
LOCATIONS	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
SCHEDULEDCLASSES	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
STUDENTS	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-

8 rows selected.

```

SQL>

```

As you can see, the output is similar to that of ALL_DEF_AUDIT_OPTS. The format for the columns indicating whether auditing is enabled, and whether it is by session or by access, and whether for success or failure, is very similar. The column names, when viewed from left to right, correspond to the following (starting with ALT): ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE, REFERENCE, EXECUTE, CREATE, READ, and WRITE.

You should note that if you do a query to count the number of rows in the DBA_OBJ_AUDIT_OPTS view, you will come up with a rather large number, as shown here:

```

SQL> SELECT COUNT(*) FROM DBA_OBJ_AUDIT_OPTS;

```

COUNT(*)
1732

```

SQL>

```

Oracle inserts a row into the view for each object in the database, whether or not auditing is enabled or configured. For this reason, it is not a good idea to query the entire set of rows in the view, but rather to always apply a WHERE clause to retrieve only those rows that you are interested in seeing.

Viewing the database audit trail

As you have previously been shown, you can enable auditing to the database or to the operating system audit trail. If you enabled auditing to the operating system, you need to use operating system tools to view the information in the audit trail. However, if the INIT.ORA file contains the setting “AUDIT_TRAIL=DB”, you can use one of several views to look at the entries in the audit trail.

If you want to get a listing of all entries in the audit trail, with just some key pieces of information, execute the following query:

```
SQL> col username format a10
SQL> col owner format a10
SQL> col action_name format a20
SQL> col obj_name format a15
SQL> SELECT USERNAME, TIMESTAMP, OWNER, OBJ_NAME, ACTION_NAME, RETURNCODE
       2 FROM DBA_AUDIT_TRAIL ORDER BY TIMESTAMP;
```

USERNAME	TIMESTAMP	OWNER	OBJ_NAME	ACTION_NAME	RETURNCODE
SYSTEM	08-JUL-01	STUDENT	INSTRUCTORS	AUDIT OBJECT	0
JOHNS	08-JUL-01	STUDENT	INSTRUCTORS	SELECT	2004
STUDENT	08-JUL-01	STUDENT	INSTRUCTORS	SELECT	0
SYSTEM	08-JUL-01	SYS	OBJ\$	SELECT	0
SYSTEM	08-JUL-01		JOHNS	ALTER USER	0
SYSTEM	08-JUL-01		JOHNS	ALTER USER	922
SYSTEM	08-JUL-01		JOHNS	ALTER USER	0
JOHNS	08-JUL-01	JOHNS	MYSTUDENTS	CREATE TABLE	933
JOHNS	08-JUL-01	JOHNS	MYSTUDENTS	CREATE TABLE	1536
JOHNS	08-JUL-01	JOHNS	MYSTUDENTS	CREATE TABLE	0
MARKS	08-JUL-01	MARKS	MYSTUD	CREATE TABLE	1950

11 rows selected.

SQL>

The value of the RETURNCODE column in the results of the query indicates the Oracle error code that was returned by the operation. A return code of zero indicates success, while any other value is the Oracle error that caused the failure. If you audit both success and failure, both a zero and nonzero set of return codes will appear in the audit trail.

To get a listing of statement audit records in the audit trail, execute a query similar to the following:

```
SQL> SELECT USERNAME, TIMESTAMP, OWNER, OBJ_NAME, RETURNCODE, PRIV_USED
       2 FROM DBA_AUDIT_STATEMENT;
```

USERNAME	TIMESTAMP	OWNER	OBJ_NAME	RETURNCODE	PRIV_USED
SYSTEM	08-JUL-01	STUDENT	INSTRUCTORS	0	AUDIT ANY

SQL>

Key Point Summary

In preparing for the Oracle8i DBA: Architecture and Administration exam, please keep in mind these points regarding privileges and auditing:

- ♦ System privileges enable you to perform a database action such as create a table or drop an index, while object privileges enable you to manipulate database objects.
- ♦ System privileges are usually granted to users and roles by the DBA. Object privileges are usually granted by the owner of the object.
- ♦ The GRANT command is used to grant privileges, and the REVOKE command is used to REVOKE privileges.
- ♦ The DBA can grant system privileges WITH ADMIN OPTION, which enables the person granted the privilege to grant it to others.
- ♦ An object owner can grant object privileges WITH GRANT OPTION, which enables the person granted the privilege to grant it to others.
- ♦ When revoking a system privilege granted WITH ADMIN OPTION, the revoke will not cascade to users that the person from whom the privilege is being revoked granted it to — that is, the revoke will not cascade.
- ♦ When revoking an object privilege granted WITH GRANT OPTION, the REVOKE will also revoke the privilege from any other person to whom the privilege was granted — that is, the revoke will cascade.
- ♦ SYSDBA and SYSOPER are special system privileges that enable the holder to start up and shut down the instance, perform database backup and restore operations, and perform other privileged tasks.
- ♦ The INIT.ORA parameter REMOTE_LOGIN_PASSWORD_FILE, when set to EXCLUSIVE will enable password file authentication for SYSDBA and SYSOPER privileges.
- ♦ Oracle supports three types of auditing: privileged operation auditing, database auditing, and value-based auditing.
- ♦ Auditing of privileged operations, such as startup and shutdown, or connecting as a user with SYSDBA privileges, is automatic and cannot be disabled. The audit trail is stored in the \$ORACLE_HOME/rdbms/audit directory on a UNIX system, or the application event log on WindowsNT/2000.

- ♦ Value-based, or application auditing, can be implemented by the developer or DBA through the use of triggers.
- ♦ Database auditing is enabled by setting the `AUDIT_TRAIL` Oracle initialization parameter to the location of the audit trail—either the database or an operating system location. When the value of the `AUDIT_TRAIL` parameter is set to `DB` or `TRUE`, the audit records will be stored in the `SYS.AUD$` table in the database. When the value of the `AUDIT_TRAIL` parameter is set to `OS`, the audit trail will be located in the operating system–specific location. The default for `AUDIT_TRAIL` is `NONE`, or `FALSE`, which means database auditing is disabled.
- ♦ The `AUDIT` and `NOAUDIT` commands enable and disable database auditing.
- ♦ Using database auditing you can audit statements executed by users, system privileges used, or access to database objects.
- ♦ Both statement and system privilege auditing will add an audit record for each occurrence of the audited operation—that is, they default to `BY ACCESS`. Object auditing will only create a single entry in the audit trail for each user session by default—that is, object auditing defaults to `BY SESSION`.
- ♦ Unless otherwise specified, both success and failure will be audited for all types of database auditing.
- ♦ Only the DBA can configure auditing, as well as view the database audit trail.



STUDY GUIDE

In preparation for writing the “Oracle8i: Architecture and Administration” and to generally improve and test your understanding of the assignment of privileges and auditing, make use of the questions, scenarios, and labs in this part of the chapter. Answer all questions, perform the labs, and work through the scenarios. Check your work and use the answers provided to determine where you may need to spend more time before taking the exam.

Assessment Questions

1. You attempt to grant the SYSDBA privilege to a user and receive the following error message:

```
ORA-01994: GRANT failed: cannot add users to public password file
```

What is the most likely reason that the grant failed? (Choose the best answer.)

- A. You do not have permissions to perform the operation.
 - B. The parameter `PASSWORD_OS_AUTHENT=TRUE` is not set.
 - C. The parameter `REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE` is not set.
 - D. You are using operating system authentication and cannot add users to a password file.
 - E. SYSDBA is not supported on Windows NT 4.0.
2. You want to grant the SELECT privilege on your Orders table to BobW and JohnS. You also want BobW to be able to grant the SELECT privilege to other users in the database. Which GRANT command(s) would you issue? (Choose all correct answers.)
 - A. `GRANT SELECT ON Orders TO BobW;`
 - B. `GRANT SELECT ON Orders TO JohnS;`
 - C. `GRANT SELECT ON Orders TO BobW WITH GRANT OPTION;`
 - D. `GRANT SELECT ON Orders TO JohnS WITH GRANT OPTION;`
 - E. `GRANT SELECT ON Orders TO BobW WITH ADMIN OPTION;`
 - F. `GRANT SELECT ON Orders TO JohnS WITH ADMIN OPTION;`
 - G. `GRANT SELECT ON Orders TO BobW, JohnS WITH GRANT OPTION;`
 - H. `GRANT SELECT ON Orders TO BobW, JohnS WITH ADMIN OPTION;`

3. While passing by Todd's desk, you notice that he is using SQL*Plus to query data in your TempOrders table. You did not grant Todd privileges to issue SELECT statements against your TempOrders table. Why is Todd able to query your table? (Choose all correct answers.)
- A. A user to whom you granted the SELECT privilege on the table also granted it to Todd.
 - B. Todd is a DBA and can query any table in the database.
 - C. You granted Todd the UPDATE privilege on the TempOrders table, which automatically grants the SELECT privilege.
 - D. Todd has been granted the SELECT ANY TABLE privilege by the DBA.
 - E. Todd has been granted the SELECT privilege on your TempOrders table by a user to whom you granted the SELECT privilege WITH ADMIN OPTION.
4. In order to create a table in their own schema in the database, which of the following must a user have? (Choose two correct answers.)
- A. The CREATE ANY TABLE privilege
 - B. A quota on their default tablespace
 - C. A quota on a tablespace
 - D. The CREATE TABLE privilege
 - E. The SYSDBA privilege
5. In order to perform recovery of the database up to a particular point in time, which of the following privileges are required? (Choose the best answer.)
- A. RESTORE DATABASE
 - B. INTERNAL
 - C. SYSOPER
 - D. SYSDBA
 - E. OSOPER
6. Which of the following queries will provide you with a listing of all system privileges that are currently active in your session? (Choose the best answer.)
- A. SELECT * FROM SESSION_PRIVS
 - B. SELECT * FROM DBA_SYS_PRIVS
 - C. SELECT * FROM MY_SESS_PRIVS
 - D. SELECT * FROM USER_SYS_PRIVS
 - E. SELECT * FROM ALL_SYS_PRIVS

7. On a Windows NT server running Oracle 8.1.7, if the `AUDIT_TRAIL` parameter for a database is set to `OS`, where will the audit trail be located? (Choose the best answer.)
- A. In the `SYS.AUD$` table.
 - B. In the path pointed to by `AUDIT_FILE_DEST`.
 - C. In the `$ORACLE_HOME/rdbms/audit` location.
 - D. In the application event log for the server.
 - E. You cannot configure `AUDIT_TRAIL=OS` on Windows NT.
8. How can you configure auditing of `UPDATE` statements on the `APP.Orders` table by the user `SteveG` on each occurrence? (Choose the best answer.)
- A. `AUDIT UPDATE ON App.Orders BY SteveG;`
 - B. `AUDIT UPDATE ON App.Orders BY SteveG WHENEVER SUCCESSFUL;`
 - C. `AUDIT UPDATE ON App.Orders BY SteveG BY ACCESS;`
 - D. `NOAUDIT SELECT, INSERT, DELETE ON App.Orders BY SteveG BY ACCESS;`
 - E. You cannot audit object access by individual users.
9. Which information can be found in the database audit trail when auditing all operations on tables? (Choose all correct answers.)
- A. The operating system username of the person performing the action
 - B. The old value of the row before the update
 - C. The new value of the row after the update
 - D. The date and time the change occurred
 - E. The application used to perform the change
 - F. The database username of the individual performing the action
10. By default, who can configure value-based auditing? (Choose the best answer.)
- A. The DBA
 - B. Any user with `SYSDBA` privileges
 - C. The object owner
 - D. Any user with `SELECT` privileges on the table
 - E. `PUBLIC`

Scenarios

1. You have been tasked with ensuring that the following requirements are met in a production database where most objects accessed by users are located in the APP schema:
 - Ensure that all users can query any table in the APP schema as well as execute any procedure, package, or function in the APP schema.
 - Allow two junior DBAs (ToddR and YuryS) to be able to change users' passwords, quotas, and other settings. YuryS should also be allowed to create new users.
 - Allow your three main developers (SusanI, SteveG, and MylesB), as well as the junior DBAs mentioned previously, to create tables, views, procedures, and indexes in the APP schema. They should not be able to drop or truncate any existing tables or other objects in the APP schema.
 - Disallow all users from inserting, updating, or deleting data in tables and views in the APP schema, except through stored procedures and packages.

How would you configure and assign the appropriate permissions to satisfy these requirements?

2. You have been tasked with implementing a corporate-wide audit policy on your existing database and ensuring that it is also implemented on all databases created from this point on. The main points of the policy are as follows:
 - All logon and logoff attempts will need to be recorded in the audit trail.
 - The audit trail will need to have any direct manipulation of it recorded.
 - Any attempts to delete data in the database will need to be recorded each time they occur.
 - Any changes to the data in the database will need to have the old and new values recorded, as well as information on who performed the change and when.
 - The creation, modification, or dropping of database objects will need to be recorded in the audit trail whenever they occur.
 - Queries on all user-created database objects will need to be recorded in the audit trail the first time they occur during a user's session.

How would you configure auditing for your existing as well as new databases to satisfy these requirements?

Lab Exercises

Lab 18-1 Granting and Revoking System and Object Privileges

1. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a new user called ToddR with a password of your choice, and grant ToddR the ability to connect to the instance, as well as a quota of 10MB on the CERTDB tablespace. Make the CERTDB tablespace the default tablespace for the user.
3. Grant ToddR the ability to create tables, views, and indexes, as well as the ability to grant the CREATE VIEW privilege to others.
4. Connect to the instance as ToddR and attempt to create a table called InstructTemp with the same structure and data as the Instructors table in the Student schema. What happens and why?
5. Connect to the instance as user Student and grant ToddR the appropriate privileges to be able to create the table in step 4. Allow ToddR to grant the privileges to other users.
6. Connect to the instance as ToddR and attempt to create the table in step 4 again. What happens and why?
7. Connect to the instance as user SYSTEM and create a user called YuryS with a password of your choice, and grant YuryS the ability to connect to the instance, as well as to query data in any table in the database.
8. Connect to the instance as YuryS and query the data in the Student.Students table. Attempt to create a table with the same structure as the Students table in the Student schema. What happens and why?
9. While still connected as YuryS, attempt to grant ToddR the ability to query the Student.Students table. What happens and why?
10. Connect to the instance as SYSTEM and attempt to revoke the ability for ToddR to query the Instructors table in the Student schema? What happens and why?
11. Perform the necessary steps to ensure that ToddR is unable to query the Instructors table in the Student schema.

Lab 18-2 Configuring Database Auditing

1. Modify your Oracle initialization file for the CERTDB instance to enable auditing to the database.
2. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER and ensure that any connections to the instance are audited, as well as any statements dealing with tables.

3. Issue the appropriate commands to audit all unsuccessful attempts to insert or update the ClassEnrollment table in the Student schema.
4. Configure auditing so that all objects from this point will have LOCK, ALTER, DELETE, GRANT, and RENAME operations audited whenever they occur.
5. Audit any use of the SELECT ANY TABLE privilege by YuryS whenever it occurs.
6. Connect as user Student and grant ToddR the SELECT privilege on the ClassEnrollment table.
7. Connect as user ToddR and attempt to insert data into the Student.ClassEnrollment table. What happens and why?
8. While still connected as ToddR, create a table in your schema with the same structure as the Student.ClassEnrollment table. What happens and why?
9. Connect as user YuryS and attempt to query data from the table created in step 8. What happens and why?
10. Connect as user SYSTEM and configure auditing of all operations on users.
11. Attempt to drop both YuryS and ToddR. What happens and why? Ensure that you have dropped both user accounts.
12. Review the audit trail to determine what events have taken place and whether they were successful.
13. Disable database auditing for the instance.

Answers to Chapter Questions

Chapter Pre-Test

1. In order to configure database auditing, you, as the DBA, need to ensure that the Oracle initialization parameter AUDIT_TRAIL is set to either DB or OS. The default for this parameter is NONE, which means that auditing is not enabled. You then need to issue AUDIT commands to configure the statements, system privileges, and/or object operations that you want audited. Both tasks need to be performed to ensure that database auditing is properly configured. After auditing has taken place, maintenance of the audit trail needs to be performed to ensure that the audit trail does not fill up.
2. If you grant a user the CREATE TABLE privilege, you also need to ensure that the user has been assigned a quota on a tablespace, or has been granted the UNLIMITED TABLESPACE privilege. Failure to do so causes the user's attempts to create a table to fail.
3. If you want a user to be able to create a package in any schema, you need to grant the user the CREATE ANY PROCEDURE privilege. Because this privilege extends to procedures, functions, and packages it will enable the user to create the package in any schema.

4. The user SYSTEM, who is a DBA, cannot grant object privileges on any tables he or she does not own or has been explicitly granted privileges to WITH GRANT OPTION. Only the object owner can grant privileges on his or her objects and not even the DBA can grant privileges, though the DBA can perform almost all other actions on the object.
5. If you want BobW to be able to grant the SELECT privilege on your Orders table to other users, while also having all other privileges on the table, you need to issue two GRANT statements as follows:

```
GRANT ALL ON Orders TO BobW;
GRANT SELECT ON Orders TO BobW WITH GRANT OPTION;
```

6. When you revoke a system privilege from a user to whom it was granted WITH GRANT OPTION, it is not automatically revoked from any other users he or she granted it to. You will need to query the DBA_SYS_PRIVS view for any occurrences of the privilege being granted to others by the user from whom you revoked it and then revoke it from users she or he granted it to manually.
7. Oracle supports value-based auditing, which is implemented through triggers and other application code; database auditing, which is configured using the AUDIT and NOAUDIT commands; and privileged operation auditing, which takes place automatically and cannot be turned off.
8. If a user who owns a table wants to audit who performs INSERTs into the table and when they took place, he or she would need to create a trigger that records this information in another table, which he or she can later review. Object owners cannot configure database auditing using the AUDIT and NOAUDIT commands — only DBAs can.
9. Audit trails on objects will be created whether or not a user who performs the operation rolls back the transaction. The simple attempt to perform an audited operation will create an entry in the audit trail.
10. The database audit trail can be located in the database in the SYS.AUD\$ table, or in an operating system location when auditing is enabled in Oracle8i. The operating system location differs according to the platform on which Oracle is running.
11. The GRANT and REVOKE commands are used to configure system and object privileges. The AUDIT and NOAUDIT commands are used to configure auditing of statements, system privileges, and object access.

Assessment Questions

1. C. You are receiving the error because the password file is not configured for exclusive use for password file authentication. In order for the command to execute successfully, you need to ensure that the INIT.ORA parameter REMOTE_LOGIN_PASSWORD_FILE=EXCLUSIVE is configured.
2. B, C. You need to issue two GRANT statements — the first to grant the SELECT privilege only to JohnS, and the second to grant the SELECT privilege on the Orders table to BobW WITH GRANT OPTION.

3. **B, D.** The only reason that Todd is able to query your TempOrders table is if he is a DBA, who automatically has the SELECT ANY TABLE privilege, or Todd has been granted the SELECT ANY TABLE privilege by the DBA. Another user to whom you only granted the SELECT privilege could not grant it to Todd unless you specified the WITH GRANT OPTION. Granting the UPDATE privilege on the table does not automatically grant the SELECT privilege.
4. **C, D.** In order to create a table in their own schema, users must have the CREATE TABLE privilege and a quota on a tablespace where the table will be stored. The tablespace does not have to be the user's default tablespace.
5. **D.** In order to perform point-in-time recovery, you need to have the SYSDBA privilege. Users holding the SYSOPER privilege can perform full database restores, but cannot issue the RECOVER DATABASE UNTIL command.
6. **A.** The best way to determine what system privileges are active for you in your session is to query the SESSION_PRIVS view. The USER_SYS_PRIVS view will not include those privileges that were granted to PUBLIC and will therefore not provide a complete picture of all available system privileges.
7. **D.** If you configure the AUDIT_TRAIL=OS Oracle initialization parameter on a Windows NT computer running Oracle 8.1.7, the audit trail will be stored in the application event log on the computer where the database resides — that is, the server machine. The AUDIT_FILE_DEST parameter is not available on Windows NT systems.
8. **E.** It is not possible to audit a single user's access to an object in Oracle8i. You can audit object access by all users and you can audit statement or system privilege usage by individual users. The ON <object> clause of the AUDIT command cannot be combined with BY USER.
9. **A, D, F.** When auditing all operations on tables (that is, you issued the AUDIT TABLE command), the audit trail will include the name of the database user performing the operation as well as his or her operating system username, the date and time when the operation occurred, the machine the user was connecting to the instance from, the operation being performed and much more. However, the AUDIT TABLE command does not audit INSERT, UPDATE, or DELETE statements issued on tables, so these operations will not be audited unless configured by the DBA. Furthermore, even if INSERT, UPDATE, and DELETE statements were configured to be audited, database auditing does not record values, but simply the action having taken place. You would need to create a trigger on the table in question to perform value-based auditing.
10. **C.** By default, the individual that can configure value-based auditing is the person who can create a trigger on the table. By default, only the owner of a table can create a trigger and hence she or he is the only individual who can configure value-based auditing.

Scenarios

1. The requirements can be satisfied as follows:

- *Ensure that all users can query any table in the APP schema as well as execute any procedure, package, or function in the APP schema.*

To satisfy this requirement, you would need to determine which objects exist in the APP schema and then grant the SELECT privilege on all tables and views, and the EXECUTE privilege on procedures, functions, and packages to PUBLIC.

You would not want to grant the SELECT ANY TABLE, SELECT ANY VIEW, or EXECUTE ANY PROCEDURE system privileges to PUBLIC as this would open up objects in other schemas to all database users.

- *Allow two junior DBAs (ToddR and YuryS) to be able to change users' passwords, quotas, and other settings. YuryS should also be allowed to create new users.*

To satisfy these requirements, issue the following commands:

```
GRANT ALTER USER TO ToddR, YuryS;  
GRANT CREATE USER TO YuryS;
```

- *Allow your three main developers (SusanI, SteveG, and MylesB), as well as the junior DBAs mentioned previously, to create tables, views, procedures, and indexes in the APP schema. They should not be able to drop or truncate any existing tables or other objects in the APP schema.*

To satisfy this requirement, you will have to provide your developers more privileges than they actually need because it is not possible to limit users to creating objects in certain schemas. Users can be allowed to create objects in their own schema or all schemas, but not only a select group of schemas. To do this, issue the following commands:

```
GRANT CREATE ANY TABLE, CREATE ANY VIEW,  
CREATE ANY PROCEDURE  
TO SusanI, SteveG, MylesB, ToddR, YuryS;
```

By allowing users to be able to create objects in any schema, you are not permitting them to drop existing objects, so the second part of this requirement is satisfied.

- *Disallow all users from inserting, updating, or deleting data in tables and views in the APP schema, except through stored procedures and packages.*

Similar to the first requirement, you would need to determine which tables and views exist in the APP schema and then REVOKE the INSERT, UPDATE, and DELETE privileges from PUBLIC on those objects. You will also need to query the DBA_TAB_PRIVS view to determine if any privileges were explicitly granted to users and revoke them as well, because revoking the privileges from PUBLIC will not reverse privileges explicitly granted to users.

2. To implement the audit policy you would need to issue the following SQL commands or perform the following:

- *All logon and logoff attempts will need to be recorded in the audit trail.*

```
AUDIT SESSION
```

- *The audit trail will need to have any direct manipulation of it recorded.*

```
AUDIT ALL ON SYS.AUD$;
```

- *Any attempts to delete data in the database will need to be recorded each time they occur.*

For existing tables and views in the database, you will need to issue explicit statements to configure auditing of DELETE statements, such as

```
AUDIT DELETE ON Orders BY ACCESS;
```

To configure auditing of DELETE statements for all tables and views created in the future, you would issue the following command:

```
AUDIT DELETE ON DEFAULT BY ACCESS;
```

- *Any changes to the data in the database will need to have the old and new values recorded, as well as information on who performed the change and when.*

To track old and new values changed in the database, you will need to implement value-based auditing through triggers and user-defined audit tables. Oracle's database auditing facility does not track old and new values for rows in the database.

- *The creation, modification, or dropping of database objects will need to be recorded in the audit trail whenever they occur.*

```
AUDIT TABLE, VIEW, PROCEDURE;
```

- *Queries on all user-created database objects will need to be recorded in the audit trail the first time they occur during a user's session.*

For existing tables and views in the database, you will need to issue explicit statements to configure auditing of SELECT statements, such as

```
AUDIT SELECT ON Orders;
```

To configure auditing of DELETE statements for all tables and views created in the future, issue the following command:

```
AUDIT SELECT ON DEFAULT;
```

For either of these AUDIT statements you do not need to specify the BY SESSION clause as this is the default for object auditing.

Lab Exercises

Lab 18-1

1. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a new user called ToddR with a password of your choice, and grant ToddR the ability to connect to the instance, as well as a quota of 10MB on the CERTDB tablespace. Make the CERTDB tablespace the default tablespace for the user.

```
SQL> CREATE USER ToddR IDENTIFIED BY password
      2  DEFAULT TABLESPACE CERTDB
      3  TEMPORARY TABLESPACE TEMP
      4  QUOTA 10M ON CERTDB;
```

User created.

```
SQL> GRANT CREATE SESSION TO ToddR;
```

Grant succeeded.

```
SQL>
```

3. Grant ToddR the ability to create tables, views, and indexes, as well as the ability to grant the CREATE VIEW privilege to others.

```
SQL> GRANT CREATE VIEW TO ToddR
      2  WITH ADMIN OPTION;
```

Grant succeeded.

```
SQL> GRANT CREATE TABLE TO ToddR;
```

Grant succeeded.

```
SQL>
```

There is no CREATE INDEX privilege, but the CREATE TABLE privilege enables you to create indexes on your own tables.

4. Connect to the instance as ToddR and attempt to create a table called InstructTemp with the same structure and data as the Instructors table in the Student schema. What happens and why?

```
SQL> connect toddr/password@certdb.delphi.bradsys.com
Connected.
SQL> CREATE TABLE InstructTemp
      2  AS SELECT * FROM Student.Instructors;
AS SELECT * FROM Student.Instructors
      *
```

ERROR at line 2:

```
ORA-00942: table or view does not exist
```

```
SQL>
```

Even though ToddR has been given the CREATE TABLE privilege, he is not able to create the table because he cannot read the Student.Instructors table. ToddR needs to be given privileges to read the Student.Instructors table as well.

5. Connect to the instance as user Student and grant ToddR the appropriate privileges to be able to create the table in step 4. Allow ToddR to grant the privileges to other users.

```
SQL> connect student/oracle@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> GRANT SELECT ON Instructors TO ToddR  
2 WITH GRANT OPTION;
```

```
Grant succeeded.
```

```
SQL>
```

6. Connect to the instance as ToddR and attempt to create the table in step 4 again. What happens and why?

```
SQL> connect toddr/password@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> CREATE TABLE InstructTemp  
2 AS SELECT * FROM Student.Instructors;
```

```
Table created.
```

```
SQL>
```

7. Connect to the instance as user SYSTEM and create a user called YuryS with a password of your choice, and grant YuryS the ability to connect to the instance, as well as to query data in any table in the database.

```
SQL> connect system/manager@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> CREATE USER YuryS IDENTIFIED BY password;
```

```
User created.
```

```
SQL> GRANT CREATE SESSION, SELECT ANY TABLE TO YuryS;
```

```
Grant succeeded.
```

```
SQL>
```

8. Connect to the instance as YuryS and query the data in the Student.Students table. Attempt to create a table with the same structure as the Students table in the Student schema. What happens and why?

```
SQL> connect yurys/password@certdb.delphi.bradsys.com
Connected.
SQL> SELECT COUNT(*) FROM Student.Students;

COUNT(*)
-----
         11

SQL> CREATE TABLE MyStudents
  2 AS SELECT * FROM Student.Students;
AS SELECT * FROM Student.Students
      *
ERROR at line 2:
ORA-01031: insufficient privileges

SQL>
```

YuryS is not able to create the table because he does not have permissions to create tables. However, he can query the Student.Students table.

9. While still connected as YuryS, attempt to grant ToddR the ability to query the Student.Students table. What happens and why?

```
SQL> GRANT SELECT ON Student.Students
  2 TO ToddR;
GRANT SELECT ON Student.Students
      *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

Even though he can query the Student.Students table, YuryS cannot grant others the privilege to do so since he does not own the table nor has he been given the SELECT privilege on the table WITH GRANT OPTION. The reason that YuryS can query the Student.Students table is because he has been granted the SELECT ANY TABLE privilege by the DBA.

10. Connect to the instance as SYSTEM and attempt to revoke the ability for ToddR to query the Instructor's table in the Student schema? What happens and why?

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> REVOKE SELECT ON Student.Instructors
  2 FROM ToddR;
REVOKE SELECT ON Student.Instructors
      *
ERROR at line 1:
```

```
ORA-01927: cannot REVOKE privileges you did not grant
```

```
SQL>
```

11. Perform the necessary steps to ensure that ToddR is unable to query the Instructors table in the Student schema.

```
SQL> connect student/oracle@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> REVOKE SELECT ON Instructors  
2 FROM ToddR;
```

```
Revoke succeeded.
```

```
SQL>
```

Lab 18-2

1. Modify your Oracle initialization file for the CERTDB instance to enable auditing to the database.

```
C:\CERTDB>SET ORACLE_SID=CERTDB
```

```
C:\CERTDB>svrmgr1
```

```
Oracle Server Manager Release 3.1.7.0.0 - Production
```

```
Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.
```

```
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production  
With the Partitioning option  
JServer Release 8.1.7.0.0 - Production
```

```
SVRMGR> connect internal/oracle;  
Connected.
```

```
SVRMGR> shutdown immediate;
```

```
Database closed.
```

```
Database dismounted.
```

```
ORACLE instance shut down.
```

```
SVRMGR> exit
```

```
Server Manager complete.
```

```
F:\CERTDB>NOTEPAD initCERTDB.ORA
```

```
<<Ensure that you insert the line AUDIT_TRAIL=DB in your INIT.ORA file>>
```

```
C:\CERTDB>svrmgr1
```

```
Oracle Server Manager Release 3.1.7.0.0 - Production
```

```
Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.
```

```
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
```

With the Partitioning option
JServer Release 8.1.7.0.0 - Production

```
SVRMGR> connect internal/oracle
Connected.
SVRMGR> startup pfile=initcertdb.ora
ORACLE instance started.
Total System Global Area                21809180 bytes
Fixed Size                               75804 bytes
Variable Size                           19607552 bytes
Database Buffers                        2048000 bytes
Redo Buffers                             77824 bytes
Database mounted.
Database opened.
SVRMGR> show parameter audit_trail
NAME                                TYPE        VALUE
-----
audit_trail                          string      DB
SVRMGR>
```

2. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER and ensure that any connections to the instance are audited, as well as any statements dealing with tables.

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> AUDIT SESSION, TABLE;

Audit succeeded.

SQL>
```

3. Issue the appropriate commands to audit all unsuccessful attempts to insert or update the ClassEnrollment table in the Student schema.

```
SQL> AUDIT INSERT, UPDATE ON Student.ClassEnrollment
2 BY ACCESS WHENEVER NOT SUCCESSFUL;

Audit succeeded.

SQL>
```

4. Configure auditing so that all objects from this point will have LOCK, ALTER, DELETE, GRANT, and RENAME operations audited whenever they occur.

```
SQL> AUDIT LOCK, ALTER, DELETE, GRANT, RENAME
2 ON DEFAULT BY ACCESS;

Audit succeeded.

SQL>
```

5. Audit any use of the `SELECT ANY TABLE` privilege by `YuryS` whenever it occurs.

```
SQL> AUDIT SELECT ANY TABLE BY YuryS;
```

```
Audit succeeded.
```

```
SQL>
```

6. Connect as user `Student` and grant `ToddR` the `SELECT` privilege on the `ClassEnrollment` table.

```
SQL> connect student/oracle@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> GRANT SELECT ON ClassEnrollment TO ToddR;
```

```
Grant succeeded.
```

```
SQL>
```

7. Connect as user `ToddR` and attempt to insert data into the `Student.ClassEnrollment` table. What happens and why?

```
SQL> connect toddr/password@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> INSERT INTO Student.ClassEnrollment VALUES  
2 (53, 1008, 'Hold', SYSDATE, 1400, NULL, NULL);  
INSERT INTO Student.ClassEnrollment VALUES  
*  
*
```

```
ERROR at line 1:  
ORA-01031: insufficient privileges
```

```
SQL>
```

8. While still connected as `ToddR`, create a table in your schema with the same structure as the `Student.ClassEnrollment` table. What happens and why?

```
SQL> CREATE TABLE MyClassEnrollment AS  
2 SELECT * FROM Student.ClassEnrollment;
```

```
Table created.
```

```
SQL>
```

`ToddR` has both the `SELECT` privilege on the `Student.ClassEnrollment` table and the `CREATE TABLE` privilege, and is able to create the table. He has also been granted a quota on his default tablespace.

9. Connect as user `YuryS` and attempt to query data from the table created in Step 8. What happens and why?

```
SQL> connect yurys/password@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> SELECT ClassID, StudentNumber, Status,
```

```
2 EnrollmentDate, PriceFROM ToddR.MyClassEnrollment;
```

CLASSID	STUDENTNUMBER	STATUS	ENROLLMEN	PRICE
50	1001	Confirmed	01-JAN-01	2000
50	1002	Confirmed	12-DEC-00	1750
50	1005	Confirmed	21-DEC-00	2000
51	1003	Cancelled	01-JAN-01	4000
51	1004	Confirmed	05-JAN-01	4000
51	1008	Confirmed	02-DEC-00	3500
53	1003	Hold	02-JAN-01	1500

7 rows selected.

SQL>

YuryS is able to query the table because he was previously granted the SELECT ANY TABLE privilege.

10. Connect as user SYSTEM and configure auditing of all operations on users.

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> AUDIT USER;
```

Audit succeeded.

SQL>

11. Attempt to drop both YuryS and ToddR. What happens and why? Ensure that you have dropped both user accounts.

```
SQL> DROP USER YuryS;
```

User dropped.

```
SQL> DROP USER ToddR;
DROP USER ToddR
```

```
*
ERROR at line 1:
ORA-01922: CASCADE must be specified to drop 'TODDR'
```

```
SQL> DROP USER ToddR CASCADE;
```

User dropped.

SQL>

You were not able to drop the user ToddR because he owns database objects. The user YuryS does not so dropping him was not an issue.

12. Review the audit trail to determine what events have taken place and whether they were successful.

```
SQL> col username format a10
SQL> col owner format a10
SQL> col action_name format a18
SQL> col obj_name format a17
SQL> SELECT USERNAME, TIMESTAMP, OWNER, OBJ_NAME, ACTION_NAME, RETURNCODE
       2 FROM DBA_AUDIT_TRAIL ORDER BY TIMESTAMP;
```

USERNAME	TIMESTAMP	OWNER	OBJ_NAME	ACTION_NAME	RETURNCODE
SYSTEM	08-JUL-01		TODDR	CREATE USER	0
TODDR	08-JUL-01	STUDENT	INSTRUCTORS	SELECT	2004
STUDENT	08-JUL-01	STUDENT	INSTRUCTORS	GRANT OBJECT	0
TODDR	08-JUL-01	STUDENT	INSTRUCTORS	SELECT	0
SYSTEM	08-JUL-01		YURYS	CREATE USER	911
SYSTEM	08-JUL-01		YURYS	CREATE USER	0
YURYS	08-JUL-01	YURYS	MYSTUDENTS	CREATE TABLE	1031
YURYS	08-JUL-01	STUDENT	STUDENTS	SELECT	0
SYSTEM	08-JUL-01	STUDENT	INSTRUCTORS	REVOKE OBJECT	1927
STUDENT	08-JUL-01	STUDENT	INSTRUCTORS	REVOKE OBJECT	0
STUDENT	08-JUL-01			LOGOFF	0
TODDR	08-JUL-01			LOGOFF	0
TODDR	08-JUL-01	STUDENT	CLASSENROLLMENT	INSERT	947
TODDR	08-JUL-01	STUDENT	CLASSENROLLMENT	INSERT	947
TODDR	08-JUL-01			LOGOFF	0
TODDR	08-JUL-01	STUDENT	CLASSENROLLMENT	INSERT	2004
TODDR	08-JUL-01	TODDR	MYCLASSENROLLMENT	CREATE TABLE	0
YURYS	08-JUL-01			LOGOFF	0
YURYS	08-JUL-01	TODDR	MYCLASSENROLLMENT	SELECT	0
SYSTEM	08-JUL-01			LOGON	0
SYSTEM	08-JUL-01		YURYS	DROP USER	0
SYSTEM	08-JUL-01		TUDDR	DROP USER	1918
SYSTEM	08-JUL-01		TODDR	DROP USER	1922
SYSTEM	08-JUL-01		TODDR	DROP USER	0

24 rows selected.

SQL>

13. Disable database auditing for the instance.

```
AUDIT_TRAIL=NONE
```

Insert the above line in your INIT.ORA file and shutdown and re-start your instance.

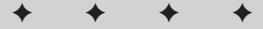
Managing Roles

EXAM OBJECTIVES

- ◆ Managing Roles
 - Create and modify roles
 - Control availability of roles
 - Remove roles
 - Use predefined roles
 - Display role information from the data dictionary

19

CHAPTER



CHAPTER PRE-TEST

1. What kinds of privileges can be assigned to roles?
2. What are the recommended guidelines when implementing roles?
3. How does a user know which roles are currently enabled for his or her session?
4. If you drop a role, what happens to the privileges that the role had assigned to it?
5. If you revoke a privilege from a role, when do users that have been assigned the role lose the privilege?
6. What are two ways that you can enable roles in your session?
7. What permissions do you need to have in order to assign object privileges to a role?
8. Who can create roles? Who can drop a role? Who can modify a role?
9. If you make a role for which you must specify a password a default role for a user, when will the user be prompted to enter the password?

In the previous chapter you learned how to manage system and object privileges for users. However, in volatile database environments, constantly having to grant and revoke privileges to users can become very time consuming and tedious. Oracle provides another database object—the role—to simplify the assignment and management of privileges.

Overview and Benefits of Roles

At various points in the previous chapters you were told to look here for more information on roles and how to make use of them. Well, here you are so we should get going.

A role, at its simplest level, is simply a container. Just like a soda pop can is a container for soda pop, a role is a container to hold privileges. The whole point of a role is to make the assignment of both system and object privileges easier for the DBA.

It is quite likely that in your database you will need to assign certain system privileges (`CREATE SESSION`, `CREATE TABLE`, and so on) as well as object privileges (`SELECT` on tables and views, `INSERT`, `UPDATE`, or `DELETE` on tables, `EXECUTE` on procedures and packages, and so on) to users. The chance of every single combination of privileges assigned to a user being unique to that user is quite small. More typically you will have a series of similar privileges that users will require to perform their jobs. At this point you have two choices: write down all the privileges needed for each set of users and assign them explicitly or create a role and assign the privileges to the role and then assign the role to the user. If you create a role and assign the privileges to the role, when you assign the role to users, they will inherit all the privileges that the role has. If you need to make a change to the privileges (that is, grant others or revoke some), making the change to the role automatically ensures that the change is inherited by all users who have been granted the role.

While creating the role may seem like more work at the outset, if you need to assign the same privileges to new users that come on stream, you only have to assign the role to the users and they automatically inherit all the privileges of the role. Furthermore, you can create multiple roles, each with privileges to perform a specific task, and then assign the roles to those users who require them.

Characteristics of roles

If you wanted to summarize roles in an Oracle8i database, the following list would be typical:

- ◆ A role is a named object in the database that is created by the DBA or a user who has been granted the CREATE ROLE system privilege. The name of each role must be unique within the database and cannot be the same as the name of an existing database user.
- ◆ Roles can be granted both system and object privileges. Any user can grant a role an object privilege, provided the user knows the name of the role.
- ◆ You use the same GRANT and REVOKE commands that you would issue to grant and revoke system and object privileges to grant and revoke roles.
- ◆ Roles can be granted to users and other roles. A role cannot be granted to itself either directly or indirectly. Any attempt to do so generates an error and the assignment will fail.
- ◆ A user may be granted more than one role. By default, all roles granted to the user are automatically enabled and the user's effective privileges are the combination of all privileges granted to the user, PUBLIC, and all privileges granted to roles that are enabled.
- ◆ You can create application roles that require a password to be enabled. Application roles are disabled by default until enabled. When an application role is enabled, all other privileges assigned to the user, and any privileges inherited from other roles that the user has been assigned are disabled, except for the application role that has been enabled and those privileges assigned to PUBLIC.
- ◆ Oracle provides a number of commands to manage which roles will be enabled when the user connects to the instance, and to change the set of enabled roles during a session.

Benefits of roles

If you are asking yourself “Why would I use roles?” and the previous answer indicating that the assignment of privileges can be made easier in the long run is not sufficient, perhaps some additional benefits of roles might be able to convince you.

The first benefit of roles, as already stated, is that roles can make the granting of privileges simpler and easier. The way this works is that instead of manually granting the same set of privileges to new users as they are added to the database, grant them once to a role you create and then grant the role to the users. Simple enough.

Another benefit of roles is that if you need to grant new privileges to users, or revoke existing privileges from users, if these were granted to a role rather than users, you only need to grant or revoke the privileges once — at the role level — instead of numerous times. Furthermore, those privileges granted will be automatically active once the grant or revoke takes place. Changes to role privileges are dynamically modified for all users holding the role.

When you grant privileges to users, those privileges will be available no matter how the user accesses the database. This means that someone using a front-end client application that presents pre-configured forms may need the same level of privileges

as someone connecting to the instance using SQL*Plus and performing interactive queries. The problem with this is that a user of the front-end application could also connect to the instance and perform deletes or other data manipulation that may be more controlled through the front-end application.

Roles that can be selectively enabled and disabled give the user additional privileges by enabling them when the user is using the front-end application, but not allowing the user to have the same set of privileges if he or she connects to the instance using SQL*Plus. Think of this as similar to a frequent flyer program for an airline. Anyone can purchase a seat on the plane, however when it comes to free upgrades to First Class, those individuals who fly more frequently have the privilege of getting the upgrade first. The capability to selectively make privileges available provides for a more secure environment than one where users have all their privileges whenever they access the database.

Roles, in a similar fashion to database users, can be authenticated by the operating system instead of the Oracle server. Creating roles that are authenticated by the operating system allows you to create groups at the OS level and map their membership to roles in the database, and grant those individuals who are members of an OS group additional privileges. This can also allow you to have individuals who just joined your company, or have been moved into new positions within the organization, that have been granted certain group membership by the system administrator automatically inherit database privileges that can allow them to do their job (and you did not have to do anything). Of course, the flip side of this coin is that you, as a DBA, have now lost some control over who can perform certain actions in the database. In other words, having roles authenticated by the OS can be both good and bad, depending on how they are implemented.

When you revoke an object privilege from a user in Oracle8i, if that user was granted the privilege WITH GRANT OPTION, anyone that the user granted the privilege to will also have it revoked. With roles, there are no cascading revokes for object privileges granted to roles because you cannot grant object privileges to a role WITH GRANT OPTION—the syntax is not allowed. However, you can grant a system privilege, or another role, to a role WITH ADMIN OPTION. Doing so allows anyone granted the role to grant those system privileges or roles to others.

Finally, if the potential of the previous benefits of roles have not yet convinced you to use them, just remember that the evaluation of roles by the database takes less work than evaluating privileges assigned to users directly. This means—in a nutshell—that roles provide better performance than granting privileges directly to users. This is because the privileges granted to a role can be cached in the SGA when they are first used and do not need to be reloaded, unless flushed out, the next time a user who has been assigned the role makes use of the privileges. Individual user privileges must be checked against the data dictionary each time a command is sent to the server, and will only be cached if they are frequently used and there is memory. Each user requires a chunk of memory to store his or her permissions—a role only requires one allocation of memory to store its privileges, no matter how many users it is granted to.

Implementing and Using Roles

In determining whether or not to create your own roles and assign them privileges, you should be aware that Oracle has a number of roles created when you create the database and run the CATPROC and CATALOG.SQL scripts. If those roles do not satisfy your requirements, you can also create your own roles (and chances are you will) using the CREATE ROLE command.

Pre-defined roles in Oracle8i

Objective

Use predefined roles

When you create a database in Oracle8i a number of roles will be created as well. Some of these still exist for backward compatibility with previous versions, whereas some are there to ease administration and to provide the DBA with the necessary privileges to do his or her job. In fact, the DBA gets all of the privileges required to manage the database through a pre-defined role called, ironically enough, “DBA.” Table 19-1 lists the pre-defined roles in a standard Oracle8i database. Other roles will be created as you install additional database options or functionality.

Table 19-1
Pre-defined Roles in an Oracle8i Database

<i>Role</i>	<i>Description</i>
DBA	<p>The DBA role is the grand poobah of all roles. Anyone that has been granted the DBA role has also been granted all system privileges available in the database, and full object privileges on almost the entire data dictionary. Furthermore, anyone holding the DBA role can also grant all system privileges and all other roles to other users or roles in the database. Essentially, you can do what you want in the database if you have been granted the DBA role. For these reasons, the DBA role should only be granted to those individuals that will need ultimate control over a database and this number should be relatively small.</p> <p>The DBA role is automatically granted to the users SYS and SYSTEM. The special user INTERNAL also has this role.</p>

Role	Description
SELECT_CATALOG_ROLE	<p>This role allows the holder to query the data dictionary. This role has been granted privileges to the DBA_ V\$ and other views in the data dictionary in order to query its contents. It gives the holder the ability to query more information from the data dictionary than they would normally be able to by querying the ALL_ and USER_ views.</p> <p>This role is automatically granted to the DBA role and can also be granted to other users or roles that need to query the data dictionary.</p>
EXECUTE_CATALOG_ROLE	<p>Allows the holder to execute stored procedures, packages, and functions in the data dictionary not available to regular users. This includes the ability to use the DBMS_LOGMNR package to view redo log file contents, make use of DBMS_PIPE to send messages between session, DBMS_RLS to implement fine-grained access control, and more.</p> <p>The role is granted to the DBA role and may be granted to other users or roles that need to execute those program units.</p>
DELETE_CATALOG_ROLE	<p>The DELETE_CATALOG_ROLE exists to allow a user other than the DBA to be able to delete the database audit trail. Oracle documentation states that this role allows the holder delete data dictionary objects, but the only object that a DELETE command works against is SYS.AUD\$.</p> <p>If the DBA does not want to maintain the log file, this role can be granted to another user to assume this responsibility.</p>
EXP_FULL_DATABASE	<p>When running the Export utility, any user can export database objects in their own schema, or tables and views that they have privileges on. However, only the holder of the EXP_FULL_DATABASE role is able to export the entire database (that is, all objects in all schemas, except those objects in the SYS schema).</p> <p>This role is granted to the DBA role by default. If other users, such as development managers in a software development shop, will need to move an entire database or perform logical backups of the database using the Export utility, then these individuals should be granted this role.</p>

Continued

Table 19-1 (continued)

<i>Role</i>	<i>Description</i>
IMP_FULL_DATABASE	<p>This role allows the holder to perform the opposite of EXP_FULL_DATABASE – to use the Import utility to import all the objects in the import file into an existing database, and, potentially, overwrite existing objects. Like EXP_FULL_DATABASE, this role is granted to the DBA role by default and should not be granted to others. The reason that only the DBA should have this role is its ability to destroy existing database objects with names the same as those in the import file.</p>
CONNECT	<p>This role is provided for backward compatibility with previous versions of Oracle and should not be used. When granted, the holder will be given the following system privileges:</p> <pre>ALTER SESSION CREATE CLUSTER CREATE DATABASE LINK CREATE SEQUENCE CREATE SESSION CREATE SYNONYM CREATE TABLE CREATE VIEW</pre> <p>Inexperienced DBAs will sometimes grant this role to users as a quick way to ensure that they can connect to the instance (that is, so the user has the CREATE SESSION privilege). If you do so, be aware that you are also granting the user the ability to create most database objects, some of which (like views, synonyms, sequences, and so on) do not require a quota on a tablespace. This may cause the number of objects in the database to increase and your control over their management to be diminished.</p>
RESOURCE	<p>Perhaps the most dangerous role to grant a user, RESOURCE is also provided for backward compatibility and should not be used. When granted, the holder of the RESOURCE role is able to make use of these system privileges:</p> <pre>CREATE CLUSTER CREATE INDEXTYPE CREATE OPERATOR CREATE PROCEDURE CREATE SEQUENCE CREATE TABLE CREATE TRIGGER CREATE TYPE UNLIMITED TABLESPACE (when granted)</pre>

<i>Role</i>	<i>Description</i>
	<p>In looking at the privileges available to the RESOURCE role, they mostly deal with object creation. In fact, that is what the role is designed to do – make it easy for the holder to create database objects. In fact, holders of the RESOURCE role should have such an easy time creating database objects that they should not be restricted in which tablespace to create the object in or how big it should grow – let’s give them UNLIMITED TABLESPACE privileges whenever they are granted the role!!!</p> <p>As a general rule of thumb, do not grant the RESOURCE role to any user except the DBA. If you decide to grant the role to users anyway, make sure that you revoke the UNLIMITED TABLESPACE privilege that is automatically granted to the user when the RESOURCE role is granted, and assign the user a quota on only those tablespaces where objects should be created. Don’t forget that UNLIMITED TABLESPACE also allows the holder to create objects on the SYSTEM tables – a bad idea at all times.</p>
AQ_ADMINISTRATOR_ROLE	This role allows the holder to administer advanced queuing in the database. You can grant others the ability to manage the queuing functionality of the database and configure queue users when making use of Oracle Advanced Queuing feature.
AQ_USER_ROLE	This role allows the holder to access, create, and delete queues. It is also used with Oracle’s Advanced Queuing feature.
SNMPAGENT	This role is used by the Oracle Intelligent Agent and in order to perform its required tasks in the database, it is granted to the DBSNMP user when the database is created and configured.

In your database you may also find that other roles are created. They are most likely there because you have enabled certain database functionality. For example, if your database will host a recovery catalog to be used by Recovery Manager for storing information about backup and restore operations, you may find that your database includes the RECOVERY_CATALOG_OWNER role. If you use Oracle Enterprise Manager, you may also have an OEM_MONITOR role created, and so on. The number of pre-defined roles depends on the features selected and enabled.

If you find that these roles meet your needs, then you should make use of them. However, if you find that they do not provide all the functionality required, or give away too many privileges, such as CONNECT and RESOURCE, then do not use them and create your own roles and assign them the privileges you want users to have.

Creating roles

Objective

Create and modify roles

Like most other security-related things in Oracle8i, you can create roles from the command line using SQL*Plus or Server Manager line mode, or by using the Security Manager component of Oracle Enterprise Manager. In order to create a role, you must have been assigned the CREATE ROLE system privilege, or the DBA role. The syntax for the CREATE ROLE command is:

```
CREATE ROLE rolename
      [NOT IDENTIFIED | IDENTIFIED
      BY password | EXTERNALLY | GLOBALLY];
```

The name of each role you create in the database cannot be the same as that of another role, nor can it be the same as that of an existing user. The reason for this is that the names of roles and users are stored in the same place in the data dictionary. If you attempt to create a role with the same name as a user, you will receive the following error:

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> CREATE ROLE Student;
CREATE ROLE Student
      *
ERROR at line 1:
ORA-01921: role name 'STUDENT' conflicts with another user or role name
```

When you issue the CREATE ROLE command, the default is to create a role with the name specified and not require any authentication to have the role enabled for the user. However, if you want to enable the role through an application, you can specify a password for the role by using the IDENTIFIED BY clause followed by the password.

You can also have the role IDENTIFIED EXTERNALLY, which means that the user must be a member of an operating system group with a name that corresponds to the role in order for it to be enabled. In order for roles to be authenticated by the operating system, you need to set the Oracle initialization parameter OS_ROLES to TRUE (the default is FALSE). Once you have decided that you want to support roles that are IDENTIFIED EXTERNALLY, you need to create groups on the server where the database resides with a naming convention as follows:

```
ora_<SID>_<ROLE>[_][d][a]]
```

The naming convention is used to ensure that the proper group membership can be assigned at the OS level to users that need the roles. The meaning of the parts that make up the group name are:

- ♦ **<SID>**— The value of the ORACLE_SID parameter for the instance to which the user will be connecting. It should ideally be the same as the name of the database. On WindowsNT/2000 this value is not case sensitive—it is case sensitive on most other platforms.
- ♦ **<ROLE>**— The name of the role you created in the database to be IDENTIFIED EXTERNALLY. In order for operating system authentication to work, the role must still be created in the database, as well as the group being created at the operating system.
- ♦ **d**— Indicates that the role specified by the <ROLE> portion will be a default role for the user. If either, or both of, **a** and **d** are specified, they must be preceded by an underscore.
- ♦ **a**— Indicates that the role specified by the <ROLE> portion will be granted to the user WITH ADMIN OPTION. If either, or both of, **a** and **d** are specified, they must be preceded by an underscore.

For example, if you create a role in the CERTDB database called StudentAdmin that will be identified externally, if you wanted that role to be the default role for some users, and the default role and granted WITH ADMIN OPTION to others, you would create two groups at the OS level called:

```
ora_CERTDB_StudentAdmin_d
ora_CERTDB_StudentAdmin_da
```

It is important to remember that when specifying names for roles in the database, and then configuring these roles to be IDENTIFIED EXTERNALLY, the length of the role name must satisfy Oracle requirements, whereas the length of the group name must satisfy operating system requirements. For example, a group name in Windows NT cannot be more than 30 characters, so the maximum size of the role name in the Oracle database is significantly less as the group name must also include the SID and the other elements that make it up.

A final note about creating roles to be IDENTIFIED EXTERNALLY is that, by default, these roles cannot be authenticated by a network operating system but must be authenticated by the computer on which the database resides. Furthermore, the user making use of the role must also be logged in to that computer. If you have users connect to the instance remotely using Net8, which is the way most databases tend to be configured, you will also need to set the value of the REMOTE_OS_ROLES INIT.ORA parameter to TRUE as well. This is needed in Windows NT/2000 environments in particular.

Specifying that the role will be IDENTIFIED GLOBALLY, which means that user must be verified by the Oracle Security Server component in order for it to be enabled.



The “Oracle8i: Architecture and Administration” exam will not test your knowledge of globally authenticated roles. Use of the Oracle Security Server and third-party authentication mechanisms is beyond the scope of the exam and this book.

To create a role `StudentAdmin` to be authenticated by the operating system, you could issue the following command:

```
SQL> CREATE ROLE StudentAdmin  
2 IDENTIFIED EXTERNALLY;
```

Role created.

```
SQL>
```

To create a role called `QueryUser`, you can issue the following command:

```
SQL> CREATE ROLE QueryUser;
```

Role created.

```
SQL>
```

To create a role to be enabled by the application by providing a password, you would issue this command:

```
SQL> CREATE ROLE TrainAppUser  
2 IDENTIFIED BY TrainPassword;
```

Role created.

```
SQL>
```

Creating roles using Oracle Enterprise Manager

You can also create roles with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Creating Roles Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to create the role as a user with appropriate permissions to do so.
2. Expand the database and then Security, as shown in Figure 19-1.
3. Right-click on Roles and select Create to create a new role in the database, as shown in Figure 19-2.

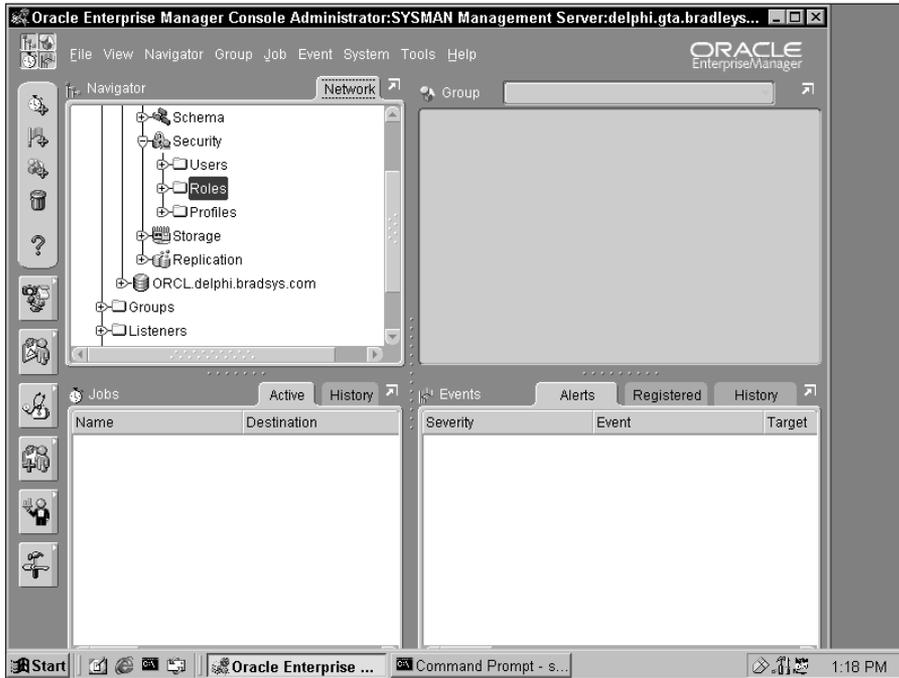


Figure 19-1: Expand the Security node of the database to locate the Roles node.

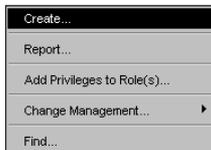


Figure 19-2: Right-click on Roles and select Create to create a new role.

4. On the Create Role dialog box enter the name of the role and then click on the Authentication drop-down list box and select an authentication mechanism, as shown in Figure 19-3.

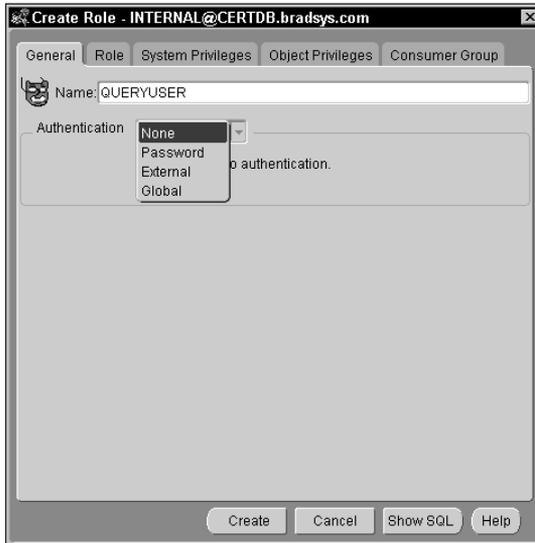


Figure 19-3: Enter the name of the role and select an authentication mechanism in the Create Role dialog box.

5. Click on the Create button to create the role. If everything is OK, you will be shown a dialog box indicating that the role was created, as shown in Figure 19-4. Click OK to exit the dialog box.

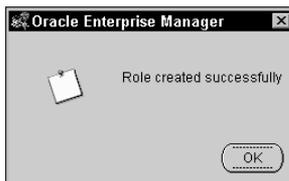


Figure 19-4: Clicking the Create button will display a dialog box indicating that the role was created successfully.

Managing privileges with roles

Objective

Create and modify roles

As mentioned before, a role is simply a container for privileges and other roles. As such, when a role is created it is the same as an empty bottle of water—it has the potential to hold water but is not terribly thirst quenching at this point. In order for a role to really become useful, you need to assign the role privileges. Luckily, the syntax for this is the same as what you have seen earlier.

To assign system privileges to a role, issue the GRANT command as shown here:

```
GRANT system_priv [, system_priv, ...]
  TO role | PUBLIC
  [, role | PUBLIC, ...]
  [WITH ADMIN OPTION];
```

If the role to which you are granting the system privilege is IDENTIFIED GLOBALLY, you cannot grant it system privileges WITH ADMIN OPTION. If you change your mind and do not want the WITH ADMIN OPTION to be specified for a system privilege granted to the role, you need to first revoke the privilege from the role and then grant it again without the WITH ADMIN OPTION.

To grant object privileges to a role, you would also issue the GRANT command with the syntax shown in the following code block. You cannot assign object privileges to roles WITH GRANT OPTION. This is not supported by Oracle and will generate an error if attempted.

```
GRANT ALL [PRIVILEGES] | object_priv [(column, column, ...)]
  [, object_priv [(column, column, ...()) , ...]
  ON [schema_name.]object_name
  TO role | PUBLIC
  [, role | PUBLIC, ...];
```

You can also grant a role to another role. The role can also be granted to the other role WITH ADMIN OPTION, so that anyone holding the role it was granted to can administer those privileges and the role. If you later want to not have the role granted WITH ADMIN OPTION, like with system privileges, you will need to revoke the role first and then grant it without specifying the WITH ADMIN OPTION. The syntax for granting a role to another role is as follows:

```
GRANT role_name [, role_name, ...]
  TO role | PUBLIC
  [, role | PUBLIC, ...]
  [WITH ADMIN OPTION];
```

It is possible to grant roles, system privileges, and object privileges to the same role. There is no restriction placed on the types of privileges that a role may hold. However, you cannot, in a single GRANT command, grant both object and system privileges—you must issue a separate grant command for each type of privilege granted. This is because the syntax for the GRANT command differs slightly for the granting of system and object privileges and Oracle will not know how to parse the statement if you mix both. Also, when granting object privileges, you must be connected as the owner otherwise you will receive an error when attempting the GRANT. For example, to grant the appropriate privileges to the QueryUser role, you could issue the following command (but make sure you are connected as the right user when doing so):

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
```

```
SQL> GRANT CREATE SESSION TO QueryUser;

Grant succeeded.

SQL> GRANT CONNECT, RESOURCE
  2 TO QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON
  2 Student.Instructors
  3 TO QueryUser;
Student.Instructors
  *
```

ERROR at line 2:
ORA-01031: insufficient privileges

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
SQL> GRANT SELECT ON
  2 Student.Instructors
  3 TO QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON
  2 Student.Students
  3 TO QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON Courses
  2 TO QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON ClassEnrollment
  2 TO QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON Locations
  2 TO QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON ScheduledClasses
  2 TO QueryUser;

Grant succeeded.

SQL>
```

Revoking system privileges from roles

After you have granted system and privileges or roles to other roles, if you want to remove some of the privileges or roles you have granted, you would need to issue the REVOKE command. The syntax to revoke a system privilege or a role from a role is as follows:

```
REVOKE system_priv | role_name
      [, system_priv | role_name, ...]
FROM role | PUBLIC
      [,role | PUBLIC, ...];
```

To remove an object privilege from a role, the syntax is as follows:

```
REVOKE ALL [PRIVILEGES] | object_priv
      [, object_priv, ...]
ON [schema_name.]object_name
FROM role | PUBLIC
      [,role | PUBLIC, ...]
[CASCADE CONSTRAINTS]
```

For example, if you wanted to revoke the RESOURCE role from QueryUser, you would issue the following command:

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
SQL> REVOKE RESOURCE FROM QueryUser;

Revoke succeeded.

SQL>
```

Granting and revoking privileges using Oracle Enterprise Manager

You can also grant and revoke privileges to/from roles with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Granting and/or Revoking Privileges to/from Roles Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to revoke the role's privileges as a user with appropriate permissions to do so.
2. Expand the database and then Security.
3. Expand Roles and right-click the role whose privileges you want to manage and select Edit, as shown in Figure 19-5.

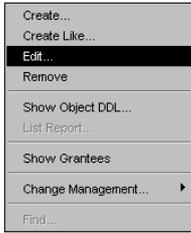


Figure 19-5: Select Edit to modify a role in Enterprise Manager

4. On the Edit Role dialog box, click on tabs for the type of privileges you want to grant or revoke (Role, System Privilege, Object Privilege) and assign or revoke the appropriate permissions, as shown in Figure 19-6.

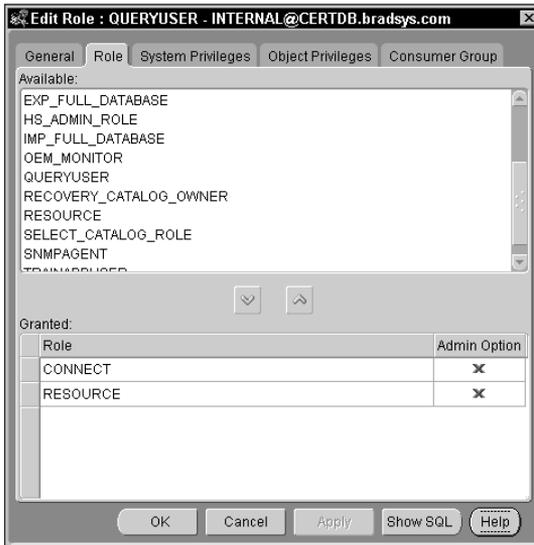


Figure 19-6: Click on the tabs in the Edit Role dialog box to grant or revoke the privileges desired.

5. When you have completed your management of privileges for the role, click Apply to save your changes and OK to exit the dialog box.

Modifying Roles

Oracle allows you to change the way that it is authenticated. It is possible to specify a password for a role that did not have one, or change a role that was previously IDENTIFIED EXTERNALLY to be authenticated by the database and not require a password, or any variation of these. The syntax for modifying a role is:

```
ALTER ROLE rolename
  [NOT IDENTIFIED | IDENTIFIED
  BY password | EXTERNALLY | GLOBALLY];
```

Modifying roles using Oracle Enterprise Manager

You can also modify roles with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Creating Roles Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to modify the role as a user with appropriate permissions to do so.
2. Expand the database and then Security and then Roles.
3. Right-click on role you want to modify and select Edit to bring up the Edit Role dialog box.
4. On the Edit role dialog box click on the Authentication drop-down list box and select an authentication mechanism.
5. When you have made your selection, click Apply to save your changes and OK to exit the dialog box.

Managing roles for users

Objective

Control availability of roles

Once you have created a role and granted the role and object and system privileges desired, you next need to assign the role to users that you want to inherit all the privileges that the role has. Ironically enough, the method for doing so is almost identical to that of assigning system privileges to users — using the GRANT command or through Oracle Enterprise Manager.

The syntax of the GRANT command to grant roles to users (as well as other roles) is:

```
GRANT role_name [, role_name, ...]
  TO user_name | role | PUBLIC
  [, user_name | role | PUBLIC, ...]
  [WITH ADMIN OPTION];
```

The syntax does not require much additional explanation because you have seen it in the previous chapter and earlier in this one, but one thing is important to note. If you grant a role WITH ADMIN OPTION, you are allowing the user to whom you are granting the role to also grant the role (and all its associated privileges) to others. Before doing so it is important to ensure that the effective results by such an action are also the desired and expected results. In other words, as with most things in the realm of security, you need to plan it carefully and test it first.

In order to grant a role to a user or another role you need to be the owner of the role (that is, you are the user that issued the CREATE ROLE command) or have been granted the GRANT ANY ROLE privilege by the DBA.

To grant the QueryUser role to MarkS and JohnS, you could issue the following command:

```
SQL> GRANT QueryUser TO JohnS, MarkS;
Grant succeeded.
SQL>
```

As you can grant a role to a user, you can also revoke the role to remove all of the role's privileges from the user. This can be accomplished using the REVOKE command or Oracle Enterprise Manager. The syntax of the REVOKE command is again similar to what you have seen previously, as follows:

```
REVOKE role_name [, role_name, ...]
FROM user_name | role | PUBLIC
[, user_name | role | PUBLIC, ...];
```

If you revoke the role from a user, the role's permissions will not be immediately taken away from the user, unless the user disconnects from the instance or disables the role. However, the user will not be able to re-enable the role on the next connection attempt or by using the SET ROLE command once it has been revoked.

To revoke the CONNECT role from JohnS, you would issue the following command:

```
SQL> REVOKE CONNECT FROM JohnS;
Revoke succeeded.
SQL>
```

Granting and revoking roles to/from users using Oracle Enterprise Manager

You can also grant and revoke roles with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Granting and Revoking Roles Using Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which you want to grant and revoke roles to users and other roles as a user with appropriate permissions to do so.
2. Expand the database and then Security and then Users, to grant and revoke roles from users, or Roles, to grant and revoke roles from other roles.
3. Right-click on role or user you want to modify and select Edit to bring up the Edit Role or Edit User dialog box.
4. On the Edit Role or Edit User dialog box click on the Role tab and select the role to grant and then click on the down arrow, or select a role to revoke from the list of granted roles and click on the up arrow, as shown in Figure 19-7 when granting and revoking roles from a user.

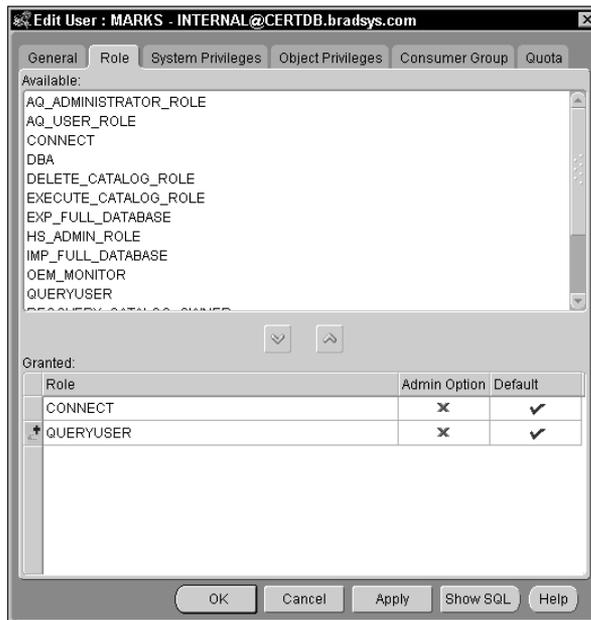


Figure 19-7: The Role tab of the Edit User dialog box allows you to grant or revoke roles from a user.

5. When you have made your selection, click Apply to save your changes and OK to exit the dialog box.

Establishing default roles for users

Once you grant a role to a user, it is automatically configured to be a default role. This means that when the user connects to the instance, the role will automatically be enabled for the user and any privileges that the role has been granted will be available to the user. If this is what you need and desire to occur, then there is no need to change this behavior. However, if you only want some of the roles granted to the user to be active when the user connects to the instance, you need to modify the set of default roles that are automatically enabled.

The ALTER USER command, or Oracle Enterprise Manager, can be used to manage a user's default roles. The syntax of the ALTER USER command to manage a user's default role list is as follows:

```
ALTER USER username DEFAULT ROLE
    role [, role, ...] | ALL [EXCEPT role [, role, ...]] | NONE;
```

If you do not want Oracle to enable all roles that a user has been granted, you must use the ALTER USER command to disable any roles that you do not want the user to have when they connect to the instance. You can then programmatically enable the roles or have the user issue the SET ROLE command to enable those roles that you disabled by default.

If you grant the user a role that requires a password, if you make that role a default role, the user will not be required to enter a password in order to make use of the privileges granted to the role. In essence, making a role that has a password a default role for the user bypasses the password requirement. In this way some users may have the role and its privileges when they connect, by default, while other users will be required to enable the role manually and specify a password in order to access the privileges granted the role.

You can also disable all roles that have been assigned to the user by using the NONE option when specifying which roles are default roles. After doing so, all roles granted to the user will be disabled and will need to be enabled using the SET ROLE command. The user will only have the capability to perform actions according to those system and object privileges that have been assigned directly to him or her, or to PUBLIC.



If you assigned the CONNECT role to the user so that he or she is able to connect to the instance, disabling all roles also disables the CONNECT role and may prevent the user from getting access to the database.

For example, if you grant the user MarkS the StudentAdmin, QueryUser and CONNECT roles, but do not want the StudentAdmin role to be enabled when the user connects, you can issue the following commands:

```
SQL> GRANT CONNECT, QueryUser, StudentAdmin TO MarkS;
```

```
Grant succeeded.
```

```
SQL> ALTER USER MarkS  
2 DEFAULT ROLE ALL EXCEPT StudentAdmin;
```

```
User altered.
```

```
SQL>
```

Establishing Default Roles for a User With Oracle Enterprise Manager

You can also configure a user's default role with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Configuring Default Roles for a User with Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which the user whose default role configuration you want to modify is created as a user with appropriate permissions to do so.
2. Expand the database and then Security and then Users.
3. Right-click on user whose default roles you want to configure and select Edit to bring up the Edit User dialog box.
4. On the Edit User dialog box click on the Role tab to display a list of roles available and those that have been assigned to the user.
5. In the list of roles granted to the user, click on the Default column for the role to make it the default (that is, display a check mark) or not make it the default (that is, display a red X), as shown in Figure 19-8.
6. When you have made your selection, click Apply to save your changes and OK to exit the dialog box.



Figure 19-8: Click the Default column of the role that you want to make the default, or not.

Enabling and disabling roles

One of the major benefits of roles is the ability to selectively grant and revoke a set of privileges by enabling and disabling roles that contain them. While a user is connected to the instance, your application (typically) can issue the SET ROLE command, or execute the DBMS_SESSION.SET_ROLE package procedure, to enable or disable roles dynamically.

A role that was created with a password will need to have the password specified when it is enabled. This allows you to further control the enabling of roles by users by ensuring that roles are only enabled while a particular application is being used to connect to the database. In essence, the user can have one base set of privileges through roles that are enabled by default when connecting interactively with SQL*Plus, and an elevated set of privileges when connecting to the instance using a front-end client application that enables other roles that have been granted to the user but require a password.

The syntax for the SET ROLE command is as follows:

```
SET ROLE ALL [EXCEPT role_name [,role_name]] | NONE |
role_name [IDENTIFIED BY password]
[ , role_name [IDENTIFIED BY password, ...]];
```

If you want to disable a role for a user, you need to issue the SET ROLE command, or execute the DBMS_SESSION.SET_ROLE procedure a second time omitting the role that you do not want the user to have enabled. In other words there is no UNSET ROLE command or its equivalent.

The syntax for the DBMS_SESSION.SET_ROLE procedure is similar to the SET ROLE command, in that everything following the SET ROLE portion needs to be included as part of the parameter sent to the procedure, as in this example:

```
SQL> EXECUTE DBMS_SESSION.SET_ROLE('ALL EXCEPT QueryUser');  
  
PL/SQL procedure successfully completed.  
  
SQL>
```

It is important to note a few restrictions on enabling roles. First, any role that has been created with the IDENTIFIED GLOBALLY clause cannot be dynamically enabled or disabled. This is because Oracle has no real control over who is able to make use of the role and relies upon the third-party authentication server to determine whether or not a user should have a role enabled.

Another important point to remember is that Oracle does not support the enabling or disabling of roles within stored procedures (or package procedures). This means that you cannot create a stored procedure in the database using the CREATE PROCEDURE command and include either the SET ROLE command or a call to the DBMS_SESSION.SET_ROLE package procedure. The reason for this deals with the security context of the stored procedures at execution time. Enabling or disabling a role within the procedure could change the security context at execution time (that is, while the procedure is running), which is not permitted by Oracle. However, you can create an anonymous PL/SQL block and include a call to the DBMS_SESSION.SET_ROLE package procedure within the anonymous PL/SQL block to enable or disable roles, as in this example:

```
SQL> BEGIN  
  2  DBMS_SESSION.SET_ROLE('ALL EXCEPT QueryUser');  
  3  END;  
  4  /  
  
PL/SQL procedure successfully completed.  
  
SQL>
```

A call to DBMS_SESSION.SET_ROLE is also permitted in client-side tools such as Oracle Forms or Oracle Reports to change the security context of the application being run, just not in stored procedures.

Finally, though it may appear obvious, when you issue the SET ROLE command and want to enable one or more roles that require a password, you need to ensure that you specify the correct password otherwise the entire statement will fail.

Dropping Roles

Objective

Remove roles

If you no longer need a role that you have been using, you can drop it from the database by issuing the DROP ROLE command, or by using Oracle Enterprise Manager. In order to drop a role you must be the user who created the role, or have been granted the DROP ANY ROLE system privilege, or have been granted the role WITH ADMIN OPTION. If any of these is not true, the command will fail and the role will not be dropped.

The syntax of the DROP ROLE command is as follows:

```
DROP ROLE role_name;
```

When you drop a role, any user or role to which the role being dropped has been granted will have it revoked at the time the role is dropped. Any privileges that the role granted its holders will also be revoked at the time the role is dropped.

To drop the StudentAdmin role from the database, you can issue the following command:

```
SQL> DROP ROLE StudentAdmin;  
  
Role dropped.  
  
SQL>
```

Dropping roles with Oracle Enterprise Manager

You can also drop roles with Oracle Enterprise Manager using the following steps:

STEP BY STEP: Dropping Default Roles with Oracle Enterprise Manager

1. Start the Oracle Enterprise Manager Console and connect to the database in which the role you want to drop is created as a user with appropriate permissions to do so.
2. Expand the database and then Security and then Roles.
3. Right-click on the role that you want to drop and select Remove, as shown in Figure 19-9.
4. If the role is assigned to users, you will be prompted to confirm that you want to drop it, as shown in Figure 19-10. If you are sure that this role should be dropped click Yes to confirm the removal of the role, otherwise click No to abort the process.

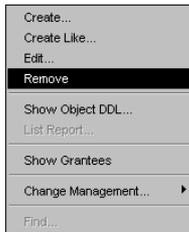


Figure 19-9: Right-click on the Role to be dropped and select Remove to drop it.

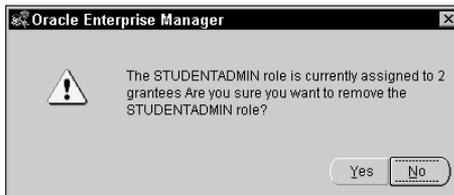


Figure 19-10: If the role being dropped is assigned to users, confirm the removal by clicking Yes on this dialog box.

Querying role information in the data dictionary

Objective

Display role information from the data dictionary

As you assign roles, you may need to determine which roles have been assigned which other roles, or what system or object privileges a role has been granted. You may also want to know which system privileges were granted WITH ADMIN OPTION to a role, or which roles have had the same clause used when granted to a user. You may also want to determine which roles are configured as default for a user. Oracle data dictionary views provide this information, as does Oracle Enterprise Manager. In your manipulation of roles and users, you have already been shown how to locate information in Oracle Enterprise Manager for roles and users. For that reason the focus here will be on the data dictionary views available to extract role information.

Oracle provides a number of data dictionary views to help you in finding out information about roles. They include:

- ♦ **DBA_ROLES** — This view returns a list of roles available in the database and whether or not a password is required to enable them, as shown here:

```
SQL> SELECT * FROM DBA_ROLES;
```

ROLE	PASSWORD
CONNECT	NO
RESOURCE	NO

```

DBA                                NO
SELECT_CATALOG_ROLE              NO
EXECUTE_CATALOG_ROLE             NO
DELETE_CATALOG_ROLE              NO
EXP_FULL_DATABASE                 NO
IMP_FULL_DATABASE                 NO
RECOVERY_CATALOG_OWNER           NO
AQ_ADMINISTRATOR_ROLE            NO
AQ_USER_ROLE                      NO
SNMPAGENT                         NO
OEM_MONITOR                       NO
HS_ADMIN_ROLE                     NO
QUERYUSER                         NO
TRAINAPPUSER                      YES

```

16 rows selected.

SQL>

- ♦ **DBA_ROLE_PRIVS**— This view presents a list of roles and whom they were granted to. The list of whom the role was granted to includes both users and other roles. The view also lets you know whether the role has been granted to the user or role WITH ADMIN OPTION and whether or not it is the default role for the user. For example, to determine which users and roles were granted the EXP_FULL_DATABASE role, you could issue the following query:

```

SQL> SELECT * FROM DBA_ROLE_PRIVS
      2 WHERE GRANTED_ROLE='SELECT_CATALOG_ROLE';

```

GRANTEE	GRANTED_ROLE	ADM	DEF
DBA	SELECT_CATALOG_ROLE	YES	YES
EXP_FULL_DATABASE	SELECT_CATALOG_ROLE	NO	YES
IMP_FULL_DATABASE	SELECT_CATALOG_ROLE	NO	YES
OEM_MONITOR	SELECT_CATALOG_ROLE	NO	YES
SYS	SELECT_CATALOG_ROLE	YES	YES

SQL>

- ♦ **ROLE_ROLE_PRIVS**— This view lists all the roles that have been granted to other roles, and whether the roles have been granted WITH ADMIN OPTION. To retrieve a list of all the roles that the DBA role has been granted, you would issue the following query:

```

SQL> SELECT * FROM ROLE_ROLE_PRIVS
      2 WHERE ROLE='DBA';

```

ROLE	GRANTED_ROLE	ADM
DBA	DELETE_CATALOG_ROLE	YES
DBA	EXECUTE_CATALOG_ROLE	YES
DBA	EXP_FULL_DATABASE	NO

```

DBA          IMP_FULL_DATABASE          NO
DBA          SELECT_CATALOG_ROLE        YES

```

```
SQL>
```

- ♦ **DBA_SYS_PRIVS**— This view, as you have seen previously in chapter 18, is used to retrieve a list of all system privileges granted to users and roles in the database and whether or not the system privileges have been granted WITH ADMIN OPTION. For example, to get a list of all system privileges granted the RESOURCE role, you can issue the following command:

```

SQL> col grantee format a15
SQL> col privilege format a30
SQL> SELECT * FROM DBA_SYS_PRIVS
      2 WHERE GRANTEE='RESOURCE';

```

GRANTEE	PRIVILEGE	ADM
RESOURCE	CREATE CLUSTER	NO
RESOURCE	CREATE INDEXTYPE	NO
RESOURCE	CREATE OPERATOR	NO
RESOURCE	CREATE PROCEDURE	NO
RESOURCE	CREATE SEQUENCE	NO
RESOURCE	CREATE TABLE	NO
RESOURCE	CREATE TRIGGER	NO
RESOURCE	CREATE TYPE	NO

```
8 rows selected.
```

```
SQL>
```

- ♦ **ROLE_SYS_PRIVS**—If you wanted to get a list of system privileges that were granted only to roles, you can query this view. It is a subset of the information presented by DBA_SYS_PRIVS because it does not include users, although the output is similar, as shown here:

```

SQL> col role format a15
SQL> SELECT * FROM ROLE_SYS_PRIVS
      2 WHERE ROLE='RESOURCE';

```

ROLE	PRIVILEGE	ADM
RESOURCE	CREATE CLUSTER	NO
RESOURCE	CREATE INDEXTYPE	NO
RESOURCE	CREATE OPERATOR	NO
RESOURCE	CREATE PROCEDURE	NO
RESOURCE	CREATE SEQUENCE	NO
RESOURCE	CREATE TABLE	NO
RESOURCE	CREATE TRIGGER	NO
RESOURCE	CREATE TYPE	NO

```
8 rows selected.
```

```
SQL>
```

- ♦ **ROLE_TAB_PRIVS**—If you wanted to get a list of all object privileges that have been granted to a role, you can query the `ROLE_TAB_PRIVS` data dictionary view. The information provided includes the name of the role, the owner of the object, the name of the object, the privilege granted, and whether the privilege was granted `WITH GRANT OPTION`, which will always indicate `NO` as roles cannot be granted object privileges `WITH GRANT OPTION`. If you granted privileges to columns on tables or views, the column name will also be displayed in the `COLUMN_NAME` column, otherwise that column will always be null. To display a list of privileges granted to roles on objects in the `Student` schema, you would issue the following query:

```
SQL> col column_name noprint
SQL> col owner noprint
SQL> col table_name format a17
SQL> col privilege format a10
SQL> SELECT * FROM ROLE_TAB_PRIVS
      2 WHERE OWNER='STUDENT';
```

ROLE	TABLE_NAME	PRIVILEGE	GRA
QUERYUSER	CLASSENROLLMENT	SELECT	NO
QUERYUSER	COURSES	SELECT	NO
QUERYUSER	INSTRUCTORS	SELECT	NO
QUERYUSER	LOCATIONS	SELECT	NO
QUERYUSER	SCHEDULEDCLASSES	SELECT	NO
QUERYUSER	STUDENTS	SELECT	NO

6 rows selected.

SQL>

- ♦ **SESSION_ROLES**—Users may need to find out which roles are enabled for their session. This can be useful when tracking down a problem for a user, or just to understand what the user's current set of privileges might be, including those privileges inherited from roles. To determine what roles are enabled in your current session, execute the following query:

```
SQL> col role format a30
SQL> SELECT * FROM SESSION_ROLES;
```

```
ROLE
-----
DBA
SELECT_CATALOG_ROLE
HS_ADMIN_ROLE
EXECUTE_CATALOG_ROLE
DELETE_CATALOG_ROLE
EXP_FULL_DATABASE
IMP_FULL_DATABASE
AQ_ADMINISTRATOR_ROLE
```

```

TRAINAPPUSER
9 rows selected.

SQL> set role all except TrainAppUser;

Role set.

SQL> SELECT * FROM SESSION_ROLES;

ROLE
-----
DBA
SELECT_CATALOG_ROLE
HS_ADMIN_ROLE
EXECUTE_CATALOG_ROLE
DELETE_CATALOG_ROLE
EXP_FULL_DATABASE
IMP_FULL_DATABASE
AQ_ADMINISTRATOR_ROLE
QUERYUSER
CONNECT
RESOURCE

11 rows selected.

SQL>

```

Note that as you issue the SET ROLE command, the output of the query will change to only display the currently enabled roles for the session.

Guidelines for using roles

Now that you are familiar with creating, modifying, and dropping roles, as well as granting and revoking privileges and other roles to a role, a few insights into the best and most effective way to make use of roles (according to Oracle) might be worthwhile. The benefits of roles were outlined earlier; however, in implementing them, a few guidelines might make the job a bit easier and save you time and effort in the long run.

The first step in the efficient use and management of roles is to break roles down into two categories: task roles and user roles. Task roles are those roles that are created in the database and assigned permissions that allow a certain application task to be performed. These types of roles are never assigned to users and are always only assigned to other roles — user roles.

An example of a task role implementation, for an order entry system, would be entering an order. In order for an order to be entered, INSERT privileges are required in the OrderHeader and OrderDetail tables (at a minimum). You may also

need to allow UPDATE privileges on the Inventory and Customer tables to allocate inventory and update the customer's credit information. Other privileges may also be required. The whole purpose of the task role is to ensure that the privileges to perform a task are assigned to it. If you need to change the privileges required, you change the privileges granted to the task role.

A user role is created and defined within the database to be assigned role and other privileges for a particular group of individuals in the organization. This role is assigned directly to users after it has been assigned other roles and privileges required by the individuals to whom the role is being granted.

For example, you may have order entry clerks in the company that are responsible for taking orders, answering client questions on order status, and even updating orders. You would create a role in the database called OrderEntryClerk and assign it the role to enter an order, update an order, perform order queries, and so on. This same role can also be assigned the CREATE SESSION system privilege so that the user can connect to the instance. The point here is that the role inherits the bulk of its privileges from the tasks those individuals holding it need to perform.

Guidelines for creating and administering roles efficiently can be summarized as follows:

1. Create roles to mimic tasks that need to be performed in the database, such as enter an order or query order status. Name these roles similar to the task that is being performed, such as OrderEntry or QueryOrder.
2. Grant privileges required to perform the task associated with the role to the role. For example, grant all object and system privileges required to enter an order to the OrderEntry role.
3. Create user roles to group similar types of users together. This can usually be accomplished quite easily because most organizations have users with similar titles. Create the roles to mimic the structure of employee groupings in the company.
4. Grant task roles to user roles to allow users to perform the appropriate tasks for their job function. Although some system privileges, such as CREATE SESSION, may be granted directly to user roles, it is generally not a good idea to grant privileges to user roles — create a task role and grant the task role to the user role instead.

While initially this may seem like more work than the time savings that it may generate, if you have an environment where new users frequently join the organization, or move to different job functions, it can make life a lot easier in the long run. Furthermore, as the new privileges need to be granted or taken away, you only need to grant or revoke them from the role and all user roles and users will have them, or not.

Key Point Summary

In preparing for the “Oracle8i DBA: Architecture and Administration” exam, please keep these points regarding roles structure and relationships in mind:

- ♦ Roles can be used to reduce the administrative overhead of granting and managing privileges for users.
- ♦ A number of pre-defined roles are created by Oracle to assist in the administration of the database. They are assigned to the DBA role by default, which is assigned to the SYSTEM and SYS users. They may also be assigned to other users, if necessary.
- ♦ The DBA and any user granted the CREATE ROLE privilege can create roles.
- ♦ Roles can require a password to be enabled or not. Roles can also be authenticated by the operating system or globally.
- ♦ Roles can be assigned system and object privileges, or other roles. In order to assign a privilege to a role, you must have sufficient privileges to do so. In other words, you cannot assign object privileges to a role if you are the DBA unless you have been given privileges by the object’s owner to do so.
- ♦ Once a role is created, it can be assigned privileges by any user that knows about it.
- ♦ Revoking privileges from a role will revoke them from the user when the user disconnects or re-enables the role. Privileges will not be revoked from the user at the same time as they are revoked from the role.
- ♦ Dropping a role revokes all privileges associated with the role from users that have been granted the role.
- ♦ Roles can be enabled or disabled within a session by using the SET ROLE command or the DBMS_SESSION.SET_ROLE package procedure.
- ♦ All roles granted to a user are automatically enabled and become the user’s default roles. The list of default roles for a user can be managed by using the ALTER USER command, or Oracle Enterprise Manager.
- ♦ You should create task roles and assign them privileges required to perform certain tasks, and user roles to categorize users. User roles should be assigned task roles to allow users to perform tasks.



STUDY GUIDE

At this point you should test your understanding of managing roles by going through the labs and answering the assessment questions and scenarios in this part of the chapter. This will help you be better prepared to write the “Oracle 8i: Architecture and Administration” exam.

Assessment Questions

1. Which of the following statements will fail when granting privileges to the role MyRole? (Choose the best answer.)
 - A. GRANT CONNECT TO MyRole;
 - B. GRANT CONNECT TO MyRole WITH ADMIN OPTION;
 - C. GRANT SELECT ON Students TO MyRole;
 - D. GRANT SELECT ON Students TO MyRole WITH GRANT OPTION;
 - E. GRANT DBA TO MyRole WITH ADMIN OPTION;
2. A user, SharonT, has been granted the JuniorDBA role that allows her to change user passwords. You revoke the JuniorDBA role from SharonT. Assuming that she was connected to the instance when you revoked the role from her, which of the following commands will she be able to successfully execute? (Choose all correct answers.)
 - A. ALTER USER ToddR IDENTIFIED BY password;
 - B. CREATE USER TomS IDENTIFIED BY password;
 - C. ALTER USER ToddR TEMPORARY TABLESPACE Temp;
 - D. DROP USER ToddR;
 - E. DROP USER TomS CASCADE;
3. What authentication methods can be used in Oracle8i for roles? (Choose all correct answers.)
 - A. Operating System Authentication
 - B. No Authentication
 - C. Password Authentication
 - D. DBA Authentication
 - E. Global Authentication

4. If TomS had the roles JuniorDBA, QueryRole, and AppDevRole granted and you wanted to ensure that only the QueryRole was enabled when TomS connected to the instance, which of the following commands would you issue? (Choose the best answer.)
- A. ALTER USER TomS DEFAULT ROLE ALL EXCEPT QueryRole;
 - B. ALTER USER TomS DEFAULT ROLE QueryRole;
 - C. SET ROLE ALL EXCEPT QueryRole;
 - D. SET ROLE ALL EXCEPT AppDevRole, JuniorDBA
 - E. ALTER USER TomS DEFAULT ROLE NONE EXCEPT QueryRole;
5. If TomS had the roles JuniorDBA, QueryRole, and AppDevRole granted and you wanted to ensure that when he connects to the instance using the AppClient software he is granted all privileges of QueryRole and AppDevRole, which command would you issue from the AppClient software? (Choose the best answer.)
- A. ALTER USER TomS DEFAULT ROLE ALL EXCEPT JuniorDBA;
 - B. ALTER USER TomS DEFAULT ROLE JuniorDBA;
 - C. SET ROLE ALL EXCEPT QueryRole;
 - D. SET ROLE ALL EXCEPT AppDevRole, JuniorDBA
 - E. SET ROLE ALL EXCEPT JuniorDBA;
6. You want to create a role called SysAdmin that will be granted some privileges to perform backups and other file-related actions for the database. You want to have your network and server administrators to be granted this role whenever they logon on the computer hosting your database. How would you issue the statement to create the role? (Choose the best answer.)
- A. CREATE ROLE SysAdmin IDENTIFIED BY HostOS;
 - B. CREATE ROLE SysAdmin IDENTIFIED GLOBALLY;
 - C. CREATE ROLE SysAdmin NOT IDENTIFIED;
 - D. CREATE ROLE SysAdmin IDENTIFIED EXTERNALLY;
 - E. CREATE ROLE SysAdmin;

7. You want to create a role called SysAdmin in the CERTDB database, which will be granted some privileges to perform backups and other file-related actions for the database. You want to have your network and server administrators to be granted this role whenever they logon on the computer hosting your database. What parameters do you need to configure in the INIT.ORA file for your instance? (Choose all correct answers.)
- A. REMOTE_OS_AUTHENT_PREFIX=OPS\$
 - B. REMOTE_OS_AUTHENT=TRUE;
 - C. OS_ROLES=TRUE;
 - D. OS_ROLE_AUTHENT=TRUE;
 - E. REMOTE_ROLE_AUTHENT=TRUE;
8. You have created a role called SysAdmin in the CERTDB database, which will be granted some privileges to perform backups and other file-related actions for the database. You want to have your network and server administrators to be granted this role whenever they logon on the computer hosting your database. The role should be the default role for these individuals but they should not be allowed to grant it to others. What operating systems group do you need to create to satisfy this requirement? (Choose the best answer.)
- A. SYSADMIN_D
 - B. ORA_SYS_ADMIN_D
 - C. ORA_CERTDB_SYSADMIN_DA
 - D. ORA_CERTDB_SYSADMIN_D
 - E. ORA_SYSADMIN_A
9. Which command would you issue from within an anonymous PL/SQL block to enable all roles except JuniorDBA for a user? (Choose the best answer.)
- A. DBMS_SESSION.SET_ROLE('ALL EXCEPT JuniorDBA');
 - B. DBMS_SESSION.SET_ROLE('JuniorDBA');
 - C. SET ROLE ALL NONE EXCEPT JuniorDBA;
 - D. SET ROLE ALL EXCEPT JuniorDBA
 - E. You cannot change roles from within anonymous PL/SQL blocks.
10. Which command would you issue from within a stored procedure to enable all roles except JuniorDBA for a user? (Choose the best answer.)
- A. DBMS_SESSION.SET_ROLE('ALL EXCEPT JuniorDBA');
 - B. DBMS_SESSION.SET_ROLE('JuniorDBA');
 - C. SET ROLE ALL NONE EXCEPT JuniorDBA;
 - D. SET ROLE ALL EXCEPT JuniorDBA
 - E. You cannot change roles from within stored procedures.

Scenario

1. You have been tasked with ensuring that the following requirements are met in a production database where most objects accessed by users are located in the APP schema:
 - Ensure that all users can query any table in the APP schema as well as execute any procedure, package, or function in the APP schema.
 - Allow two junior DBAs (ToddR and YuryS) to be able to change users' passwords, quotas, and other settings. YuryS should also be allowed to create new users.
 - Allow your three main developers (SusanI, SteveG, and MylesB), as well as the junior DBAs mentioned previously, to create tables, views, procedures, and indexes in the APP schema. They should not be able to drop or truncate any existing tables or other objects in the APP schema.
 - Allow all users to insert, update, or delete data in tables and views in the APP schema only when accessing the database through the AppClient application software.

The turnover at the company is quite high and you want to ensure that the granting of permissions can be performed as efficiently as possible. How would you configure and assign the appropriate permissions to satisfy these requirements?

Lab Exercise

Lab 19-1 Creating, Managing, and Using Roles

1. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a new role called TrainingAppDev and assign it a password of "TrainPass." Grant the role privileges to create tables, views, procedures, and select data from all tables in the Student schema.

Create a second role called QueryUser and grant it only the privileges to SELECT from any table in the Student schema.
3. Create a user called ToddR with a password of your choice and make sure he has sufficient privileges to connect to the instance. Assign ToddR a quota of 10MB on the CERTDB tablespace and make CERTDB the user's default tablespace and TEMP the users temporary tablespace.
4. Grant the QueryUser role to ToddR. Also grant the TrainingAppDev role to ToddR but make sure that it is not enabled by default.

5. Connect to the instance as ToddR and determine which roles are enabled for your session.
6. Attempt to create a table called MyStudents that is a copy of the Students table in the Student schema. What happens and why?
7. Enable the TrainingAppDev role and attempt to create the table again. What happens and why?
8. Connect as the user SYSTEM and make the TrainingAppDev role the default for ToddR but ensure that the QueryUser role is not.
9. Connect to the instance as ToddR and attempt to create a table called MyInstructors that is a copy of the Student.Instructors table. What happens and why? Was the TrainingAppDev role enabled when ToddR connected to the instance?
10. In a second SQL*Plus session, connect to the instance as user Student with a password of ORACLE. Revoke the SELECT privilege on all tables in the Student schema from the TrainingAppDev role.
11. In the session connected as ToddR, query the Student.Instructors table. What happens and why? Disconnect and reconnect from the instance and query the table again. Do you notice any difference? Do NOT disconnect.
12. In another session (you can use the same one where you connected as user Student) connect to the instance as SYSTEM and drop the TrainingAppDev role.
13. In the session connected as ToddR, drop and re-create the MyStudents table. Were you successful?

Answers to Chapter Questions

Chapter Pre-Test

1. You can assign system privileges, object privileges, and other roles to roles. You can also assign role and system privileges to a role WITH ADMIN OPTION.
2. When designing security for your database and application, create task roles to perform specific tasks in the database like enter an order, or query order status, and then assign the required privileges to perform the task to the role. Create user roles to categorize users according to job function and responsibility. Assign task roles corresponding to the user's responsibility to the user roles.
3. The user can query the SESSION_ROLES data dictionary view to determine which roles are currently enabled for the session.
4. If you drop a role, all users that were granted the role and had it enabled will lose all privileges that the role was granted.

5. If you revoke a privilege from a role, users currently connected to the instance and that have the role enabled will continue to be able to use the privilege until they disconnect or disable the role. The revocation of the privilege from the user takes place the next time the role is enabled.
6. You can enable roles in a session using the SET ROLE command or by executing the DBMS_SESSION.SET_ROLE package procedure.
7. In order to assign object privileges to a role, you must be the owner of the object or have been granted privileges on the object WITH GRANT OPTION. Not even the DBA can assign object privileges unless they have been granted to him/her.
8. Roles can be created by the DBA or any user who has been granted the CREATE ROLE privilege, either directly or indirectly through a role. The same user that created the role can drop it, as well as any user with the DROP ANY ROLE privilege. Roles can be modified by the user who created them, or anyone with the ALTER ANY ROLE privilege. Any user can grant and revoke privileges to or from a role.
9. If you make a role that requires a password to be enabled a default role for a user, the user will never be prompted for the password and the role will automatically be enabled when the user connects to the instance. Oracle assumes that default roles should all be enabled, and any password requirements ignored.

Assessment Questions

1. **D.** It is not possible to grant object privileges to roles WITH GRANT OPTION, so this statement will fail. All other statements will succeed, assuming you have the privileges required to perform the operation.
2. **A, C.** While SharonT remains connected to the instance and the JuniorDBA role is enabled for her she is still able to issue the ALTER USER command to change passwords or anything else for a user. This is because revoking the role does not change the security context of the user until the role is disabled or the user disconnects. When she disconnects or issues the SET ROLE command, the JuniorDBA role's privileges will no longer be available.
3. **A, B, C, E.** Roles in an Oracle8i database can be authenticated by a password, the operating system or globally. Roles can also be created or modified so no password or authentication mechanism is needed to enable them.
4. **B.** If you want only the QueryRole role to be the default role when TomS connects to the instance, you would alter the user and specify QueryRole as the only default role. The SET ROLE command cannot be used to configure default roles for a user.
5. **E.** If you wanted TomS to have all privileges of the QueryRole and AppDevRole, you would need to issue a SET ROLE command from the application that enabled those roles. Option E enables all roles except JuniorDBA, but is the right answer because TomS has only been assigned three roles — JuniorDBA, QueryRole and AppDevRole.

6. **D.** Because you want your network and server administrators to be granted the role automatically, and because you probably do not have control over who these individuals are, you need to create roles that will be authenticated by the operating system — that is, IDENTIFIED EXTERNALLY.
7. **C.** If you want Oracle to support roles authenticated by the operating system, you need to ensure that the OS_ROLES initialization parameter is set to TRUE for the instance.
8. **D.** If you want the SysAdmin role to be the default role for all server and network administrators that have been granted it, you need to ensure that all of those users are in the group ORA_CERTDB_SYSADMIN_D at the operating system level. The group must have the ORA_<SID>_ prefix, followed by the role name, and ending in _D to indicate that the role will be a default role for the user.
9. **A.** You would execute the DBMS_SESSION.SET_ROLE package procedure to enable all roles for the user except the JuniorDBA role. You cannot issue a SET ROLE command from within an anonymous PL/SQL block.
10. **E.** You cannot enable or disable roles from within stored procedures.

Scenario

1. The requirements can be satisfied as follows:

- *Ensure that all users can query any table in the APP schema as well as execute any procedure, package, or function in the APP schema.*

To satisfy this requirement you would need to determine which objects exist in the APP schema and then grant the SELECT privilege on all tables and views, and the execute privilege on procedures, functions, and packages to PUBLIC.

You do not need to create any additional roles because all database users need to have access to database objects in the App schema.

- *Allow two junior DBAs (ToddR and YuryS) to be able to change users' passwords, quotas and other settings. YuryS should also be allowed to create new users.*

To satisfy these requirements you would create a new role called JuniorDBA and grant it the necessary privileges, and then grant the role to ToddR and YuryS. You would also need to grant the CREATE USER privilege directly to YuryS as well, and make a note of the action.

- *Allow your three main developers (SusanI, SteveG, and MylesB), as well as the junior DBAs mentioned previously, to create tables, views, procedures, and indexes in the APP schema. They should not be able to drop or truncate any existing tables or other objects in the APP schema.*

To satisfy this requirement you should create a role called AppDevRole, for example, and grant it the CREATE ANY TABLE, CREATE ANY VIEW, and CREATE ANY PROCEDURE system privileges. You would then assign the role to SusanI, SteveG, and MylesB, as well as the JuniorDBA role. By allowing users to be able to create objects in any schema, you are not permitting them to drop existing objects, so the second part of this requirement is satisfied.

- *Allow all users to insert, update or delete data in tables and views in the APP schema only when accessing the database through the AppClient application software.*

Similar to the first requirement, you would need to determine which tables and views exist in the APP schema and then REVOKE the INSERT, UPDATE and DELETE privileges from PUBLIC on those objects. You will also need to query the DBA_TAB_PRIVS view to determine if any privileges were explicitly granted to users and revoke them as well because revoking the privileges from PUBLIC will not reverse privileges explicitly granted to users. You would then create a role called AppDataUpdate, for example, and grant it the appropriate privileges. You would then grant the role to the users that will need to perform the changes to the data and make sure that it is not configured as a default role. The role would be enabled by using the SET ROLE command or executing the DBMS_SESSION.SET_ROLE procedure from within the AppClient application.

Lab Exercise

Lab 19-1

1. Connect to the CERTDB instance as user SYSTEM with a password of MANAGER.
2. Create a new role called TrainingAppDev and assign it a password of "TrainPass." Grant the role privileges to create tables, views, procedures, and select data from all tables in the Student schema.

Create a second role called QueryUser and grant it only the privileges to SELECT from any table in the Student schema.

```
SQL> CREATE ROLE TrainingAppDev
  2 IDENTIFIED BY TrainPass;
```

Role created.

```
SQL> CREATE ROLE QueryUser;
```

Role created.

```
SQL> GRANT CREATE TABLE, CREATE VIEW, CREATE PROCEDURE
```

```
2 TO TrainingAppDev;

Grant succeeded.

SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
SQL> GRANT SELECT ON Students
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON ScheduledClasses
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON Locations
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON Instructors
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON Courses
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON CourseAudit
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON ClassEnrollment
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL> GRANT SELECT ON BatchJobs
2 TO TrainingAppDev, QueryUser;

Grant succeeded.

SQL>
```

- 3.** Create a user called ToddR with a password of your choice and make sure he has sufficient privileges to connect to the instance. Assign ToddR a quota of 10MB on the CERTDB tablespace and make CERTDB the user's default tablespace and TEMP the user's temporary tablespace.

```
SQL> connect system/manager@certdb.delphi.bradsys.com
Connected.
```

```
SQL> CREATE USER ToddR IDENTIFIED BY password
 2  DEFAULT TABLESPACE CertDB
 3  TEMPORARY TABLESPACE Temp
 4  QUOTA 10M ON CertDB;
```

User created.

```
SQL> GRANT CREATE SESSION TO ToddR;
```

Grant succeeded.

```
SQL>
```

- 4. Grant the QueryUser role to ToddR. Also grant the TrainingAppDev role to ToddR but make sure that it is not enabled by default.**

```
SQL> GRANT QueryUser, TrainingAppDev
 2  TO ToddR;
```

Grant succeeded.

```
SQL> ALTER USER ToddR
 2  DEFAULT ROLE ALL EXCEPT TrainingAppDev;
```

User altered.

```
SQL>
```

- 5. Connect to the instance as ToddR and determine which roles are enabled for your session.**

```
SQL> connect toddr/password@certdb.delphi.bradsys.com
Connected.
```

```
SQL> SELECT * FROM SESSION_ROLES;
```

```
ROLE
-----
QUERYUSER
```

```
SQL>
```

- 6. Attempt to create a table called MyStudents that is a copy of the Students table in the Student schema. What happens and why?**

```
SQL> CREATE TABLE MyStudents
 2  AS SELECT * FROM Student.Students;
AS SELECT * FROM Student.Students
*
```

ERROR at line 2:

```
ORA-01031: insufficient privileges
```

```
SQL> SELECT * FROM SESSION_PRIVS;
```

```
PRIVILEGE
```

```
-----  
CREATE SESSION
```

```
SQL>
```

ToddR is unable to create the table because he does not have the CREATE TABLE privilege.

- 7. Enable the TrainingAppDev role and attempt to create the table again. What happens and why?**

```
SQL> SET ROLE TrainingAppDev IDENTIFIED BY TrainPass;
```

```
Role set.
```

```
SQL> SELECT * FROM SESSION_ROLES;
```

```
ROLE
```

```
-----  
TRAININGAPPDEV
```

```
SQL> CREATE TABLE MyStudents  
2 AS SELECT * FROM Student.Students;
```

```
Table created.
```

```
SQL> SELECT * FROM SESSION_PRIVS;
```

```
PRIVILEGE
```

```
-----  
CREATE SESSION  
CREATE TABLE  
CREATE VIEW  
CREATE PROCEDURE
```

```
SQL>
```

The TrainingAppRole, once enabled, allowed ToddR to create the table because the CREATE TABLE privilege was assigned to the role.

- 8. Connect as the user SYSTEM and make the TrainingAppDev role the default for ToddR but ensure that the QueryUser role is not.**

```
SQL> connect system/manager@certdb.delphi.bradsys.com  
Connected.
```

```
SQL> ALTER USER ToddR DEFAULT ROLE ALL EXCEPT QueryUser;
```

```
User altered.
```

```
SQL>
```

- 9.** Connect to the instance as ToddR and attempt to create a table called MyInstructors that is a copy of the Student.Instructors table. What happens and why? Was the TrainingAppDev role enabled when ToddR connected to the instance?

```
SQL> connect ToddR/password@certdb.delphi.bradsys.com
```

```
Connected.
```

```
SQL> CREATE TABLE MyInstructors
  2 AS SELECT * FROM Student.Instructors;
```

```
Table created.
```

```
SQL> SELECT * FROM SESSION_ROLES;
```

```
ROLE
```

```
-----
TRAININGAPPDEV
```

```
SQL>
```

The table creation succeeded because the TrainingAppRole was enabled when ToddR connected to the instance because it was configured as a default role for the user. No password needed to be specified because it was a default role, which bypasses password requirements.

- 10.** In a second SQL*Plus session, connect to the instance as user Student with a password of ORACLE. Revoke the SELECT privilege on all tables in the Student schema from the TrainingAppDev role.

```
SQL> REVOKE SELECT ON Students
  2 FROM TrainingAppDev;
```

```
Revoke succeeded.
```

```
SQL> REVOKE SELECT ON ScheduledClasses
  2 FROM TrainingAppDev;
```

```
Revoke succeeded.
```

```
SQL> REVOKE SELECT ON Locations
  2 FROM TrainingAppDev;
```

```
Revoke succeeded.
```

```
SQL> REVOKE SELECT ON Instructors
```

```
2 FROM TrainingAppDev;
```

Revoke succeeded.

```
SQL> REVOKE SELECT ON Courses
2 FROM TrainingAppDev;
```

Revoke succeeded.

```
SQL> REVOKE SELECT ON CourseAudit
2 FROM TrainingAppDev;
```

Revoke succeeded.

```
SQL> REVOKE SELECT ON ClassEnrollment
2 FROM TrainingAppDev;
```

Revoke succeeded.

```
SQL> REVOKE SELECT ON BatchJobs
2 FROM TrainingAppDev;
```

Revoke succeeded.

```
SQL>
```

- 11.** In the session connected as ToddR, query the Student.Instructors table. What happens and why? Disconnect and reconnect from the instance and query the table again. Do you notice any difference? Do NOT disconnect.

```
SQL> SELECT FirstName, LastName
2 FROM Student.Instructors;
```

FIRSTNAME	LASTNAME
Michael	Harrison
Susan	Keele
David	Ungar
Kyle	Jamieson
Lisa	Cross

```
SQL> connect toddr/password@certdb.delphi.bradsys.com
Connected.
```

```
SQL> SELECT FirstName, LastName
2 FROM Student.Instructors;
FROM Student.Instructors
*
```

```
ERROR at line 2:
ORA-00942: table or view does not exist
```

```
SQL>
```

ToddR could query the table even though Student revoked the SELECT privilege from the TrainingAppDev role because the privileges associated with the role do not get revoked from the user until he disconnects or disables the role. Re-connecting to the instance accomplished that and the second query failed.

- 12.** In another session (you can use the same one where you connected as user Student) connect to the instance as SYSTEM and drop the TrainingAppDev role.

```
SQL> DROP ROLE TrainingAppDev;
```

```
Role dropped.
```

```
SQL>
```

- 13.** In the session connected as ToddR, drop and re-create the MyStudents table. Were you successful?

```
SQL> DROP TABLE MyStudents;
```

```
Table dropped.
```

```
SQL> CREATE TABLE MyStudents
  2 AS SELECT * FROM Student.Students;
AS SELECT * FROM Student.Students
      *
```

```
ERROR at line 2:
ORA-00942: table or view does not exist
```

```
SQL> SELECT * FROM SESSION_ROLES;
```

```
no rows selected
```

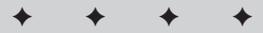
```
SQL>
```

The creation failed because the role was dropped, and also because ToddR no longer has SELECT privileges on the Student.Students table.

National Language Support

Being able to receive Oracle messages in their native language is important to users. Furthermore, a large database that is accessed by many users from different locales may need to support multiple languages. This part of the book, containing but a single chapter, introduces you to Oracle's National Language Support (NLS). You will learn what NLS features are available in Oracle8i, as well as how to change them for a user's session or for an individual command. Data dictionary views allowing a user to determine their NLS settings will also be explained.

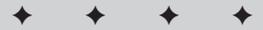
VI



In This Part

Chapter 20

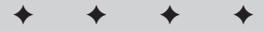
Using National Language Support



Using National Language Support

20

CHAPTER



EXAM OBJECTIVES

- ◆ Using National Language Support
 - Choose a character set and national character set for the database
 - Specify the language-dependent behavior using initialization parameters, environment variables, and the ALTER SESSION command
 - Use the different types of National Language Support (NLS) parameters
 - Explain the influence of language-dependent application behavior
 - Obtain information about NLS usage

CHAPTER PRE-TEST

1. What is the difference between the CHARACTER SET and NATIONAL CHARACTER SET clauses of the CREATE DATABASE command?
2. Where can you specify NLS parameters?
3. Can you have one set of NLS parameters configured and running for your session and still output data in a SQL query using a different set of NLS parameters?
4. What are two ways that NLS parameters settings can be changed for a user?
5. Which masking characters should you use if you want to represent numeric data with proper thousand and decimal separators (for example, 231 17.2256) using the local NLS settings?
6. How can you ensure that two sides of an expression in the WHERE clause of a SQL query use linguistic instead of binary comparisons?
7. What three values need to be specified when setting the NLS_LANG environment variable?
8. Which types of character sets cannot be used with the CHARACTER SET clause of the CREATE DATABASE command?
9. Is it possible to create two indexes on the LastName column of a table, one that would store and sort the data using a Spanish linguistic sort and the other using an English linguistic sort?
10. When performing a direct export, what should the value of the NLS_LANG parameter be set to?

If you are a user in Paris, France of a database that is hosted on an Oracle server in the United Kingdom, do you want to see Oracle error messages in French or in English? Well, the obvious answer would be that you want to see error messages from Oracle in your native language—French. This, and many other aspects of allowing users to have data returned in the format that they expect, is the function of National language Support (NLS) that was introduced in Oracle several versions ago and further expanded in Oracle8i. Oracle provides a number of ways to ensure that users will see data as they expect it, that a sort will present data in the proper order, and that date and currency information makes sense to a user anywhere in the world, no matter how the information is physically stored in the database.

Overview of National Language Support (NLS)

National Language Support in Oracle8i consists of a number of different elements. First among these is the ability to choose a character set and National Language (NLS) character set when creating an Oracle database. With few exceptions, the choice you make here cannot be changed later so you need to ensure that you pick the character set and NLS character set that is going to satisfy your requirements.

The second element of NLS support in Oracle deals with the client's NLS configuration and language, territory and character set the client is using when querying Oracle databases, as well as modifying the data within them. Oracle's network component (Net8) provides the ability to translate from the client's NLS character set to the database's NLS character set, and vice versa, to ensure that the database will always store the data as needed while the client and user will always see the data as they expect.

Oracle allows the user to make changes to the NLS settings for his or her session from the defaults that may be configured for them. Furthermore, you can make use of NLS system functions and masking characters to convert and display data in the proper NLS format.

Oracle's support for National Language elements includes the following features:

- ♦ Support for over 45 languages, 60 territories, 60 linguistic sort options, and many character sets.
- ♦ Support for both single-byte, multi-byte, and fixed-width character sets used in West European, East European, Middle Eastern, East Asian, and Southeast Asian languages.
- ♦ The ability to display date, currency, and other information in the format expected by the user's territory (that is, Canada uses the English language and the dollar, while the United Kingdom uses the English language and the pound as currency).

- ♦ The ability to sort data based upon the linguistic requirements of the client's language and territory, or a format specified by the user.
- ♦ Separate client and server NLS settings are supported with automatic conversion carried out by Net8 and the server.
- ♦ Ability to modify NLS settings on the fly as required ensuring that data is returned and formatted as needed by the user or application.
- ♦ Database utility and error messages are displayed in the client's configured language. Oracle's tools and other products provide support for 26 languages out of the box.
- ♦ Support for date and time formats according to ISO standards worldwide.
- ♦ Support for Julian, Gregorian, Japanese, Imperial, and Thai Buddha calendars so that local requirements can be satisfied.
- ♦ Proper masking of numeric data to satisfy local requirements (for example, in the United States the number 1234.56 is represented as 1,234.56 while in France the same number would appear as 1.234,56 to take into account French numeric formatting).
- ♦ Currency symbols and the accounting designations of credit and debit conform to the local territory NLS requirements to ensure that the user does not get presented confusing output.

The whole point of NLS in Oracle8i is to ensure that users see what they expect when they expect it no matter how the database is configured. This applies to sorted output, currency, date, time, error and status messages, and so on. NLS provides all of this functionality.

Choosing a Character Set and National Language Character Set for Your Database

Objective

Choose a character set and national character set for the database

The first step in implementing support for the languages that will be used to store database data is to pick a character set and National Language (NLS) character set for your database. This is not an option. As you saw in Chapter 4, the CREATE DATABASE command requires that both the CHARACTER SET and NATIONAL CHARACTER SET clauses have a value provided. You were also informed at that time to keep the character set and NLS character set fairly similar so as not to introduce inefficiencies into your database when processing data in NCHAR, NVARCHAR2, and NCLOB columns, which use the NLS character set. Of course, the \$64,000 question is which character set and NLS character set should you pick. The short answer is — the one that best fits your requirements (useless answer, but correct).

Character sets and encoding schemes

When choosing a character set, one of the things that you need to be aware of is what a character set and NLS character set actually means for the type of character data that can be stored and where it can be stored. In understanding this, you need to consider the data types supported by Oracle columns and the encoding schemes that are used by computers and operating systems to organize character data.

A character set encoding scheme determines how the character data will be physically stored in the database and how many bits and bytes will be needed to store the individual character. Essentially, the encoding scheme maps a binary value (or decimal, or hexadecimal) to a specific letter or character in the character set. A single encoding scheme can support many character sets but a character set will only use one of the available encoding schemes. The encoding scheme used will have an impact on the number of characters that can be supported by the character set, so choosing one appropriate to the language that will be stored in the database is important.

Oracle supports four types of encoding schemes:

- ♦ Single-byte (7 or 8 bit)
- ♦ Varying-width
- ♦ Fixed-width
- ♦ Unicode

When using a single-byte encoding scheme, each character in the character set takes up one byte of data. Oracle supports two types of single-byte encoding schemes: 7-bit (which supports 128 characters from decimal 0 to decimal 127, or 2⁷) or 8-bit (which supports 256 characters from decimal 0 to decimal 255, or 2⁸). The character sets that correspond to the 7-bit and 8-bit single-byte encoding schemes include US7ASCII (a 7-bit characters set that is the default character set for Oracle when creating a database), ISO 8859-1 Western European character set (WE8ISO8859P1, the recommended character set for databases whose data will contain character data in any of the Western European languages such as English, German, French, and so on), EBCDIC Code Page 500 8-bit West European (WE8ENC500 used on EBCDIC-based platforms such as AS400 or IBM mainframes), DEC 8-bit West European (WE8DEC used on OpenVMS and other Compaq/Digital platforms), and others. The character set available will determine which actual character that you would see on paper would appear on the screen when a specific numerical (that is, binary, decimal or hex) is stored in the database. For example, the hex values used to store character data when using the WE8ISO8859P1 character set would store characters displayed in Figure 20-1. A different character set would have different values for hex value D4, but all character sets have the same values for all characters whose hex value is less than or equal to 7F.

	0	1	2	3	4	5	6	7	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p	NBSP	°	À	Ð	à	ø
1	SOH	DC1	!	1	A	Q	a	q	±	±	Á	Ñ	á	ñ
2	STX	DC2	"	2	B	R	b	r	²	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	c	s	³	³	Ã	Ó	ã	ó
4	EOT	DC4	\$	4	D	T	d	t	´	´	Ä	Ô	ä	ô
5	ENQ	NAK	%	5	E	U	e	u	µ	µ	Å	Õ	å	õ
6	ACK	SYN	&	6	F	V	f	v	¶	¶	Æ	Ö	æ	ö
7	BEL	ETB	'	7	G	W	g	w	·	·	Ç	×	ç	+
8	BS	CAN	(8	H	X	h	x	,	,	È	Ø	è	ø
9	HT	EM)	9	I	Y	i	y	¸	¸	É	Ù	é	ù
A	NL	SUB	*	:	J	Z	j	z	¹	¹	Ê	Ú	ê	ú
B	VT	ESC	+	;	K	[k	¸	º	º	Ë	Û	ë	û
C	NP	FS	<	<	L	\	l	¸	»	»	Ì	Ü	ì	ü
D	CR	GS	-	=	M]	m	¸	¼	¼	Í	Ý	í	ý
E	SO	RS	>	>	N	^	n	¸	½	½	Î	ÿ	î	ÿ
F	SI	US	/	?	O	_	o	¸	¾	¾	Ï	ß	ï	ÿ
									DEL	¸	Ï	ß	ï	ÿ

Figure 20-1: The WE8ISO8859P1 character set displays characters according to the hex value stored. Use the top number/letter as the first part of the hex value and the side number/letter as the second. For example, hex 4A is a capital A whereas hex 7F is the DEL character/key.



Tip

The first 128 characters of any character set will always be the same as they represent the ASCII table of common computer-related characters. Only after hex 7F or decimal 128 do you have differences in character sets.

Varying-width multibyte character sets can have one or more bytes represent a single character. In The North American and Western Europe we don't normally see this as all of our characters can be represented by a single byte. However in Asian countries because the "alphabet" consists of a great many characters, you need more than one byte to represent them. In a varying width character set, a method needs to exist to determine whether the byte that is sent and stored is the whole character or only the first part of a multibyte character. Some multibyte encoding schemes use the value of the most significant bit (a portion of a byte—8 bits in a byte) to indicate if the whole character has been received or another byte follows with the rest of the character. Other systems use a shift-out or shift-in control code where a shift-out code will indicate that the bytes that follow are all part of multibyte characters, until a shift-in code is sent to turn off the stream. Oracle handles this appropriately for each character set and type of operating system used. Examples of varying-width multibyte character sets include Japanese Extended UNIX Code (JEUC) and Chinese GB2312-80 (CGB2312-80).

Fixed-width multibyte character sets are similar to varying-width in that they support multibyte characters, but will always occupy the same number of bytes for each character, regardless of the number of bytes the character physically requires. Fixed-width multibyte character sets are typically 16-bits (although 24-bit and 32-bit character sets are supported), which means that each character will occupy two or more bytes of physical storage. This can introduce storage inefficiencies since the letter "A" can be stored as a single byte, but when used with a fixed-width multibyte character set it will always occupy at least two bytes, thereby wasting at least a byte of storage. Examples of fixed-width multibyte character sets include JA16EUCFIXED

(a 16-bit fixed-width subset of the JA16EUC varying-width character set) and ZHT32TRISFIXED (a 32-bit fixed-width subset of the ZHT32TRIS varying-width character set). As a general rule, most fixed-width multibyte character sets will have the word `FIXED` as part of the character set name.

Unicode is a worldwide character encoding scheme that allows the representation of any character, including computer-specific technical and other symbols, as well as characters used for publishing and other special purposes. There are two Unicode standards that are supported by Oracle—Unicode 1.1 and Unicode 2.0. Unicode 2.0 currently supports the representation of 38,885 different characters. Unicode character sets can use any of the previously identified encoding schemes (single-byte, varying-width multibyte, and fixed-width multibyte). For example, the UCS2 Unicode character set is a fixed-width two-byte character set called “Universal Character Set—two-byte form.” The UTF8 (Universal Character Set Transformation Format) character set is a multibyte varying-width character set. Both UCS2 and UTF8 can support Unicode 1.1 or 2.0. Of the Unicode character sets available, UTF8 is recommended as it allows the representation of ASCII (single-byte) characters using a single byte instead of multiple bytes, thereby making storage more efficient.

Character sets and datatypes

When you create a database in Oracle8i, you need to choose two types of character sets by specifying a value for the `CHARACTER SET` and `NATIONAL CHARACTER SET` clauses of the `CREATE DATABASE` command. The reason for this is that character data using the character set specified by the `CHARACTER SET` clause will be stored in columns whose database is any of the following:

- ♦ `CHAR`
- ♦ `VARCHAR2`
- ♦ `LONG`
- ♦ `CLOB`

Columns with datatypes of `NCHAR`, `NVARCHAR2`, or `NCLOB` will have their data stored according to the character set specified by the `NATIONAL CHARACTER SET` clause of the `CREATE DATABASE` command. The reason for the difference is to allow for efficient storage of data and the possibility of having a single-byte `CHARACTER SET` and a multi-byte `NATIONAL CHARACTER SET`, if the database requirements warrant it. In all cases, the `CHARACTER SET` chosen determines the characters allowed in object names (tables, stored procedures, views, and so on) as well as the code for any stored programs (triggers, stored procedures, user-defined functions, packages, and so on).

You should note that for both the CHARACTER SET and NATIONAL CHARACTER SET clauses, you are able to specify a multi-byte character set, either varying-width or fixed width. However, where there are no restrictions on which character sets can be used for the NATIONAL CHARACTER SET clause, you cannot use a fixed-width multibyte character set with the CHARACTER SET clause of the CREATE DATABASE command. Specifically, the following character sets are not supported when specifying the CHARACTER SET clause: JA16EUCFIXED, ZHS16GBKFIXED, JA16DBCSFIXED, KO16DBCSFIXED, ZHS16DBCSFIXED, JA16SJISFIXED, ZHT32TRISFIXED, KO16KSC5601FIXED, ZHS16CGB231280FIXED, ZHT32EUCFIXED, ZHT16BIG5FIXED, and ZHT16DBCSFIXED.

Guidelines for choosing a character set and NLS Character set

Now that you have some idea of what character sets and NLS character sets are available and how the mapping of a hex value to a character works, you need to determine which CHARACTER SET and NATIONAL CHARACTER SET you need to use to support the character-based data requirements of your database. In making your decision keep these points in mind:

- ♦ The character set chosen will be used to store data in CHAR, VARCHAR2, LONG, and CLOB columns.
- ♦ The NLS character set chosen will be used to store data in NCHAR, NVARCHAR2, and NCLOB columns. National character sets are not supported for LONG columns as there is no NLONG datatype.
- ♦ The database CHARACTER SET and NATIONAL CHARACTER SET cannot be changed in versions prior to Oracle 8.1.6. Once you have selected the character set and NLS character set for a database, you are stuck with them and must export the database, re-create it, and then import the data back into the newly created database to affect a change in Oracle 8.1.5 or earlier.

In Oracle 8.1.6 or later, you can change the character set and NLS character set of a database, but only to a character set that is a strict superset of the existing character set or NLS character set. This means that if your original character set was US7ASCII and you wanted to change it to WE8ISO8859P1, you would be allowed to do so because the latter is a strict superset of the former. However, you cannot go from US7ASCII to JEUC (Japanese Extended UNIX Code) because the latter is not a strict superset of the former. In general, the rules of changing character sets essentially mean that you can go from a single-byte character set to another single-byte character set, but not to a multibyte varying or fixed width. Other combinations are also quite restrictive.



For more information on the rules for changing character sets and NLS character sets of a database, consult the *Oracle8i National Language Support Guide* in the Oracle documentation set.

In order to change a character set after database creation, you should perform a complete backup prior to doing so and then shutdown and restart the instance with `RESTRICTED SESSION` enabled. You can then issue the `ALTER DATABASE CHARACTER SET` or `ALTER DATABASE NATIONAL CHARACTER SET` command with the appropriate character set as a parameter. Following the change, you should shutdown the instance and once again perform a backup before re-starting it to allow users to access the data.

- ♦ If you do not specify a `CHARACTER SET` when issuing the `CREATE DATABASE` command, the character set will default to `US7ASCII`.
- ♦ If you do not specify a `NATIONAL CHARACTER SET` when issuing the `CREATE DATABASE` command, the value specified for `CHARACTER SET` will be used for the NLS character set for the database.
- ♦ If you do not plan to store data in more than one character set, make the `CHARACTER SET` and `NATIONAL CHARACTER SET` the same.
- ♦ If you specify a `NATIONAL CHARACTER SET` that differs widely from the `CHARACTER SET` specified, be aware that not all users may be able to see the data stored in the `NCHAR`, `NVARCHAR2`, and `NCLOB` columns. Using Unicode character sets helps to minimize this issue if you are running client application software on Windows-based computers.
- ♦ If you are concerned about the storage costs of your data, use varying-width multibyte character sets or NLS character sets. This will require a bit more overhead when storing and reading data, but will not consume as much disk space as fixed-width multi-byte character sets would.
- ♦ Use fixed-width multibyte character sets for NLS data to provide better performance. You cannot, in most cases, use fixed-width multibyte character sets for the `CHARACTER SET` of a database.
- ♦ If you are in North America or Western Europe and are not sure what character set and NLS character set to choose, use `WE8ISO8859P1` — it's the one that is designed to store most of the characters you will be seeing.
- ♦ Do not use `US7ASCII` unless you only intend to store American English character data. In general, don't use `US7ASCII` for anything.

As stated previously, figure out what the character data will look like in terms of languages that will be used and areas of the world where the database will be accessed. Based upon that information, determine whether you will need to support any other characters in `NCHAR`, `NVARCHAR2`, and `NCLOB` columns, and, if so, find out what languages that data will be created in. Based upon the information available, choose the right combination of `CHARACTER SET` and `NATIONAL CHARACTER SET`.

Specifying Language-Dependent Behavior in Oracle

Objective

Specify the language-dependent behavior using initialization parameters, environment variables, and the ALTER SESSION command

Configuring what characters will be supported by character-type columns in your Oracle database is only part of the equation. The other part deals with how Oracle will display error and other messages, perform sorting and other language-specific operations, and so on. By default, unless otherwise configured, Oracle will display all messages in English (actually the language it defaults to is “American”), and use data/time, currency, and numeric formats that correspond to the typical United States Settings. If you do not want the defaults to be used, you can configure Oracle to use specific NLS settings that make more sense to you.

Language-dependent behavior in Oracle can be specified in three places:

- ♦ **INIT.ORA File**—Specifying language settings at the Oracle initialization file for an instance will configure the default settings to be used for the instance. This does not mean that these settings will apply at the client level automatically, but rather if no other client settings are specified, then the instance-wide settings will be used.
- ♦ **Operating System Environment Variable**—If you want to configure default NLS settings for all client applications, you can set and configure the NLS_LANG environment variable to the language, territory and character set you want to use. When you install software using the Oracle Universal Installer, the machine-specific regional settings are checked and mapped to the appropriate NLS_LANG values on Windows-based computers. This will allow you to have proper NLS settings out of the box, but you can change what the Oracle Universal Installer picked for another value by changing the NLS_LANG environment variable.
- ♦ **ALTER SESSION Command**—The last, and most specific place, to configure NLS settings is by issuing an ALTER SESSION command to change the value of a specific NLS parameter. This can allow you to have different NLS settings when you connect to one application that you would have in another, or to change the NLS settings when running a report or PL/SQL procedure.

INIT.ORA parameters for configuring NLS behavior

When setting NLS characteristics for the instance, the values supplied to two INIT.ORA parameters determine the language-dependent behavior of a whole range of elements. The two parameters, what other INIT.ORA parameters they automatically configure, and their default values are shown in Table 20-1.

Table 20-1
Default Values for Language-Dependent
INIT.ORA Parameters in Oracle8i

<i>PARAMETER</i>	<i>Parameters Automatically Configured</i>	<i>Default Value</i>
NLS_LANGUAGE		AMERICAN
	NLS_DATE_LANGUAGE	AMERICAN
	NLS_SORT	BINARY
NLS_TERRITORY		AMERICA
	NLS_CURRENCY	\$
	NLS_ISO_CURRENCY	AMERICA
	NLS_DATE_FORMAT	DD-MON-YY
	NLS_NUMERIC_CHARACTERS	

The meaning of each of the initialization parameters is outlined in Table 20-2.

Table 20-2
Language-Dependent INIT.ORA Parameters in Oracle8i

<i>PARAMETER</i>	<i>Description</i>
NLS_LANGUAGE	Specifies the language that will be used to display error, and status messages.
NLS_DATE_LANGUAGE	Specifies the language that will be used when printing day and month names and their abbreviations, as well as the proper values for the symbols to represent AM, PM, AD, and BC.
NLS_SORT	Specifies the sort sequence to be used to sort data. Binary indicates that sorting will be done according to the hex value of the character. Other values allow you to perform language-specific sorts.
NLS_TERRITORY	Specifies country- or location-specific values for currencies, date format and the like. Unlike NLS_LANGUAGE, which actually sets the language for messages, NLS_TERRITORY does not configure any NLS settings itself but is used to set default values for other NLS parameters (shown in Table 20-1) as a group.

Continued


```

-----
nls_calendar                string
nls_comp                    string ANSI
nls_currency                string
nls_date_format             string
nls_date_language          string
nls_dual_currency          string
nls_iso_currency           string
nls_language                string ENGLISH
nls_numeric_characters     string
nls_sort                    string
nls_territory               string CANADA
nls_time_format             string
nls_time_tz_format         string
nls_timestamp_format       string
nls_timestamp_tz_format    string
SQL>

```

To see how this affects the display of the current date, you can issue the following query:

```

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
-----
01-07-27

SQL>

```

You will notice that the date format is not the default of DD-MON-YY but rather has been changed to YY-MM-DD, which is the official default date format for Canadian English.

Using the NLS_LANG environment variable to configure NLS settings

While the INIT.ORA parameters can be used to change the default behavior of the instance, in situations where users from many locations will need to access the database, or as might be the case in Canada and Europe, where different languages might be used by people in the same building, you can configure the default NLS settings for a client by using the NLS_LANG environment variable.

The NLS_LANG environment variable performs much the same kind of configuration as the NLS_TERRITORY and NLS_LANGUAGE Oracle initialization parameters do for an instance, but also adds information about the character set that is being used by the client. When setting INIT.ORA parameters, the character set for the database is already determined but on a client there is no guarantee that the database's

character set is best for the client computer. Oracle performs automatic conversion between character sets so having a character set on the client computer that is different from the database is not a problem.

The format of the NLS_LANG environment variable is as follows:

```
NLS_LANG=language_territory.characterset
```

For example, if you want a Windows-based client for use in England with the WE8ISO8859P1 character set, you would configure the environment variable in the AUTOEXEC.BAT file or in the System Properties for the computer to look like this:

```
NLS_LANG=English_England.WE8ISO8859P1
```

On a Unix computer using the Bourne shell, you will need to issue the following commands in your startup script or on the command line:

```
NLS_LANG=English_England.WE8ISO8859P1; export NLS_LANG
```

In addition to setting the NLS_LANG environment variable, you can also provide values for environment variables with the same name as the INIT.ORA parameters outlined in Table 20-2. The setting of an environment variable on the client side overrides the default INIT.ORA value provided on the server when the client executes SQL statements and receives messages from the server. Furthermore, you can also configure the following additional environment variables:

- ♦ **NLS_CREDIT** — Specifies the symbol to use for displaying numeric data that has been masked to include the CREDIT accounting notation.
- ♦ **NLS_DEBIT** — Specifies the symbol to use for displaying numeric data that has been masked to include the DEBIT accounting notation.
- ♦ **NLS_LIST_SEPARATOR** — Specifies the character to use to separate a value in a list of values returned from the server.
- ♦ **NLS_MONETARY_CHARACTERS** — Specifies the characters that indicate monetary units such as \$ for dollars and ¢ for cents.
- ♦ **NLS_NCHAR** — Specifies the character set for the client application to use when modifying NCHAR, NVARCHAR2, and NCLOB columns. If not specified, the client application will use the same character set as specified by the NLS_LANG parameter.

Changing NLS settings for a session

The most precise level of modifying Oracle's language-dependent behavior is to change NLS settings for a particular user session. This can be accomplished using the ALTER SESSION command or the DBMS_SESSION.SET_NLS package procedure.

The ALTER SESSION command allows you to change any of the parameters outlined in Table 20-2 or the environment variables in the preceding section. For example, to modify the date mask for your session, you can issue the following command:

```
ALTER SESSION SET NLS_DATE_FORMAT='DD Month, YYYY';
```

Querying the current date from the DUAL table would return the data as follows:

```
SQL> col sysdate format a20
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
-----
27 July      , 2001

SQL>
```

The other method that you can use to modify NLS settings for a session is the DBMS_SESSION.SET_NLS package procedure. This procedure is what you would use to change the NLS settings from within a PL/SQL program unit because the ALTER SESSION command cannot be used in PL/SQL. This allows an Oracle Forms-based application, for example, to make changes to NLS settings from within the application without requiring the user to know what the appropriate parameter is or how to execute the ALTER SESSION command.

The syntax of the DBMS_SESSION.SET_NLS procedure is as follows:

```
execute DBMS_SESSION.SET_NLS('NLS_parameter','value');
```

For example, to use the DBMS_SESSION.SET_NLS package procedure to set the date format to DD-MM-YYYY', you would execute the following command, and then query DUAL to test it:

```
SQL> execute DBMS_SESSION.SET_NLS('NLS_DATE_FORMAT','DD-MM-YYYY');

PL/SQL procedure successfully completed.

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
-----
27-07-2001

SQL>
```

Note that in specifying the format mask for the date, you need to include three single quotes on each side of the mask. This is because one single quote is needed for the mask, a second single quote is needed to specify the parameter, and a third single quote is needed to pass the single-quote enclosed mask to the process that

will actually change the `NLS_DATE_FORMAT`. As a general rule of thumb, ensure that you triple single-quote any parameters that need to be in single quotes when using the `DBMS_SESSION.SET_NLS` package procedure.

When using the `ALTER SESSION` command, or the `DBMS_SESSION.SET_NLS` package procedure, you can also specify additional NLS settings to change that are not available when using environment variables. This includes the following:

- ♦ **`NLS_TIME_FORMAT`** — Specifies the way in which time data will be displayed. The default is derived from the value of the `NLS_TERRITORY` parameter.
- ♦ **`NLS_TIMESTAMP_FORMAT`** — Specifies the way in which timestamp data will be displayed. Timestamp data includes both date and time data and its default is derived from the value of the `NLS_TERRITORY` parameter.
- ♦ **`NLS_TIME_TZ_FORMAT`** — Specifies the way in which time data will be displayed when including timezone information. The default is derived from the value of the `NLS_TERRITORY` parameter.
- ♦ **`NLS_TIMESTAMP_TZ_FORMAT`** — Specifies the way in which timestamp data that includes timezone information will be displayed. Timestamp data includes both date and time data and its default is derived from the value of the `NLS_TERRITORY` parameter.

Using NLS in Oracle8i

Objective

Use the different types of National Language Support (NLS) parameters

Explain the influence of language-dependent application behavior

Once you have configured NLS parameters for your instance, client environment or session, the behavior of a number of operations in Oracle changes. For example, when performing a sort, you will have data stored according to the value of the `NLS_SORT` parameter instead of the default of `BINARY`. You will also see error messages displayed on the language that you have configured for `NLS_LANGUAGE` and day and month names printed in the language specified by `NLS_DATE_LANGUAGE`. All this is designed to ensure that the user experience when using Oracle is what is expected instead of something that Oracle would force on the user.

Sorting

The `NLS_SORT` parameter determines the type of sorting that will be used whenever a user includes an `ORDER BY` clause in the SQL query, or a sort is required to take place in order to satisfy a SQL statement. The default sort order is determined by the value specified by the `NLS_LANGUAGE` sort parameter.

To test out what happens in sorting, create a table called `SortDemo` with a single column called `Letter` that will be used to hold each of the letters of the alphabet, as well as some accented and other letters. For example, if the `SortDemo` table

contained the following letters, you will notice that the accented characters in the table are all displayed at the end of the list of rows when the default NLS_SORT value of BINARY is set and the ORDER BY clause is used:

```
SQL> SELECT LETTER FROM SORTDEMO
2  ORDER BY LETTER;
```

```
LETTER
-----
a
b
c
d
e
f
g
h
i
j
k
... (letters from l to w)
x
y
z
à
ø
ñ
σ
ñ
—
û
è
ã
```

36 rows selected.

```
SQL>
```

If you query to determine how many letters sort lower than the letter “z,” and how many sort higher than “z” by issuing the following queries, you would get 25 for the number lower than “z” (z is the 26th) and 10 for the accented characters:

```
SQL> SELECT COUNT(*) FROM SORTDEMO
2  WHERE LETTER < 'z';
```

```
COUNT(*)
-----
25
```

```
SQL> SELECT COUNT(*) FROM SORTDEMO
```

```

2 WHERE LETTER > 'z';

COUNT(*)
-----
10

SQL>

```

Now, if you change the sort order to French, for example, and issued the same queries to determine how many letters sort higher or lower than “z,” you would get very different answers:

```

SQL> ALTER SESSION SET NLS_SORT=FRENCH;

Session altered.

SQL> SELECT COUNT(*) FROM SORTDEMO
2 WHERE LETTER < 'z';

COUNT(*)
-----
35

SQL> SELECT COUNT(*) FROM SORTDEMO
2 WHERE LETTER > 'z';

COUNT(*)
-----
0

SQL>

```

The reason for the change is that when using linguistic sorting, which changing `NLS_SORT` allows you to do, Oracle will place the accented characters in the proper linguistic sort order and not in the order specified by their binary or hex value.

You can also decide to perform a linguistic sort for a specific SQL statement by using the `NLSSORT` function. Many NLS parameters also have equivalent functions that allow you to format data according to a specific NLS settings, such as representing currency in US dollars even if the user is in France because the company reports all of its financial transactions in US dollars.

To test the affect of this, if you change the `NLS_SORT` parameter for the session back to `BINARY` and then issue a query to determine how many letters sort lower than “z” with and without the `NLSSORT` function, you should see the difference. Note that both the Letter column of the table and the letter “z” had to have the `NLSSORT` function used to ensure that a proper comparison took place, that is, “z” was properly placed in the French sort order before being compared to all other letters.

```
SQL> ALTER SESSION SET NLS_SORT=BINARY;

Session altered.

SQL> SELECT COUNT(*) FROM SORTDEMO
  2  WHERE LETTER < 'z';

  COUNT(*)
-----
        25

SQL> SELECT COUNT(*) FROM SORTDEMO WHERE
  2  NLSSORT(LETTER,'NLS_SORT=FRENCH') <
  3  NLSSORT('z','NLS_SORT=FRENCH');

  COUNT(*)
-----
        35

SQL>
```

Using NLSSORT to create linguistic indexes

One of the neat side effects on NLS in Oracle8i is the ability to create linguistic indexes by combining NLS and Oracle8i's new function-based indexing feature. For example, if you wanted to create an index on the SortDemo table that would index the Letter column according to the French linguistic sort, you can issue the following command:

```
SQL> CREATE INDEX French_Letter ON SortDemo
  2  (NLSSORT(Letter,'NLS_SORT=French'));

Index created.

SQL>
```

In order to make use of the index, you need to ensure that the NLS_COMP parameter is set to ANSI so that linguistic comparisons of operators in the WHERE clause take place instead of the default of BINARY comparison. To test this, issue the following commands:

```
SQL> ALTER SESSION SET NLS_LANGUAGE=FRENCH;

Session altered.

SQL> ALTER SESSION SET NLS_COMP=BINARY;

Session altered.

SQL> SELECT COUNT(*) FROM SORTDEMO
```

```

2 WHERE LETTER < 'z';

COUNT(*)
-----
25

SQL> ALTER SESSION SET NLS_COMP=ANSI;

Session altered.

SQL> SELECT COUNT(*) FROM SORTDEMO
2 WHERE LETTER < 'z';

COUNT(*)
-----
35

SQL>

```

Using NLS parameters in SQL functions

A number of SQL functions will allow you to make use of NLS parameters and masking characters to ensure that the display of data satisfies NLS requirements. The functions that make use of NLS parameters, and the parameters that can be used are outlined in Table 20-3.

Table 20-3
SQL Functions Supporting NLS Parameters in Oracle8i

<i>Function</i>	<i>NLS Parameters Supported</i>
TO_DATE	NLS_DATE_LANGUAGE NLS_CALENDAR
TO_NUMBER	NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY
TO_CHAR	NLS_DATE_LANGUAGE NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY NLS_CALENDAR NLS_UPPER

<i>Function</i>	<i>NLS Parameters Supported</i>
	NLS_LOWER
	NLS_INITCAP
NLSSORT	NLS_SORT

For example, to display the ClassID, CourseNumber, and StartDate of each class in the ScheduledClasses table using Italian date format, you can issue the following commands:

```
SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
```

```
SQL> SELECT ClassID, CourseNumber,
2 TO_CHAR(StartDate,'DD MON, YYYY',
'NLS_DATE_LANGUAGE=ITALIAN')
3 FROM ScheduledClasses
4* ORDER BY ClassID
SQL> /
```

```
CLASSID COURSENUMBER TO_CHAR(STAR
-----
50          100 06 GEN, 2001
51          200 13 GEN, 2001
53          100 14 FEB, 2001
```

```
SQL>
```

Oracle8i also supports the use of certain special marking characters when using NLS formatting of your data. The following number format masks are available to be used in SQL functions, and SQL*Plus, to format numbers for output:

- ♦ **D**—Used to represent the decimal separator in a numeric string.
- ♦ **G**—Used to represent the group, or thousand, separator in a numeric string.
- ♦ **L**—Used to represent the local currency symbol in a string.
- ♦ **C**—Used to represent the ISO currency symbol in a string.
- ♦ **U**—Used to represent the dual currency symbol (the Euro) in a string.

As well as numeric formatting masks, the following masking characters can also be used in date formatting to conform to NLS requirements:

- ♦ **RM** or **rm**—Used to specify the representation of a month's number using Roman numerals.

- ♦ **IW**—Used to represent the ISO week number when displaying a masked date.
- ♦ **IYYY, IYY, IY, and I**—The capital “I” is used to represent the ISO year when displaying a masked date.

To see how the format characters work, consider the following set of SQL statements where the `NLS_LANGUAGE` is set to Italian and `NLS_TERRITORY` is set to Italy. Masking a currency display using both the standard \$ and the local currency, with decimal and group separators will yield very different results, as you can see:

```
SQL> ALTER SESSION SET NLS_LANGUAGE=ITALIAN;
```

```
Session altered.
```

```
SQL> ALTER SESSION SET NLS_TERRITORY=ITALY;
```

```
Session altered.
```

```
SQL> col CourseName format a40
```

```
SQL> SELECT CourseName,
  2 TO_CHAR(RetailPrice,'$9,999.99') as "Dollars",
  3 TO_CHAR(RetailPrice,'L9G999D99') as "Lira"
  4* FROM Courses
SQL> /
```

COURSENAME	Dollars	Lira
Basic SQL	\$2,000.00	L.2.000,00
Advanced SQL	\$2,000.00	L.2.000,00
Performance Tuning your Database	\$4,000.00	L.4.000,00
Database Performance Basics	\$4,000.00	L.4.000,00
Database Administration	\$4,500.00	L.4.500,00
Backing up your database	\$3,000.00	L.3.000,00
Basic PL/SQL	\$2,500.00	L.2.500,00
Advanced PL/SQL	\$2,000.00	L.2.000,00
Using your PL/SQLskills	\$1,750.00	L.1.750,00

```
9 rows selected.
```

```
SQL>
```

Using Import/Export, SQL*Loader, and NLS

The Oracle Import and Export utilities (`IMP` and `EXP`), as well as `SQL*Loader`, will make use of the NLS settings of the client to determine whether or not conversion needs to take place between the format the data is stored in on the server and the format of the export file, or vice versa. The value of the `NLS_LANG` environment

variable on the client (that is, where IMP, EXP or SQL*Loader are being run) will determine the format of the data in the import, export, or loader file.

If you are performing a conventional Export or Import, the value of the NLS_LANG environment variable can be anything because Import or Export will properly convert from the character set on the client to the database character set, or vice versa. This means that when performing a conventional Export, the export file created by the Export utility will always be in the character set specified by the NLS_LANG environment variable on the computer where the Export was run. On an Import, the NLS_LANG environment variable should be set to the character set of the export file that is being imported into the database. The Import utility will convert from the session character set to the database character set automatically when performing a conventional import, but the file being imported is assumed to already be in that character set, so NLS_LANG needs to be appropriately set.

When performing a direct-path Export, the character set of the database and the character set of the client computer where the Export utility is being run need to be the same. This is because no conversion takes place from the database character set to the export file. For this to work properly, you need to ensure that you have set the value of the NLS_LANG variable to include the same character set as the database is using. On a direct-path Import, the data in the file being imported will be converted directly to the database character set during the import. Though it is not required, it is still recommended that the NLS_LANG environment variable for the client session where the Export utility is being run to match the character set of the file being imported.

The same rules hold true for SQL*Loader, except that SQL*Loader also provides the ability to specify the character set of the load file on the command line. The CHARACTERSET command-line parameter for the SQL*Loader executable is used to determine the character set of the data contained in the load file. This will be used to properly convert to the database character set to ensure that the data looks as expected when queried after the load. To perform a load and specify the character set of the load file, you would invoke SQL*Loader as follows:

```
C:\> sqlldr control=mycontroctl characterset=WE8IS08859P1
```

Getting Information about NLS Settings

Objective

Obtain information about NLS usage

A number of data dictionary views can be queried by the user or the DBA to determine the current NLS settings for the instance or session. These include the NLS_DATABASE_PARAMETERS, NLS_INSTANCE_PARAMETERS, NLS_SESSION_PARAMETERS and V\$NLS_PARAMETERS data dictionary views.

The `NLS_DATABASE_PARAMETERS` data dictionary view allows you to determine the currently running NLS parameters for the database. This includes those NLS parameters that have been assigned default values. To get a list of the defaults that will be inherited, unless overridden at the instance, the client or session level, execute the following query:

```
SQL> SELECT * FROM NLS_DATABASE_PARAMETERS;
```

PARAMETER	VALUE
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA
NLS_CURRENCY	\$
NLS_ISO_CURRENCY	AMERICA
NLS_NUMERIC_CHARACTERS	.,
NLS_CHARACTERSET	WE8ISO8859P1
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	DD-MON-RR
NLS_DATE_LANGUAGE	AMERICAN
NLS_SORT	BINARY
NLS_TIME_FORMAT	HH.MI.SSXF AM
NLS_TIMESTAMP_FORMAT	DD-MON-RR HH.MI.SSXF AM
NLS_TIME_TZ_FORMAT	HH.MI.SSXF AM TZH:TZM
NLS_TIMESTAMP_TZ_FORMAT	DD-MON-RR HH.MI.SSXF AM TZH:TZM
NLS_DUAL_CURRENCY	\$
NLS_COMP	BINARY
NLS_NCHAR_CHARACTERSET	WE8ISO8859P1
NLS_RDBMS_VERSION	8.1.7.0.0

18 rows selected.

```
SQL>
```

If you want to determine which NLS parameters were explicitly set at the instance level by including `INIT.ORA` NLS parameter, you can query the `NLS_INSTANCE_PARAMETERS` data dictionary view, as follows:

```
SQL> SELECT * FROM NLS_INSTANCE_PARAMETERS;
```

PARAMETER	VALUE
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA
NLS_SORT	
NLS_DATE_LANGUAGE	
NLS_DATE_FORMAT	
NLS_CURRENCY	

```

NLS_NUMERIC_CHARACTERS
NLS_ISO_CURRENCY
NLS_CALENDAR
NLS_TIME_FORMAT
NLS_TIMESTAMP_FORMAT
NLS_TIME_TZ_FORMAT
NLS_TIMESTAMP_TZ_FORMAT
NLS_DUAL_CURRENCY
NLS_COMP

```

```
15 rows selected.
```

```
SQL>
```

As you can see, only the `NLS_LANGUAGE` and `NLS_TERRITORY` parameters were explicitly specified in the `INIT.ORA` file for this instance. The other NLS parameters have NULLs in the `VALUE` column, so they were not specified and used the defaults.

To get the same set of information, you can also issue the following command in `SQL*Plus` or `Server Manager Line Mode`:

```
SQL> SHOW PARAMETER NLS;
```

NAME	TYPE	VALUE
nls_calendar	string	
nls_comp	string	
nls_currency	string	
nls_date_format	string	
nls_date_language	string	
nls_dual_currency	string	
nls_iso_currency	string	
nls_language	string	AMERICAN
nls_numeric_characters	string	
nls_sort	string	
nls_territory	string	AMERICA
nls_time_format	string	
nls_time_tz_format	string	
nls_timestamp_format	string	
nls_timestamp_tz_format	string	

```
SQL>
```

The main difference between the two is the output, although both the `NLS_INSTANCE_PARAMETERS` view and the `SHOW PARAMETER NLS` command provide the same information.

Finally, if you want to determine what the current settings of NLS parameters for your session are, you can query the `NLS_SESSION_PARAMETERS` view, or the

V\$NLS_PARAMETERS dynamic performance view, as shown in the following examples:

```
SQL> SELECT * FROM NLS_SESSION_PARAMETERS;
```

PARAMETER	VALUE
NLS_LANGUAGE	ENGLISH
NLS_TERRITORY	CANADA
NLS_CURRENCY	\$
NLS_ISO_CURRENCY	CANADA
NLS_NUMERIC_CHARACTERS	,
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	RR-MM-DD
NLS_DATE_LANGUAGE	ENGLISH
NLS_SORT	BINARY
NLS_TIME_FORMAT	HH24:MI:SSXFF
NLS_TIMESTAMP_FORMAT	RR-MM-DD HH24:MI:SSXFF
NLS_TIME_TZ_FORMAT	HH24:MI:SSXFF TZH:TZM
NLS_TIMESTAMP_TZ_FORMAT	RR-MM-DD HH24:MI:SSXFF TZH:TZM
NLS_DUAL_CURRENCY	\$
NLS_COMP	BINARY

15 rows selected.

```
SQL> SELECT * FROM V$NLS_PARAMETERS;
```

PARAMETER	VALUE
NLS_LANGUAGE	ENGLISH
NLS_TERRITORY	CANADA
NLS_CURRENCY	\$
NLS_ISO_CURRENCY	CANADA
NLS_NUMERIC_CHARACTERS	,
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	RR-MM-DD
NLS_DATE_LANGUAGE	ENGLISH
NLS_CHARACTERSET	WE8ISO8859P1
NLS_SORT	BINARY
NLS_TIME_FORMAT	HH24:MI:SSXFF
NLS_TIMESTAMP_FORMAT	RR-MM-DD HH24:MI:SSXFF
NLS_TIME_TZ_FORMAT	HH24:MI:SSXFF TZH:TZM
NLS_TIMESTAMP_TZ_FORMAT	RR-MM-DD HH24:MI:SSXFF TZH:TZM
NLS_DUAL_CURRENCY	\$
NLS_NCHAR_CHARACTERSET	WE8ISO8859P1
NLS_COMP	BINARY

17 rows selected.

```
SQL>
```

You should note that the V\$NLS_PARAMETERS view includes the values for NLS_CHARACTERSET and NLS_NCHAR_CHARACTERSET parameters that are not

returned by the NLS_SESSION_PARAMETERS view. Also, you can use the V\$NLS_PARAMETERS view when the instance is in a NOMOUNT or MOUNT state, whereas NLS_SESSION_PARAMETERS, and the other NLS_ views are only available when the instance is in an OPEN state.

One last view that might be of interest is V\$NLS_VALID_VALUES. If you need to determine what the valid values of the NLS_LANGUAGE, NLS_SORT, NLS_TERRITORY, and NLS_CHARACTERSET or NLS_NCHAR_CHARACTERSET parameters is (or even what character sets are supported by your version of Oracle), this view will provide the information. For example, if you wanted to get a list of all possible values for NLS_SORT that are available in your database, you can execute the following query:

```
SQL> col parameter format a25
SQL> col value format a25
SQL> SELECT * FROM V$NLS_VALID_VALUES
      2 WHERE PARAMETER = 'SORT';
```

PARAMETER	VALUE
SORT	BINARY
SORT	WEST_EUROPEAN
SORT	XWEST_EUROPEAN
SORT	GERMAN

... (data removed to save space)

SORT	ESTONIAN
SORT	ASCII7
SORT	JAPANESE
SORT	MALAY
SORT	PUNCTUATION
SORT	XPUNCTUATION
SORT	CANADIAN_FRENCH
SORT	VIETNAMESE
SORT	EEC_EURO
SORT	LATVIAN
SORT	BENGLI
SORT	XFRENCH
SORT	INDONESIAN
SORT	ARABIC_MATCH
SORT	ARABIC_ABJ_SORT
SORT	ARABIC_ABJ_MATCH
SORT	EEC_EUROPA3
SORT	CZECH_PUNCTUATION
SORT	XCZECH_PUNCTUATION
SORT	UNICODE_BINARY
SORT	EBCDIC
SORT	GENERIC_BASELETTER

69 rows selected.

SQL>

Key Point Summary

In preparing for the “Oracle8i DBA: Architecture and Administration” exam, please keep these points regarding National Language Support (NLS) in mind:

- ♦ NLS allows you to configure the database and the client software to display information and messages in a way that makes sense to the user based upon his or her location.
- ♦ NLS includes support for languages of messages, days and months, character set to store data required by the language being used, linguistic sorting, numeric, currency and date/time format display to adhere to local standards.
- ♦ Oracle supports single-byte 7-bit and 8-bit character sets, varying-width character sets, and fixed width character sets.
- ♦ You need to specify both a CHARACTER SET and NATIONAL CHARACTER SET when creating a database. Any of the character sets available can be used for the NATIONAL CHARACTER SET, but only single-byte and varying-width character sets can be used for the CHARACTER SET of the database.
- ♦ CHARACTER SET defaults to US7ASCII, while NATIONAL CHARACTER SET defaults to the value of CHARACTER SET. You should not use US7ASCII, but rather WE8ISO8859P1 if you are in North America or Western Europe.
- ♦ Once you choose a CHARACTER SET or NATIONAL CHARACTER SET for a database, you cannot change it unless you are running Oracle 8.1.6 or later and the character set you want to change it to is a strict superset of the one that is already specified.
- ♦ The CHARACTER SET chosen will be used to store data in CHAR, VARCHAR2, LONG, and CLOB columns. The NATIONAL CHARACTER SET chosen will be used to store data in NCHAR, NVARCHAR2, and NCLOB columns.
- ♦ Unless you have a specific reason not to do so, choose the same or a closely related character set for both the CHARACTER SET and NATIONAL CHARACTER SET clause of the CREATE DATABASE command.
- ♦ NLS parameters can be specified using INIT.ORA parameters for the instance, operating system environment variables for the client computer, and the ALTER SESSION command or DBMS_SESSION.SET_NLS package procedures for an individual user session.
- ♦ The effective settings will be those specified closest to the session — that is, INIT.ORA parameter values can be overridden by the environment variable settings, which can themselves be overwritten by session-level settings
- ♦ TO_CHAR, TO_NUMBER, TO_DATE and a few other SQL functions support the use of NLS parameters as well as masking characters.
- ♦ You can create linguistic indexes to sort data according to a specific NLS_SORT order by creating a function-based index using the NLSSORT function, and specifying the sort order you want the index to be created with.

- ♦ The `NLS_COMP` parameter determines whether `BINARY` or `ANSI` (that is, linguistic) comparisons will take place in the `WHERE` clause of a SQL statement.
- ♦ You can verify the current settings of NLS parameters for the database and session by querying the `NLS_DATABASE_PARAMETERS`, `NLS_INSTANCE_PARAMETERS`, and `NLS_SESSION_PARAMETERS` data dictionary views. You can also query the `V$NLS_PARAMETERS` dynamic performance view in a `NOMOUNT`, `MOUNT`, or `OPEN` state to get the values of NLS parameters for the current session.



STUDY GUIDE

The study guide portion of this chapter contains labs, assessment questions, and scenarios that you can use to reinforce your knowledge of National Language Support in Oracle8i and become better prepared for the exam.

Assessment Questions

1. Which of the following character sets cannot be used with the CHARACTER SET clause of the CREATE DATABASE command? (Choose the best answer.)
 - A. UTL8
 - B. JA16EUC
 - C. US7ASCII
 - D. JA16EUCFIXED
 - E. All of the above
 - F. None of the above
2. Which of the following character sets cannot be used with the NATIONAL CHARACTER SET clause of the CREATE DATABASE command? (Choose the best answer.)
 - A. UTL8
 - B. JA16EUC
 - C. US7ASCII
 - D. JA16EUCFIXED
 - E. All of the above
 - F. None of the above
3. You want to ensure that you can store Chinese language characters in your database. You decide to issue the CREATE DATABASE command with the following clauses:

```
CHARACTER SET WE8ISO8859P1  
NATIONAL CHARACTER SET ZHS16CGB231280
```

What datatype columns can you use to store your Chinese language characters? (Choose all correct answers.)

- A. CHAR
- B. NUMBER
- C. NVARCHAR2
- D. NLONG
- E. NCLOB

4. Your NLS_LANG environment variable is configured as follows:

```
NLS_LANG=German_Germany.WE8ISO8859P1
```

You want to ensure that dates get printed using the following mask:

```
DD-MON-YY
```

What do you need to do in order to ensure that dates print properly? (Choose two correct answers.)

- A. ALTER SESSION SET NLS_DATE_FORMAT='DD-MON-YY'
- B. ALTER SESSION SET NLS_DATE_MASK='DD-MON-YY'
- C. DBMS_SESSION.SET-NLS('NLS_DATE_MASK','DD-MON-YY')
- D. Use the TO_CHAR function with the NLS_DATE_MASK parameter specifying a mask of 'DD-MON-YY'
- E. DBMS_SESSION.SET-NLS('NLS_DATE_FORMAT','DD-MON-YY')

5. Your NLS_LANG environment variable is configured as follows:

```
NLS_LANG=German_Germany.WE8ISO8859P1
```

You have issued the following command:

```
ALTER SESSION SET NLS_DATE_LANGUAGE='English'
```

Assuming today is July 16, 2001, what will be the result of the following command? (Choose the best answer):

```
SELECT TO_CHAR(SYSDATE, 'DD-MONTH-YY') FROM DUAL;
```

- A. 16-JUILLET-01
- B. 16-JULI -01
- C. 16-JULY -01
- D. None of the above

6. To create a database with a character set other than US7ASCII, which environment variable must be configured? (Choose the best answer.)
- A. NLS_LANG
 - B. NLS_CHARACTER_SET
 - C. NLS_ORA33
 - D. ORA_NLS33
 - E. NLS_NCHAR_CHARACTERSET
7. If you want to determine what NLS parameters were modified using the INIT.ORA file, which of the following queries would you issue in SQL*Plus? (Choose all correct answers.)
- A. SELECT * FROM NLS_DATABASE_PARAMETERS
 - B. SELECT * FROM NLS_INSTANCE_PARAMETERS
 - C. SELECT * FROM NLS_SESSION_PARAMETERS
 - D. SELECT * FROM V\$NLS_PARAMETERS
 - E. SHOW PARAMETER NLS
8. If you want to determine which parameters were currently active for your user session, which of the following queries would you issue in SQL*Plus? (Choose two correct answers.)
- A. SELECT * FROM NLS_DATABASE_PARAMETERS
 - B. SELECT * FROM NLS_INSTANCE_PARAMETERS
 - C. SELECT * FROM NLS_SESSION_PARAMETERS
 - D. SELECT * FROM V\$NLS_PARAMETERS
 - E. SHOW PARAMETER NLS
9. If you configure the NLS_TERRITORY parameter for the instance by setting Oracle initialization parameters, which of the following other NLS settings will be automatically configured? (Choose all correct answers.)
- A. NLS_SORT
 - B. NLS_DATE_FORMAT
 - C. NLS_NCHAR_CHARACTERSET
 - D. NLS_NUMERIC_CHARACTERS
 - E. NLS_DATE_LANGUAGE

10. If you issue the `ALTER SESSION SET NLS_LANGUAGE=French` command in SQL*Plus, which of the following other NLS settings will be automatically configured? (Choose all correct answers.)
- A. `NLS_SORT`
 - B. `NLS_DATE_FORMAT`
 - C. `NLS_NCHAR_CHARACTERSET`
 - D. `NLS_NUMERIC_CHARACTERS`
 - E. `NLS_DATE_LANGUAGE`

Scenario

1. You have been hired as an Oracle consultant for a division within a department of the Canadian federal government. The division that hired you has its main headquarters in Toronto, where about 500 staff are located, and regional offices in Halifax (50 staff), Quebec City (200 staff), Montreal (100 staff), Ottawa (250 staff), Vancouver (200 staff), Calgary (150 staff) and Winnipeg (50 staff). The staff at all locations, with the exception of Quebec City, Montreal, and Ottawa, speak primarily English, while Quebec City staff speak primarily French. Staff in Ottawa and Montreal are split about 50/50 with regard to English and French-speaking individuals. In all cases, computers that are used by staff are configured to conform to language and other regional settings that are comfortable to the individual using it.

You have been asked to make recommendations on the server and client settings for a database that will be hosted on a Sun Solaris computer in the main office in Toronto. The database will be accessed by all staff in all locations on a semi-regular basis. No matter who accesses that database, information should be presented to those users in the official language of their choice and in the format expected.

As part of the database requirements, you will need to perform nightly exports of some of the data in the database to be imported into an analytical database that is located in Ottawa. You have been asked to use direct exports.

What character set and NLS character set do you need to specify for the database that will be created?

What environment variables will need to be configured on the Sun Solaris computer when you issue the `CREATE DATABASE` command? When you perform an Export?

How can you ensure that clients all receive error and status messages, as well as date, time, currency, and numeric formatting as they expect, based upon their computer's language and regional settings?

Lab Exercises

Lab 20-1 Changing NLS Parameters for the Instance

1. Using a text editor, modify the INIT.ORA file for the CERTDB database to include the following NLS parameters:

```
NLS_LANGUAGE=FRENCH  
NLS_TERRITORY=CANADA
```

2. Connect to the CERTDB instance using SQL*Plus or Server Manager Line Mode and shutdown and re-start your instance.
3. Execute a SQL query to check the NLS settings for the database and ensure that the change was successful.
4. Execute the following query and note its results:

```
SELECT SYSDATE FROM DUAL;
```

Lab 20-2 Modifying and Testing NLS Parameters for a Session

1. Connect to the CERTDB instance as user Student with a password of Oracle.
2. Execute a SQL query to determine your session-level NLS settings. Are they the same as for the database? Why or why not?
3. Modify your NLS settings so that language and territory are Italian and Italy, respectively.
4. Query the ScheduledClasses table for the ClassID, CourseNumber and StartDate of all classes in the table.
5. Alter the NLS_DATE_FORMAT parameter to change the format to display dates to “Dd fmMonth, YYYY” and issue your query in question 4 again.
6. Change the territory and language to be Canada and English, and re-execute the query in question 4. What happens?
7. Use the TO_CHAR system function to display the StartDate of classes in the ScheduledClasses table in French using the same date format as in question 5.
8. Play with NLS until you feel comfortable with its capabilities.

Answers to Chapter Questions

Chapter Pre-Test

1. The difference between the CHARACTER SET and NATIONAL CHARACTER SET clauses of the CREATE DATABASE command is that the CHARACTER SET clause specifies how data in CHAR, VARCHAR2, LONG and CLOB columns will be stored, whereas the NATIONAL CHARACTER SET clause specifies how data in NCHAR, NVARCHAR2, and NCLOB columns will be stored.
2. You can specify NLS parameters at the instance level, by setting NLS-related INIT.ORA parameters, for the client computer by using the NLS_LANG and other environment variables, and at the session level by setting NLS parameters using the ALTER SESSION command or DBMS_SESSION.SET_NLS package procedure.
3. It is possible in Oracle8i to have one set of NLS parameters configured and running for your session and still output data in a SQL query using a different set of NLS parameters. This is done by including the required NLS parameters in the TO_CHAR, TO_DATE or TO_NUMBER functions when formatting data. Unless you use these, and a few other functions, the session-level NLS settings will apply.
4. NLS parameters settings can be changed for a user by issuing the ALTER SESSION command with the appropriate parameter and value, or executing the DBMS_SESSION.SET_NLS package procedure.
5. If you want to represent numeric data with proper thousand and decimal separators (for example, 23117.2256) using the local NLS settings, you should use the G and D masking characters to represent the group (thousand) separator and decimal separator, as in the following example:

```
TO_CHAR(23117.2256, '99G999D9999')
```

6. If you want to ensure that two sides of an expression in the WHERE clause of a SQL query will use linguistic instead of binary comparisons, make sure that the value of the NLS_COMP parameter is set to ANSI and not BINARY. Depending on the NLS_LANGUAGE and NLS_TERRITORY values configured for your session, this may happen automatically. Query the NLS views to ensure that the NLS_COMP parameter is properly configured.
7. When setting the NLS_LANG environment variable you need to specify the language, territory and character set in the following format:

```
language_territory.characterset
```

as in the following example:

```
SET NLS_LANG=French_Canada.WE8ISO8859P1
```

8. Fixed-width multibyte character sets cannot be used with the CHARACTER SET clause of the CREATE DATABASE command. They can, however, be used with the NATIONAL CHARACTER SET clause of the CREATE DATABASE command. Only single-byte and varying-width multibyte character sets are supported for the CHARACTER SET clause of the CREATE DATABASE command.

9. It is possible to create two indexes on the LastName column of a table, one that would store and sort the data using a Spanish linguistic sort and the other using an English linguistic sort, by issuing the following commands:

```
CREATE INDEX Spanish_LastName ON MyTable
    (NLSSORT(LastName, 'NLS_SORT=Spanish'));

CREATE INDEX Binary_LastName ON MyTable
    (NLSSORT(LastName, 'NLS_SORT=BINARY'));
```

10. When performing a direct export the value of the NLS_LANG parameter on the client where the EXP utility is being run should be set to the same character set as the database. If the character set of the client computer is not the same as the database, the EXP utility will exit with an error.

Assessment Questions

1. **D.** You cannot use any multibyte fixed with character sets in the CHARACTER SET clause of the CREATE DATABASE command. The only character set listed that fit that description was JA16EUCFIXED.
2. **F.** Oracle places no restriction on which character sets can be used in the NATIONAL CHARACTER SET clause of the CREATE DATABASE command, so any of the character sets can be used and none will cause an error.
3. **C, E.** In order to start Chinese-language characters, you would need to create tables with columns that support the NATIONAL CHARACTER SET chosen. In the datatypes of columns listed, only NVARCHAR2 and NCLOB support the NLS character set — since there is no NLONG datatype. CHAR and VARCHAR2 columns will store characters that use the CHARACTER SET, which is Western European and not Chinese-language.
4. **A, E.** Running the ALTER SESSION command or the DBMS_SESSION.SET_NLS package procedure and passing either the NLS_DATE_FORMAT parameter with the proper mask will allow the dates to print as needed. There is no NLS_DATE_MASK parameter.
5. **C.** Because you used the ALTER SESSION command to change the date language to be English, the query will return a result in English and not German or French (which you did not specify anywhere).
6. **D.** In order to create a database with a character set other than US7ASCII, you need to ensure that the ORA_NLS33 environment variable is set to point to the location of the NLS data files on your hard disk.

7. **B, E.** In order to determine which NLS parameters were modified by using the INIT.ORA file, you can query either the NLS_INSTANCE_PARAMETERS view or issue the SHOW PARAMETER NLS command in SQL*Plus or Server Manager Line Mode.
8. **C, D.** To determine what the currently active NLS settings are for your session, you can query the NLS_SESSION_PARAMETERS or V\$NLS_PARAMETERS views.
9. **B, D.** Setting NLS_TERRITORY NLS parameter will also set default values for NLS_CURRENCY, NLS_ISO_CURRENCY, NLS_DATE_FORMAT, and NLS_NUMERIC_CHARACTERS.
10. **A, E.** Setting the NLS_LANGUAGE NLS parameter will also set default values for the NLS_SORT and NLS_DATE_LANGUAGE parameters.

Scenario

1. For this database, you will need to select a character set of WE8ISO8859P1, which supports all of the English and French characters that will be used to store data in the database. Although you do not know whether or not columns of datatype NCHAR, NVARCHAR2 or NCLOB will be used, as the only apparent requirement is English and French, you can also select WE8ISO8859P1 as the NATIONAL CHARACTER SET for the database, or possibly use one of the Unicode character sets such as UTF8.

In order to issue the CREATE DATABASE command to create a database with the WE8ISO8859P1 character set, the ORA_NLS33 environment variable must be set and configured to point to the location of the NLS files. If this is not done, the CREATE DATABASE command will fail with an error that it cannot find the NLS files.

When performing a direct-path Export from the database, you will need to ensure that the NLS_LANG environment variable is set to have the same character set as the database, otherwise the Export will fail. You will need to set it to one of the following:

```
SET NLS_LANG=English_Canada.WE8ISO8859P1
SET NLS_LANG=French_Canada.WE8ISO8859P1
```

Finally, to ensure that each individual's NLS settings are properly configured to return error and other messages in the proper language, you would need to set the NLS_LANG environment variable on the client computer (either in the registry or using proper commands for the OS being used) to one of the above values.

Lab Exercises

Lab 20-1

- Using a text editor, modify the INIT.ORA file for the CERTDB database to include the following NLS parameters:

```
NLS_LANGUAGE=FRENCH
NLS_TERRITORY=CANADA
```

- Connect to the CERTDB instance using SQL*Plus or Server Manager Line Mode and shutdown and re-start your instance.
- Execute a SQL query to check the NLS settings for the database and ensure that the change was successful.

```
SQL> col parameter format a25
SQL> col value format a25
SQL> SELECT * FROM NLS_DATABASE_PARAMETERS;
```

PARAMETER	VALUE
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA
NLS_CURRENCY	\$
NLS_ISO_CURRENCY	AMERICA
NLS_NUMERIC_CHARACTERS	.,
NLS_CHARACTERSET	WE8ISO8859P1
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	DD-MON-RR
NLS_DATE_LANGUAGE	AMERICAN
NLS_SORT	BINARY
NLS_TIME_FORMAT	HH.MI.SSXF AM

PARAMETER	VALUE
NLS_TIMESTAMP_FORMAT	DD-MON-RR HH.MI.SSXF AM
NLS_TIME_TZ_FORMAT	HH.MI.SSXF AM TZH:TZM
NLS_TIMESTAMP_TZ_FORMAT	DD-MON-RR HH.MI.SSXF AM TZH:TZM

NLS_DUAL_CURRENCY	\$
NLS_COMP	BINARY
NLS_NCHAR_CHARACTERSET	WE8ISO8859P1
NLS_RDBMS_VERSION	8.1.7.0.0

18 rows selected.

```
SQL> SELECT * FROM NLS_INSTANCE_PARAMETERS;
```

PARAMETER	VALUE
NLS_LANGUAGE	FRENCH
NLS_TERRITORY	CANADA

```

NLS_SORT
NLS_DATE_LANGUAGE
NLS_DATE_FORMAT
NLS_CURRENCY
NLS_NUMERIC_CHARACTERS
NLS_ISO_CURRENCY
NLS_CALENDAR
NLS_TIME_FORMAT
NLS_TIMESTAMP_FORMAT

```

```

PARAMETER                                VALUE
-----

```

```

NLS_TIME_TZ_FORMAT
NLS_TIMESTAMP_TZ_FORMAT
NLS_DUAL_CURRENCY
NLS_COMP

```

15 rows selected.

SQL>

4. Execute the following query and note its results:

```

SELECT SYSDATE FROM DUAL;

```

```

SQL> SELECT SYSDATE FROM DUAL;

```

```

SYSDATE
-----
28-JUL-01

```

SQL>

Lab 20-2

1. Connect to the CERTDB instance as user Student with a password of Oracle.
2. Execute a SQL query to determine your session-level NLS settings. Are they the same as for the database? Why or why not?

```

SQL> connect student/oracle@certdb.delphi.bradsys.com
Connected.
SQL> col parameter format a25
SQL> col value format a30
SQL> SELECT * FROM NLS_SESSION_PARAMETERS;

```

```

PARAMETER                                VALUE
-----
NLS_LANGUAGE                             ENGLISH
NLS_TERRITORY                             CANADA
NLS_CURRENCY                              $
NLS_ISO_CURRENCY                          CANADA
NLS_NUMERIC_CHARACTERS                    ,
NLS_CALENDAR                              GREGORIAN

```

```

NLS_DATE_FORMAT          RR-MM-DD
NLS_DATE_LANGUAGE       ENGLISH
NLS_SORT                 BINARY
NLS_TIME_FORMAT         HH24:MI:SSXFF
NLS_TIMESTAMP_FORMAT    RR-MM-DD HH24:MI:SSXFF
NLS_TIME_TZ_FORMAT      HH24:MI:SSXFF TZH:TZM
NLS_TIMESTAMP_TZ_FORMAT RR-MM-DD HH24:MI:SSXFF TZH:TZM
NLS_DUAL_CURRENCY       $
NLS_COMP                BINARY

```

15 rows selected.

SQL>

The NLS parameters are not the same as the database or instance settings probably because a different value for NLS_LANG has been specified in the registry of the Windows-based client, or by using the NLS_LANG environment variable.

- 3. Modify your NLS settings so that language and territory are Italian and Italy, respectively.**

```
SQL> ALTER SESSION SET NLS_LANGUAGE=ITALIAN;
```

Session altered.

```
SQL> ALTER SESSION SET NLS_TERRITORY=ITALY;
```

Session altered.

SQL>

- 4. Query the ScheduledClasses table for the ClassID, CourseNumber and StartDate of all classes in the table.**

```
SQL> SELECT ClassID, CourseNumber, StartDate
       2 FROM ScheduledClasses;
```

```

  CLASSID COURSENUMBER STARTDATE
-----
          50           100 06-GEN-01
          51           200 13-GEN-01
          53           100 14-FEB-01

```

SQL>

- 5. Alter the NLS_DATE_FORMAT parameter to change the format to display dates to “Dd fmMonth, YYYY” and issue your query in question 4 again.**

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='Dd fmMonth, YYYY';
```

Session altered.

```
SQL> SELECT ClassID, CourseNumber, StartDate
```

```

2 FROM ScheduledClasses;

CLASSID COURSENUMBER STARTDATE
-----
50          100 06 Gennaio, 2001
51          200 13 Gennaio, 2001
53          100 14 Febbraio, 2001

```

SQL>

Instead of using ALTER SESSION to modify the date format, you can also use TO_CHAR function, if this is a one-time query, as follows:

```

SQL> SELECT ClassID, CourseNumber,
2 TO_CHAR(StartDate,'Dd fmMonth, YYYY') "StartDate"
3 FROM ScheduledClasses;

```

```

CLASSID COURSENUMBER StartDate
-----
50          100 06 Gennaio, 2001
51          200 13 Gennaio, 2001
53          100 14 Febbraio, 2001

```

SQL>

6. Change the territory and language to Canada and English, and re-execute the query in question 4. What happens?

```
SQL> ALTER SESSION SET NLS_LANGUAGE=ENGLISH;
```

Session altered.

```
SQL> EXECUTE DBMS_SESSION.SET_NLS('NLS_TERRITORY','CANADA');
```

PL/SQL procedure successfully completed.

```
SQL> SELECT * FROM V$NLS_PARAMETERS;
```

PARAMETER	VALUE
NLS_LANGUAGE	ENGLISH
NLS_TERRITORY	CANADA
NLS_CURRENCY	\$
NLS_ISO_CURRENCY	CANADA
NLS_NUMERIC_CHARACTERS	,
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	RR-MM-DD
NLS_DATE_LANGUAGE	ENGLISH
NLS_CHARACTERSET	WE8ISO8859P1
NLS_SORT	BINARY
NLS_TIME_FORMAT	HH24:MI:SSXFF

```

NLS_TIMESTAMP_FORMAT      RR-MM-DD HH24:MI:SSXFF
NLS_TIME_TZ_FORMAT        HH24:MI:SSXFF TZH:TZM
NLS_TIMESTAMP_TZ_FORMAT   RR-MM-DD HH24:MI:SSXFF TZH:TZM
NLS_DUAL_CURRENCY         $
NLS_NCHAR_CHARACTERSET    WE8ISO8859P1
NLS_COMP                  BINARY

```

17 rows selected.

SQL>

- 7.** Use the `TO_CHAR` system function to display the `StartDate` of classes in the `ScheduledClasses` table in French using the same date format as in question 5.

```

SQL> SELECT ClassID, CourseNumber,
2      TO_CHAR(StartDate, 'Dd fmMonth, YYYY',
3             'NLS_DATE_LANGUAGE=FRENCH') "StartDate"
4      FROM ScheduledClasses;

```

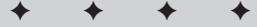
```

CLASSID COURSENUMBER StartDate
-----
50      100 06 Janvier, 2001
51      200 13 Janvier, 2001
53      100 14 Fàvrier, 2001

```

SQL>

- 8.** Play with NLS until you feel comfortable with its capabilities.



What's on the CD-ROM

This appendix provides you with information on the contents of the CD-ROM that accompanies this book.

There are 12 programs included on this CD. Some of the applications are full working versions of software while many others are evaluation or trial versions of some of the most useful Oracle utilities available.

The software included on the CD-ROM is:

- ◆ Adobe Acrobat Reader
- ◆ Bible Series Certification Test Engine and “Oracle8i: Architecture and Administration” sample exam from Hungry Minds, Inc.
- ◆ Oracle8i: Architecture and Administration (1Z0-023) Prep Exam from Self-Test Software (Trial Version)
- ◆ Aria ZIM and Oracle SAM from ZIM Technologies Inc. (Evaluation Version)
- ◆ ER/Studio from Embarcadero Technologies (Trial Version)
- ◆ DB/Artisan from Embarcadero Technologies (Trial Version)
- ◆ Knowledge Base for Oracle Administration 2000.2 from RevealNet, Inc (Trial Version)
- ◆ Instant Message for Oracle v2.4 from RevealNet, Inc (Trial Version)

Also included are scripts and source code examples from the book and an electronic, searchable version of the book that can be viewed with Adobe Acrobat Reader.

The CD-ROM does **not** include Oracle8i. If you do not have a copy of Oracle8i Enterprise Edition, which is recommended to successfully complete the labs, you can download a trial copy

from Oracle's Web site after joining the Oracle Technology Network at <http://technet.oracle.com>. The link to the download location is <http://technet.oracle.com/software/content.html>. You can also order a trial copy of Oracle8i Enterprise Edition for free (if you live in the United States) from the Oracle Store at <http://store.oracle.com>. Finally, if you do not want to download the software, or want to receive free updates for a year and CD-ROMs with the Oracle software on them, you can order a Technology Track through the Oracle Technology network at <http://technet.oracle.com/software/track.html>. Technology Tracks cost \$200 each (at the time of writing) and are available for WindowsNT/2000, Linux, and Sun SPARC platforms.

System Requirements

Make sure that your computer meets the minimum system requirements listed in this section. If your computer doesn't match up to most of these requirements, you may have a problem using the contents of the CD.

For Microsoft Windows NT (Service Pack 6a or later) or Windows 2000 (Service Pack 1 or later):

- ◆ PC with a Pentium II processor running at 300 MHz or faster
- ◆ At least 64MB of RAM
- ◆ At least 2GB of free hard disk space
- ◆ A CD-ROM drive

Although many of the software products included on the CD will work with Windows 98 or Windows Millennium Edition, if you want to install them and Oracle8i Enterprise Edition on the same computer, you will need either Windows NT or Windows 2000.

For Linux:

- ◆ PC with a Pentium II processor running at 300 MHz or faster
- ◆ At least 64MB of RAM
- ◆ At least 2GB of free hard disk space.
- ◆ A CD-ROM drive

Even though the software included on the CD-ROM runs in Windows environments only, you can make use of the scripts and sample code on a Linux computer that has Oracle8i Enterprise Edition for Linux installed.

Using the CD with Microsoft Windows

To install the items from the CD to your hard drive, follow these steps:

1. Insert the CD into your computer's CD-ROM drive.
2. If you have AutoPlay enabled, a window will appear with the following options: Install, Explore, eBook, Links, and Exit.

Install: Gives you the option to install the supplied software and/or the author-created samples on the CD-ROM.

Explore: Allows you to view the contents of the CD-ROM in its directory structure.

eBook: Allows you to view an electronic version of the book.

Links: Opens a hyperlinked page of Web sites.

Exit: Closes the autorun window.

To install the items from the CD to your hard drive, follow these steps:

1. Insert the CD into your computer's CD-ROM drive.
2. Click Start ⇨ Run...
3. In the dialog box that appears, type `d:\setup.exe`, where *d* is the letter of your CD-ROM drive.
4. Click OK.

Using the CD with Linux

To install the items from the CD to your hard drive, follow these steps:

1. Log in as root.
2. Insert the CD into your computer's CD-ROM drive.
3. Mount the CD-ROM.
4. Launch a graphical file manager.

What's on the CD

The CD-ROM contains source code examples, applications, and an electronic version of the book. Following is a summary of the contents of the CD-ROM arranged by category.

Source code

The scripts required to create the database used by the labs in each chapter of the book can be found in the SCRIPTS folder at the root of the CD-ROM. In the SCRIPTS folder, you may also find a SOLUTIONS folder, which will list the solutions for some of the more complex lab questions.

Applications

The SOFTWARE folder contains all of the trial and other software referenced earlier in this appendix. Each application is housed in its own folder within the SOFTWARE folder, as follows:

- ◆ **SELFTEST** — Oracle8i: Architecture and Administration (1Z0-001) Prep Exam from Self-Test Software (Trial Version). This program allows you to closely simulate the experience of writing the exam. The version provided is an evaluation of a version of the exam that you can purchase from the vendor and includes a smaller number of questions than the actual retail product.
- ◆ **ACROBAT** — Adobe Acrobat Reader. This is a fully functional version of the Adobe Acrobat Reader that can be used to read the electronic version of this book provided on the CD-ROM in the HUNGRYMINDS folder.
- ◆ **TEST** — Bible Series Certification Test Engine and “Oracle8i: Architecture and Administration” sample exam from Hungry Minds Inc. This test engine and exam include questions that can be used to prepare you for the exam. This software is a fully working version.
- ◆ **ARIAZIM** — Aria ZIM and Oracle SAM from ZIM Technologies Inc. (Evaluation Version). The Oracle Server Access Module (SAM) is in the ORASAM folder within the ARIAZIM folder of the CD-ROM. Zim is a powerful and flexible environment for developing and deploying all types of database applications. Zim’s entity-relationship model and fully integrated Object Dictionary permit progressive program development, whether your information processing system is among the simplest or the most complex. The straightforward, English-like syntax of the Application Development Language and its provision for customized user interfaces and programs, permit you to make any application easy to use. In both application development and database management, Zim provides the tools to get the job done.

Zim has the added capability of manipulating and retrieving data from third-party data sources as well as Zim’s own database. Client applications for SQL database servers are designed and developed as complete Zim application systems. The objective of the Zim product is 100 percent source code portability of applications including seamless access to databases being managed by SQL relational database management systems (SQL servers) supplied by independent vendors. In fact, a Zim Client-Server database application may consist of an arbitrary mixture of tables managed by an SQL server and entity sets and data relationships managed by Zim. The Oracle SAM allows you to use the power of ZIM with an Oracle database.

Please note that once you install the software, you will need to contact ZIM Technologies to activate it. Instructions are provided in the ARIAZIM folder on the CD-ROM.

- ♦ **ERSTUDIO** — ER/Studio from Embarcadero Technologies (Trial Version). ER/Studio is a data modeling application for logical and physical database design and construction. Its powerful, multi-level design environment addresses the everyday needs of database administrators, developers, and data architects who build and maintain large, complex database applications.

ER/Studio's progressive interface and processes have been logically organized to effectively address the ease-of-use issues that have plagued data modeling tools for the past decade. The application equips the user to create, understand, and manage the mission-critical database designs within an enterprise. It offers strong logical design capabilities, bi-directional synchronization of logical and physical designs, automatic database construction, and Java application generation, accurate reverse engineering of databases, and powerful HTML-based documentation and reporting facilities.
- ♦ **DBARTISAN** — DBArtisan is the leading database administration solution for managing Oracle, Sybase, Microsoft SQL Server, and IBM DB2 Universal databases. Its rich feature set helps database professionals maximize the availability, performance, and security of databases throughout the enterprise. DBArtisan enables DBAs to concurrently manage multiple databases from a single graphical console, creating dramatic productivity gains for both experienced and novice database professionals

DBArtisan automates and streamlines day-to-day administrative tasks such as creating user accounts, altering objects, migrating schemas between development, test and production, and much more. And because DBArtisan provides a common user interface for all major DBMS platforms, it enables businesses to lower costs and boost productivity by standardizing on a single database administration solution.
- ♦ **DBAKNOW** — Answer 70 to 80 percent of your system administration questions instantly with RevealNet's Knowledge Base for Oracle Administration. With over 2,800 topics of carefully designed expertise and code, you'll benefit from continuous day-to-day solutions for managing Oracle database objects, tuning and administering Oracle systems, and optimizing the networking environment.

From backup and recovery to network configuration and database tuning, you will find thousands of pre-built methods, insights and examples you need to get the job done quickly and accurately.
- ♦ **INSTMSG** — Instant Message for Oracle v2.4 from RevealNet, Inc (Trial Version). RevealNet's *Instant Messages* lookup is the fastest way to access over 25,000 Oracle messages — including Oracle8i error messages! Within seconds of encountering an error message, you'll have the full description with recommended actions.

Shareware programs are fully functional, free trial versions of copyrighted programs. If you like particular programs, register with their authors for a nominal fee and receive licenses, enhanced versions, and technical support.

Freeware programs are free, copyrighted games, applications, and utilities. You can copy them to as many PCs as you like — free — but they have no technical support.

GNU software is governed by its own license, which is included inside the folder of the GNU software. There are no restrictions on distribution of this software. See the GNU license for more details.

Trial, demo, or evaluation versions are usually limited either by time or functionality (such as being unable to save projects).

Electronic version of Oracle8i DBA: Architecture and Administration Certification Bible (Exam 1Z0-023)

The complete (and searchable) text of this book is on the CD-ROM in Adobe's Portable Document Format (PDF), readable with the Adobe Acrobat Reader (also included). For more information on Adobe Acrobat Reader, go to www.adobe.com.

Troubleshooting

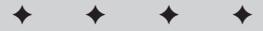
If you have difficulty installing or using the CD-ROM programs, try the following solutions:

- ◆ **Turn off any anti-virus software that you may have running.** Installers sometimes mimic virus activity and can make your computer incorrectly believe that it is being infected by a virus. (Be sure to turn the anti-virus software back on later.)
- ◆ **Close all running programs.** The more programs you're running, the less memory is available to other programs. Installers also typically update files and programs; if you keep other programs running, installation may not work properly.

If you still have trouble with the CD, please call Hungry Minds Customer Service phone number: (800) 762-2974. Outside the United States, call (317) 572-3994 or e-mail at techsupdum@hungryminds.com. Hungry Minds, Inc. will provide technical support only for installation and other general quality control items; for technical support on the applications themselves, consult the program's vendor or author.



Practice Exam



1. Which of the following operations can only be performed by a SYSDBA? (Choose the best answer.)
 - A. STARTUP
 - B. SHUTDOWN
 - C. RECOVER DATABASE
 - D. CREATE DATABASE
2. Which of the following is not part of an Oracle instance? (Choose the best answer.)
 - A. LGWR
 - B. init<sid>.ora
 - C. Data Dictionary Cache
 - D. Log Buffer
3. Which of the following connection requests is authenticated by the database? (Choose the best answer.)
 - A. CONNECT /
 - B. CONNECT sys/oracle
 - C. CONNECT internal
 - D. CONNECT sys/oracle AS SYSDBA
4. Which of the following statements is true? (Choose the best answer.)
 - A. An instance can only open one database at a time.
 - B. A database can only be opened by one instance.
 - C. A user can only establish one session with the server at a time.
 - D. A database can only have one tablespace.

5. Which of the following memory structures is not part of the SGA? (Choose the best answer.)
- A. Shared Pool
 - B. Log Buffer
 - C. Buffer Cache
 - D. PGA
6. Which of the following processes changes the values of blocks in the Database Buffer Cache? (Choose the best answer.)
- A. The User Process
 - B. PMON
 - C. The Server Process
 - D. SMON
 - E. DBWn
7. Which of the following database files store index segments? (Choose the best answer.)
- A. The datafiles
 - B. The log files
 - C. The control files
 - D. The parameter file
8. When a change is made to a row, which file is that change recorded in first? (Choose the best answer.)
- A. The datafile that contains the segment the row belongs to
 - B. The datafile(s) that contain the data dictionary
 - C. The datafile(s) that contain the rollback segment used by the transaction
 - D. The current online redo log file
9. What should you set the REMOTE_LOGIN_PASSWORDFILE parameter to in order to allow you to grant SYSDBA privileges to database users other than SYS and INTERNAL? (Choose the best answer.)
- A. EXCLUSIVE
 - B. NONE
 - C. SHARED
 - D. ENABLE

10. Which of the following tools can be used in order to create a password file on Windows NT? (Choose two best answers.)
- A. oradim
 - B. svrmgrl
 - C. sqlplus
 - D. orapwd
11. Which component of the Oracle Enterprise Manager is responsible for executing scheduled operations? (Choose the best answer.)
- A. The Console
 - B. The Management Server
 - C. The Agent
 - D. The Shared repository
12. Which of the following examples represent the correct syntax for setting the parameter values in the parameter file? (Choose two best answers.)
- A. db_block_buffers = 500
 - B. db_block_buffers: 500
 - C. rollback_segments = (rbs01, rbs02)
 - D. rollback_segments = (rbs01; rbs02)
13. In order to allow a user to start the database, you can grant them the _____ or _____ privileges. (Choose two best answers.)
- A. DBA
 - B. SYSDBA
 - C. SYSOPER
 - D. SYSADMIN
14. The MAX_DUMP_FILE_SIZE parameter controls the maximum size of which files? (Choose the best answer.)
- A. The Alert Log
 - B. All trace files, except the Alert Log
 - C. The Export dump file
 - D. The audit trail

15. What methods can you use in order to display the current parameter settings for the session? (Choose two best answers.)
- A. Use the SHOW PARAMETER command
 - B. Use the PARAMETER LIST command
 - C. Use the DBA_PARAMETERS view
 - D. Use the V\$PARAMETER view
16. In order to start the database using the parameter file 'C:\INITORCL.ORA', you need to issue which of the following statements? (Choose the best answer.)
- A. STARTUP IFILE = C:\INITORCL.ORA
 - B. STARTUP PFILE = C:\INITORCL.ORA
 - C. STARTUP PARFILE = C:\INITORCL.ORA
 - D. STARTUP @C:\INITORCL.ORA
17. Which of the following doesn't OFA specifically address? (Choose the best answer.)
- A. Efficient disk space utilization
 - B. Ease of administration
 - C. Database performance
 - D. Improved fault tolerance
18. Which of the scripts must you run after the creation of the database to create the data dictionary views? (Choose two correct answers.)
- A. sql.bsq
 - B. catalog.sql
 - C. catview.sql
 - D. catproc.sql
19. Which view would you query in order to find out which tables a specific view is based on? (Choose the best answer.)
- A. DBA_TABLES
 - B. V\$VIEWS
 - C. DBA_VIEWS
 - D. DBA_OBJECTS

- 20.** You need to find out how much of the space allocated to a table is above the high water mark. What is the correct syntax for calling the `UNUSED_SPACE` procedure in the `DBMS_SPACE` package? (Choose the best answer.)
- A. `EXECUTE DBMS_SPACE.UNUSED_SPACE(<parameters>)`
 - B. `RUN DBMS_SPACE.UNUSED_SPACE(<parameters>)`
 - C. `EXECUTE DBMS_SPACE(UNUSED_SPACE(<parameters>))`
 - D. `RUN {DBMS_SPACE.UNUSED_SPACE(<parameters>)}`
- 21.** You reorganized a table by exporting, dropping, and reimporting it into the database. Now the stored procedures that refer to that table show `INVALID` as their status. What do you need to do before they can be used again? (Choose the best answer.)
- A. You must manually recompile them before they can be used.
 - B. You must drop and recreate them.
 - C. You should have exported and reimported them together with the table. Now you have to repeat the export and import.
 - D. Nothing. They will be automatically recompiled the next time they are called.
- 22.** Database event triggers cannot be defined for which of the following event types? (Choose the best answer.)
- A. An `INSERT` into a table
 - B. A `STARTUP` or `SHUTDOWN`
 - C. A logon or logoff
 - D. A `CREATE`, `ALTER`, or `DROP` of a schema object
- 23.** Which of the following files is read when a database is mounted? (Choose the best answer.)
- A. The parameter file
 - B. The control file
 - C. The datafile(s)
 - D. The password file
 - E. The log files

24. Which of the following are not stored in the control file? (Choose the best answer.)
- A. The location of datafiles
 - B. The location of log files
 - C. The current log sequence number
 - D. The user names and passwords
25. Which of the following will happen if one of the three control files mentioned in the CONTROL_FILES parameter in the parameter file cannot be found during startup? (Choose the best answer.)
- A. The instance will not start.
 - B. The database will not mount.
 - C. The database will mount but not open.
 - D. The database will open because there are two other copies of the file available.
26. When a transaction commits, which file is the first one to receive the changes? (Choose the best answer.)
- A. The current online redo log file
 - B. The datafile containing the segment that was changed
 - C. The system datafile
 - D. The control file
27. Which of the following statements is true? (Choose the best answer.)
- A. A log file group must consist of at least two members.
 - B. All members of a log file group must be the same size.
 - C. All log files must be the same size.
 - D. Members of a log file group must be located on separate hard drives.
28. Which of the following files is/are not used by the LogMiner utility during analysis? (Choose the best answer.)
- A. Archived log files
 - B. Online redo log files
 - C. Control file
 - D. Dictionary file

29. Which of the following usually causes the highest degree of tablespace fragmentation? (Choose the best answer.)
- A. Rollback segments
 - B. Segments in a temporary tablespace
 - C. Sort segments in a permanent tablespace
 - D. Index segments
30. What can the ALTER TABLESPACE command be used for? (Choose the best answer.)
- A. Making a permanent tablespace temporary
 - B. Physically moving a datafile to a new location
 - C. Dropping a datafile
 - D. Changing the location of a datafile that belongs to SYSTEM tablespace
31. The statement CREATE TABLESPACE data03 DATAFILE 'C:\ORADATA\DATA03.DBF' REUSE will succeed if which of the following are true? (Choose two correct answers.)
- A. The C:\ORADATA directory exists.
 - B. The specified file already exists.
 - C. The specified file does not exist.
 - D. The DATA03 tablespace already exists.
32. Which of the following statements is true? (Choose the best answer.)
- A. All of the datafiles belonging to the same tablespace must be on the same disk.
 - B. All of the datafiles belonging to the same tablespace can be on different disks.
 - C. All of the datafiles belonging to the same tablespace must be on different disks.
 - D. A tablespace cannot have more than one datafile.
33. A tablespace cannot be taken offline if: (Choose the best answer.)
- A. It has any permanent objects in it.
 - B. It has any active rollback segments in it.
 - C. There is an active transaction modifying data in it.
 - D. It is a temporary tablespace.

- 34.** Which of the following is not a valid way to increase the size of a tablespace? (Choose the best answer.)
- A. By adding a datafile to it
 - B. By increasing the size of one or more of its datafiles
 - C. By setting AUTOEXTEND to ON for one or more of its datafiles
 - D. By setting AUTOEXTEND to ON for the tablespace
- 35.** Which of the following block space utilization parameters controls when a block becomes unavailable for INSERTs? (Choose the best answer.)
- A. INITRANS
 - B. MAXTRANS
 - C. PCTFREE
 - D. PCTUSED
- 36.** Which of the following segment types stores rows ordered by the PRIMARY KEY? (Choose the best answer.)
- A. An index
 - B. A table
 - C. A cluster
 - D. An index-organized table
- 37.** Which data dictionary view can you query in order to find the number of rows in a table? (Choose the best answer.)
- A. DBA_ROWS
 - B. DBA_TABLES
 - C. DBA_SEGMENTS
 - D. DBA_OBJECTS
- 38.** Which of the following cannot be specified for a rollback segment? (Choose the best answer.)
- A. PCTINCREASE
 - B. MINEXTENTS
 - C. MAXEXTENTS
 - D. OPTIMAL
 - E. NEXT

39. You are getting frequent errors similar to the following:

```
ORA-01562: failed to extend rollback segment number 62  
ORA-01650: unable to extend rollback segment RBS62 by 20 in  
tablespace RBS
```

What can you do in order to eliminate them? (Choose the best answer.)

- A. Recreate the rollback segments in RBS tablespace with bigger extents
- B. Increase MAXEXTENTS for all rollback segments in the RBS tablespace
- C. Increase OPTIMAL for all rollback segments in the RBS tablespace
- D. Increase the size of the RBS tablespace

40. You try to offline a tablespace and receive the following error:

```
ORA-01546: tablespace contains active rollback segment 'RBS0'
```

What do you need to do before you can take the tablespace offline? (Choose the best answer.)

- A. You must take all rollback segments in it offline.
- B. You must drop all rollback segments in it.
- C. You can keep the rollback segments in it online, but you must kill all sessions using them.
- D. You cannot take the rollback tablespace offline.

41. When creating a cluster, you need to specify the cluster key. Which of the following statements regarding the cluster key is true? (Choose the best answer.)

- A. It must correspond with the primary key for each of the tables in the cluster.
- B. It can only consist of one column.
- C. All tables in the cluster must have a corresponding column or columns with the same name and datatype as the cluster key.
- D. All tables in the cluster must have a column or columns with the same datatype and size as the cluster key.

42. Which of the following LOB data types stores character data in the database's national character set? (Choose the best answer.)

- A. BLOB
- B. CLOB
- C. NCLOB
- D. BFILE

43. When creating a temporary table, how can you specify that data in it should persist from one transaction to another? (Choose the best answer.)
- A. ON COMMIT DELETE ROWS
 - B. ON COMMIT PRESERVE ROWS
 - C. ON COMMIT RETAIN ROWS
 - D. ON COMMIT KEEP ROWS
44. You query the DBA_TABLES view in order to find the number of chained and migrated rows in a table. However, the CHAIN_CNT column is empty (NULL). What must you do in order to populate it? (Choose the best answer.)
- A. ALTER TABLE COMPUTE STATISTICS
 - B. ALTER TABLE ANALYZE
 - C. ANALYZE TABLE ESTIMATE STATISTICS
 - D. Nothing. A NULL value means that there are no chained or migrated rows in the table.
45. What type of an index should you use on a column with very low cardinality? (Choose the best answer.)
- A. B-tree
 - B. Reverse Key B-tree
 - C. UNIQUE
 - D. Bitmap
46. When do you use a high PCTFREE value for an index? (Choose the best answer.)
- A. When you expect that the underlying table will grow considerably over time
 - B. When you expect that inserts into the underlying table will have key values that fall within the current range
 - C. When you expect that inserts into the underlying table will have key values that fall outside of the current range
 - D. When you want the index to automatically maintain a large amount of free space in the leaf blocks
47. Which of the following options is not available for Reverse Key indexes? (Choose the best answer.)
- A. NOSORT
 - B. PCTFREE
 - C. UNIQUE
 - D. A Reverse Key index cannot be a composite index

48. Which of the following is not a valid state for a constraint? (Choose the best answer.)
- A. DISABLED NOVALIDATE
 - B. DISABLED VALIDATE
 - C. ENABLED NOVALIDATE
 - D. ENABLED VALIDATE
 - E. All of the above are valid states.
49. You try to enable a constraint and receive an error. How can you find out which rows are violating the constraint? (Choose the best answer.)
- A. Use the VIOLATIONS INTO clause
 - B. Use the EXCEPTIONS INTO clause
 - C. Use the LIST VIOLATIONS option
 - D. Query the V\$CONSTRAINT_VIOLATIONS view
50. Which of the following does not have a direct-path option? (Choose the best answer.)
- A. SQL*Loader
 - B. INSERT
 - C. Export
 - D. Import
51. After a load using SQL*Loader, in which of the following files will you find only the records that were rejected during the load? (Choose the best answer.)
- A. The control file
 - B. The log file
 - C. The bad file
 - D. The discard file
52. Before transporting a tablespace or a number of tablespaces using Oracle's new Transportable Tablespace feature, you want to check for potential problems. Which of the following packages can you use in order to check that the set is self-contained? (Choose the best answer.)
- A. DBMS_DESCRIBE
 - B. DBMS_TTS
 - C. DBMS_OUTPUT
 - D. DBMS_SPACE

53. In order for the CPU_PER_SESSION limit to be enforced, which of the following parameters do you need to enable? (Choose the best answer.)
- A. RESOURCE_LIMIT
 - B. AUDIT_TRAIL
 - C. TIMED_STATISTICS
 - D. SQL_TRACE
54. You need to force the users to change their passwords every 30 days. Which of the following profile limits should you use? (Choose the best answer.)
- A. PASSWORD_REUSE_MAX
 - B. PASSWORD_REUSE_TIME
 - C. PASSWORD_LIFE_TIME
 - D. PASSWORD_GRACE_TIME
55. How can you drop a profile that has been assigned to users? (Choose the best answer.)
- A. First, you must assign a different profile to all users. Then, you can drop the profile.
 - B. You can use the DROP PROFILE CASCADE command to drop the profile.
 - C. You cannot drop a profile that has been assigned to users.
 - D. You cannot drop a profile.
56. Which of the following does not form part of the COMPOSITE_LIMIT? (Choose the best answer.)
- A. IDLE_TIME
 - B. CONNECT_TIME
 - C. CPU_PER_SESSION
 - D. LOGICAL_READS_PER_SESSION
 - E. PRIVATE_SGA
57. You cannot use the ALTER USER command in order to (Choose the best answer.)
- A. Assign password restrictions to the user.
 - B. Unlock the user's account.
 - C. Reset the user's password.
 - D. Give the user a privilege.

58. Which view can you query in order to find which tablespace is a user's temporary tablespace? (Choose the best answer.)
- A. DBA_TABLESPACES
 - B. V\$TABLESPACE
 - C. DBA_USERS
 - D. DBA_PROFILES
59. If a user frequently deletes large numbers of rows from tables, how much quota does that user need in the rollback tablespace? (Choose the best answer.)
- A. Equivalent to the size of the largest rollback segment
 - B. Equivalent to the largest delete operation they are likely to perform
 - C. UNLIMITED
 - D. None
60. "SELECT ANY TABLE" is a _____ privilege and can be granted WITH _____ OPTION. (Choose the best answer.)
- A. System, GRANT
 - B. Object, GRANT
 - C. System, ADMIN
 - D. Object, ADMIN
61. Of the following statements, which are true? (Choose two best answers.)
- A. SYSADMIN and SYSOPER privileges can be granted WITH ADMIN OPTION.
 - B. SYSADMIN and SYSOPER privileges cannot be granted WITH ADMIN OPTION.
 - C. SYSADMIN and SYSOPER privileges can be granted to roles.
 - D. SYSADMIN and SYSOPER privileges cannot be granted to roles.
62. You granted a role to Kim with the appropriate option to allow Kim to re-grant it to others. Kim granted the role to three other users. You now need to revoke the role from Kim. What will be the effect of revoking the role? (Choose the best answer.)
- A. The role cannot be revoked from Kim.
 - B. The role will be revoked from Kim and all others Kim granted it to.
 - C. The role will be revoked from Kim only.
 - D. You have to manually revoke the role from all users Kim granted it to before revoking it from Kim.

63. What can the ALTER ROLE statement be used for? (Choose the best answer.)
- A. To assign a default role to a user
 - B. To add password protection to a role
 - C. To drop the role
 - D. To grant privileges to the role
64. How can you change the default sort sequence for your session? (Choose two best answers.)
- A. By using the ALTER SESSION command
 - B. By using the ALTER SYSTEM command
 - C. By using the DBMS_SESSION.SET-NLS procedure
 - D. By using the NLS_SESSION_PARAMETERS view
65. Which of the following NLS parameters can only be set as a client environment variable? (Choose the best answer.)
- A. NLS_LANGUAGE
 - B. NLS_TERRITORY
 - C. NLS_LANG
 - D. NLS_DATE_FORMAT

Sample Exam Answers

Well, you made it through the sample test. Now you can check your answers against this guide. After each correct answer, we included a reference to the chapter and section that explains the concepts tested in the question.

1. **D.** All the other operations listed can also be performed by a user with SYSOPER privileges. For more information, see Chapter 2, “Privileged Users.”
2. **B.** Oracle defines the instance as the SGA and the background processes used to access the database. The init<sid>.ora file contains the parameters used to start the instance, but it is not a part of the instance itself. The Log Writer process (LGWR) is one of the background processes that are part of an instance, so answer A is incorrect. The Data Dictionary Cache (Answer C) and the Log Buffer (Answer D) are part of the SGA and therefore part of the instance. For more information, see Chapter 1, “Oracle Instance.”

- 3. B.** The first request (Answer A) will use the operating system authentication (if configured) to connect to Oracle as a non-privileged user. The requests in answers C and D will use the password file or the operating system authentication to connect as a SYSDBA. The request in answer B will attempt to connect to Oracle as user sys without the SYSDBA privileges and will be authenticated by the database itself. By default, it will fail since the password for sys in the database is “change_on_install.” That is normally changed, but changing it to “oracle” is a poor choice. For more information, see Chapter 2, “Privileged Users,” and Chapter 17, “Authentication Mechanisms.”
- 4. A.** An instance can only open one database at a time. Answer B is incorrect because in the Parallel Server environment, the database is opened by multiple instances. Answer C is incorrect because a user can establish multiple concurrent sessions to the server, unless their profile restricts them. Answer D is incorrect because a database can (and should) have more than one tablespace. For more information, see Chapter 1, “Oracle Architecture Overview.”
- 5. D.** The PGA is allocated to each session and contains session-specific information. It is not part of the SGA. The other three memory allocations are major parts of the SGA. For more information, see Chapter 1, “Oracle Instance.”
- 6. C.** The Server Process changes the values of the data and rollback blocks in the Buffer Cache. From there, the dirty blocks are written to disk by DBWn, so answer E is incorrect. The PMON process (Answer B) is responsible for cleaning up after failed user connections. The SMON process (Answer D) performs instance recovery, coalesces free space in tablespaces, and reclaims space no longer used by temporary segments. The User Process (Answer A) is spawned when a user starts a tool such as SQL*Plus and establishes the connection to the server. For more information, see Chapter 1, “Processing SQL Statements.”
- 7. A.** Index and other segments are stored in the datafiles. The log files (Answer B) store a sequential record of changes to the data. The control files (Answer C) contain the physical structure of the database and the synchronization information, as well as a record of backups and archived logs. The parameter file (Answer D) is not part of the database at all; it contains parameters used by the instance on startup. For more information, see Chapter 1, “Oracle Database.”
- 8. D.** The log file is always the first file to receive the record of any change made to data. Even if DBWn is forced to write a dirty block to the datafile before a transaction commits, the LGWR will flush the log buffer to disk first, so answers A and C are wrong. Answer B is incorrect because the data dictionary is usually unaffected by changes made to rows; even if a change forces another extent to be allocated, the extent allocation is logged and recorded in the log file before the datafile. For more information, see Chapter 1, “Processing SQL Statements.”

9. **A.** If you set the value for the `REMOTE_LOGIN_PASSWORDFILE` to `NONE`, the only way to allow users to connect as `SYSDBA` is to make them members of appropriate groups at the operating system level, so answer B is incorrect. Answer C is wrong because the `SHARED` setting does not allow you to grant `SYSDBA` to any users other than `SYS` and `INTERNAL`. The `ENABLE` setting (Answer D) is invalid and will generate an error on instance startup. For more information, see Chapter 2, “Authenticating Privileged Users.”
10. **A and D.** The Oracle Password File utility (`orapwd`, Answer D) is available on both Windows NT and UNIX platforms and designed to create a password file. The `oradim` utility (Answer A, Windows NT only) is normally used to work with the Oracle service, but it can also create a password file. Server Manager (Answer B) and SQL*Plus (Answer C) cannot be used to create a password file, so these answers are wrong. For more information, see Chapter 2, “Authenticating Privileged Users,” and Chapter 4, “Creating a Database.”
11. **C.** The OEM utilizes a three-tier architecture: the Java-based GUI is the presentation layer, the Management Server is the middle tier providing shared services such as the security, job, event and discovery. The third tier consists of the managed nodes and databases. The Agent is the Oracle Enterprise Manager component that runs on the managed servers and executes the scheduled operations, allowing “Lights-out Management.” For more information, see Chapter 2, “Oracle Enterprise Manager.”
12. **A and C.** The syntax in answers B and D is invalid and will generate errors. For more information, see Chapter 3, “Parameter File (INIT.ORA).”
13. **B and C.** Users who have been granted `SYSDBA` or `SYSOPER` privileges can start up and shut down a database. The `DBA` role (Answer A) does not allow a user to perform those operations, so answer A is incorrect. There is no such thing as `SYSADMIN` in Oracle, so answer D is wrong, as well. For more information, see Chapter 2, “Privileged Users.”
14. **B.** The `MAX_DUMP_FILE_SIZE` parameter controls the size of trace files created by Oracle (for example, by enabling the `SQL_TRACE` parameter). The other files mentioned are not affected by this parameter. For more information, see Chapter 3, “Parameter File (INIT.ORA).”
15. **A and D.** For example, to see the current values for sort-related parameters, you could use the “`SHOW PARAMETER sort`” statement or the “`SELECT name, value FROM V$PARAMETER WHERE name like '%sort%'`” query. The `PARAMETER LIST` command (Answer B) does not exist, and neither does the `DBA_PARAMETERS` view (Answer C). In addition, answer C cannot possibly be correct since `DBA_` views get the data from the data dictionary, which does not hold parameter values. For more information, see Chapter 3, “Managing Oracle Instance Settings.”

- 16. B.** In order to specify the parameter file the instance should use on startup, you use the PFILE option. The IFILE keyword (Answer A) is used in a parameter file in order to include settings from another file into current configuration. The PARFILE parameter is used with utilities such as SQL*Loader and DBVERIFY in order to specify a large number of parameters. The “@” symbol (Answer D) is used to execute a script, not to specify a parameter file. All three of these invalid options will generate an error when used with the STARTUP command. For more information, see Chapter 3, “Starting Up and Shutting Down an Oracle Instance.”
- 17. B.** Optimal Flexible Architecture is Oracle’s recommendation for optimizing the locations of application, data, and log files for best performance, reliability, and ease of administration. It does not specifically address efficient space use since the total space used will be the same no matter where you locate the files. When you install Oracle, the structure created by the Universal Installer conforms to OFA; when you create a database, you should follow the OFA guidelines. For more information, see Chapter 4, “Oracle’s Optimal Flexible Architecture.”
- 18. B.** The catalog.sql script, which needs to be run after the creation of the database, creates the data dictionary (USER_, ALL_ and DBA_) views. The sql.bsq script (Answer A) is run during the database creation and creates the base tables; catproc.sql (Answer D) creates PL/SQL support and loads Oracle built-in packages into the data dictionary; catview.sql (Answer C) does not exist. For more information, see Chapter 5, “Creating Data Dictionary Views.”
- 19. C.** The DBA_VIEWS view has the ‘TEXT’ column that contains the definition of the view. Answer A is incorrect because the DBA_TABLES view deals with table details, not views. The V\$VIEWS view (Answer B) does not exist. The DBA_OBJECTS view does contain some information on views, but it does not contain their definitions. For more information, see Chapter 5, “Data Dictionary View Types.”
- 20. A.** In order to execute a procedure in a package, you should use the following syntax: “EXECUTE <package_name>.<procedure_name>(<parameters>).” The syntax in other answers is incorrect. For more information, see Chapter 5, “Stored PL/SQL Program Units.”
- 21. D.** When you export, drop, and re-import a table, you create a new object with the same name, which invalidates the compiled versions of the stored procedures that reference it. The procedures themselves are still fine since the name of the table does not change. The next time each one of these procedures is called, it will be automatically recompiled and become VALID. For more information, see Chapter 5, “View Stored Object Information.”
- 22. A.** DML on a schema object is not a database event. All other answers list events that can be monitored by using database event triggers, a new feature in Oracle8i. For more information, see Chapter 5, “Stored PL/SQL Program Units.”

23. **B.** When a database is mounted, the control file is read. The parameter file (Answer A) is read when the instance is started (NOMOUNT). The password file (Answer D) is used in order to authenticate privileged users, not to mount a database. The datafiles (Answer C) and the log files (Answer E) are read when the database is opened. For more information, see Chapter 3, “Starting Up and Shutting Down an Oracle Instance.”
24. **D.** The control file contains records about the physical structure of the database and synchronization information, not the user names and passwords. For more information, see Chapter 6, “Control File Contents.”
25. **B.** All copies of the control file that are mentioned in the parameter file must be present and valid in order for the database to mount. For more information, see Chapter 6, “Multiplexing the Control Files.”
26. **A.** When a transaction commits, the LGWR process flushes the contents of the redo log buffer to the current online redo log file. The changed blocks will eventually be written out to the datafile containing the segment by DBWn, but that may not happen until the next checkpoint. The system datafile is not likely to be affected by a change to a user table, but even if it is, the log will receive the changes first. The control file does not get affected by changes to segments. For more information, see Chapter 7, “Oracle Use of Redo Log Files.”
27. **B.** All members of a log file group are copies of each other, so they must be the same size. Although it is recommended that you multiplex the log files, a log file group can have only one member, so answer A is incorrect. There is usually no reason to create log file groups of different sizes, but Oracle does not require that all groups be the same size, so answer C is incorrect. Locating members of the same group on different drives makes sense (Answer D), but it is not required by Oracle. For more information, see Chapter 7, “Redo Log File Structure.”
28. **C.** During analysis, the LogMiner utility imports and analyzes online or archived log files (Answers A and B) and uses a dictionary file (Answer D) in order to resolve object ID’s to names. The control file is not used by the LogMiner utility. For more information, see Chapter 7, “Using Log Miner.”
29. **C.** If a user’s temporary tablespace is a permanent tablespace (by default, SYSTEM), sorting activity by that user that exceeds the SORT_AREA_SIZE parameter results in the creation of a sort segment in that tablespace. When the sort is complete, the segment is deallocated and the space is released, fragmenting the tablespace. Rollback and index segments do not cause as much fragmentation, so answers A and D are wrong. In a temporary tablespace (Answer B), the only segments that can be created are sort segments, and they cause no fragmentation because space is retained while the database is running and totally deallocated when the instance is shut down. For more information, see Chapter 9, “Planning the Location of Segments.”

- 30. A.** As long as the tablespace does not contain any permanent segments, it can be made temporary by using the ALTER TABLESPACE command. Answer B is incorrect because to relocate a file to a new location, you need to physically move the file by using the operating system tools and then use the ALTER TABLESPACE RENAME DATAFILE or ALTER DATABASE RENAME FILE statement to update the control file and the data dictionary with the new location of the file. Answer C is incorrect because you must use the ALTER DATABASE command in order to drop a datafile by itself. Answer D is incorrect because you can only use the ALTER TABLESPACE RENAME DATAFILE command when the database is open, the tablespace is offline, and the SYSTEM tablespace cannot be taken offline. In order to change the location of a datafile that is part of the SYSTEM tablespace, the database needs to be mounted, and you need to use the ALTER DATABASE RENAME FILE command. For more information, see Chapter 8, “Tablespaces.”
- 31. A and B.** When you create a tablespace, you need to specify at least one datafile. Oracle will not create directories for them—the specified directory must exist. Since you are not specifying the SIZE of the datafile but using the REUSE keyword instead, the specified datafile must already exist. If the file does not exist (Answer C), you will get an error. Trying to create two tablespaces with the same name (Answer D) will also result in an error. For more information, see Chapter 8, “Creating Tablespaces.”
- 32. B.** A tablespace can have multiple datafiles, so answer D is incorrect. These datafiles can be on different disks to increase performance, but they don’t have to be. For more information, see Chapter 8, “Creating Tablespaces.”
- 33. B.** A temporary tablespace (Answer D) can be taken offline if necessary. You can take a tablespace offline whether or not it has any permanent objects in it (Answer A). An active transaction modifying data in the tablespace will be allowed to complete by using a DEFERRED ROLLBACK segment created automatically when the tablespace goes offline, so answer C is incorrect. A tablespace that has an active rollback segment in it cannot be taken offline—you need to take the segment offline first. For more information, see Chapter 8, “Maintaining Tablespaces.”
- 34. D.** The AUTOEXTEND setting applies to datafiles, not tablespaces. All other options describe valid ways of increasing the size of a tablespace. For more information, see Chapter 8, “Resizing Tablespaces.”
- 35. C.** The INITRANS parameter (Answer A) controls the initial number of transaction slots allocated in the block header. The MAXTRANS parameter (Answer B) controls the maximum number of transaction slots allocated in the block header if necessary. The PCTFREE and PCTUSED parameters control when the block goes on and off the free list. When the free space in a block falls to PCTFREE percent, it becomes unavailable for inserts until the used space in it falls to PCTUSED percent (Answer D). For more information, see Chapter 9, “Block Space Utilization.”

- 36. D.** Rows in an index-organized table are stored sorted by the value of the primary key. An index (Answer A) does not store rows — it contains entries pointing to rows in a table. A table (Answer B) stores rows unordered; a cluster (Answer C) stores rows grouped (not ordered) by the value of the cluster key, which may or may not be the primary key of any of the clustered tables. For more information, see Chapter 9, “Types of Segments.”
- 37. B.** The `DBA_TABLES` view contains the `NUM_ROWS` column that shows the number of rows in a table. In order to get a non-NULL value, you need to analyze the table before querying this view. The `DBA_ROWS` view (Answer A) does not exist; the other two (Answers C and D) do not have this information. For more information, see Chapter 11, “Getting Information about Tables.”
- 38. A.** `PCTINCREASE` cannot be specified for a rollback segment and is always 0. The `MINEXTENTS` (Answer B) can be specified and has to be at least 2. `MAXEXTENTS` (Answer C) can (and should) be specified in order to prevent uncontrolled extension of a rollback segment. `INITIAL` (Answer E) and `NEXT` can be specified for rollback segments and should be the same. `OPTIMAL` (Answer D) can only be specified for rollback segments and controls when and how far they shrink. For more information, see Chapter 10, “Creating Rollback Segments.”
- 39. D.** The error indicates that no more space could be allocated in the tablespace, so its size needs to be increased either by resizing a datafile or by adding another one. Since it is the actual amount of available space that is a problem, modifying the storage settings for the rollback segments (Answers A, B, and C) is not likely to help significantly reduce the frequency of errors. For more information, see Chapter 10, “Troubleshooting Rollback Segments.”
- 40. A.** A tablespace cannot be taken offline until you offline all rollback segments in it. Dropping them (Answer B) is not necessary. Even if you have no transactions using any of the rollback segments in the tablespace, you will still need to take them offline before you can offline the tablespace, so answer C is incorrect. Answer D is wrong because all tablespaces except `SYSTEM` can be taken offline. For more information, see Chapter 10, “Troubleshooting Rollback Segments.”
- 41. D.** Each table in the cluster must have a column or columns with the same structure (type and size) as the cluster key. The names do not have to be the same, so answer C is incorrect. A cluster key can consist of multiple columns, so answer B is incorrect. Answer A is incorrect because the cluster key is independent of the primary keys of the tables in it. For more information, see Chapter 11, “Creating Tables on Clusters.”
- 42. C.** The `BLOB` (Answer A) is for storing binary data; the `CLOBs` (Answer B) store character data in the database character set; the `BFILE` data type (Answer D) is actually a pointer to an operating system file external to the database itself. For more information, see Chapter 11, “Built-in Oracle Data Types.”

- 43. B.** A temporary table will only keep data placed into it for the duration of each transaction by default. In order to make it keep the data placed into it for the duration of the session, you need to use the `ON COMMIT PRESERVE ROWS` option. Answer A is the exact opposite, so it is incorrect. Answers C and D are not valid options. For more information, see Chapter 11, “Temporary Tables.”
- 44. C.** In order to populate statistics-related columns in the `DBA_TABLES` view, the table must be analyzed by using the `ANALYZE TABLE` command. Answer D is wrong – if there are no chained or migrated rows in a table, it will show “0” in the `CHAIN_CNT` column after analysis. Answers A and B show invalid commands. For more information, see Chapter 11, “Getting Information about Tables.”
- 45. D.** Bitmap indexes are designed for low-cardinality (few distinct values) columns, where B-tree indexes are inefficient. Therefore, answers A and B are wrong. Answer C is incorrect because a unique index enforces that each key value occurs only once in the table, and therefore cannot work with a low-cardinality column. For more information, see Chapter 12, “Types of Indexes.”
- 46. B.** `PCTFREE` in an index is not maintained, so answer D is incorrect. Answer A is almost right but incomplete: you use a high `PCTFREE` when you want to leave a lot of free space in the index for future inserts that fall within the current range. If the new values will fall outside of the current range, a high `PCTFREE` is unnecessary no matter how much growth you anticipate. Answer C is the opposite of the correct answer: if the new values are expected to fall outside of the current range, you can set `PCTFREE` to a low number. For more information, see Chapter 12, “Types of Indexes.”
- 47. A.** A Reverse Key index cannot be created using the `NOSORT` option. It can, however, be unique, so answer C is incorrect. It can also be a composite index, so answer D is wrong, too. You can specify `PCTFREE` for a Reverse Key index, so answer B is incorrect. For more information, see Chapter 12, “Types of Indexes.”
- 48. E.** The `DISABLED NOVALIDATE` state (Answer A) means that the constraint is completely disabled. The `DISABLED VALIDATE` state means that although the constraint is disabled, no changes are allowed on the constrained column, so the integrity of existing data is preserved. `ENABLED NOVALIDATE` (Answer C) means that the new data is checked for compliance with the constraint, but violations are allowed to exist in the existing data. `ENABLED VALIDATE` (Answer D) requires compliance for both existing and new data. For more information, see Chapter 13, “Constraint States.”
- 49. B.** In order to find and correct violations of the constraint, you should use the `EXCEPTIONS INTO` clause while enabling it and query the `EXCEPTIONS` table created by running the `utlexcpt.sql` script. All other answers list nonexistent clause, option, and view. For more information, see Chapter 13, “Constraint States.”

- 50. D.** There is no such thing as a direct import. You can use the `DIRECT=Y` option for SQL*Loader (Answer A) and Export (Answer C) to speed up the process. In order to invoke a direct-path INSERT (Answer B), you need to use the `/*+APPEND*/` hint. For more information, see Chapter 14, “Loading Data Using Direct-Load INSERTs”, “SQL*Loader Data Load Paths”, and Chapter 15, “Export Types.”
- 51. C.** Records rejected by SQL*Loader or by Oracle Server during a load go into the bad file. The control file (Answer A) contains the instructions for the load, the data description, and selection criteria. The log file (Answer B) contains the summary statistics and details for the load. The discard file contains records that did not satisfy any of the selection criteria and were not loaded because of that. For more information, see Chapter 14, “Files Used by SQL*Loader.”
- 52. B.** The `DBMS_TTS` package can be used in order to check whether the tablespace set is self-contained and whether it contains objects that cannot be exported. The `DBMS_DESCRIBE` package (Answer A) is used to describe the arguments of a stored procedure. The `DBMS_OUTPUT` package (Answer C) is used to display information or accumulate it in a buffer. The `DBMS_SPACE` package (Answer D) provides segment space usage information. For more information, see Chapter 15, “Transportable Tablespaces.”
- 53. A.** Until the `RESOURCE_LIMIT` parameter is set to `TRUE`, all resource use limits set in a user’s profile are ignored. The `AUDIT_TRAIL` parameter (Answer B) configures where the audit records are created. The `TIMED_STATISTICS` parameter (Answer C) is useful for tuning Oracle and enables monitoring of time-based statistics. The `SQL_TRACE` parameter enables tracing of statements for diagnostic purposes. For more information, see Chapter 16, “Resource Management.”
- 54. C.** The `PASSWORD_LIFE_TIME` parameter controls the expiry of a user’s password. The `PASSWORD_REUSE_MAX` parameter (Answer A) limits the number of times a user can reuse the same password. The `PASSWORD_REUSE_TIME` parameter can be set to prevent a user from reusing the same password within a given number of days. The `PASSWORD_GRACE_TIME` parameter (Answer D) allows the user to use their old password for a number of days after it expires. For more information, see Chapter 16, “Password Management.”
- 55. B.** If you need to drop a profile that has been assigned to users, you can use the `CASCADE` option. All users the profile was assigned to will now have the `DEFAULT` profile assigned to them. This makes answers A, C, and D incorrect. For more information, see Chapter 16, “Dropping Profiles.”
- 56. A.** The `COMPOSITE_LIMIT` is a weighted sum of the limits in answers B, C, D, and E. In order to change the weight (importance) of each, you can use the `ALTER RESOURCE COST` command. For more information, see Chapter 16, “Resource Management.”

- 57. D.** In order to give a privilege to a user, you use the GRANT command. Answer A is incorrect because you can control the user's password limits with profiles and use the ALTER USER command to assign the profile to the user. In order to unlock the user's account (Answer B) or reset their password (Answer C), you use the ALTER USER command, as well. For more information, see Chapter 17, "Modifying Users," and Chapter 18, "Granting System Privileges" and "Granting Object Privileges."
- 58. C.** The TEMPORARY_TABLESPACE column in the DBA_USERS view shows the user's temporary tablespace. Since the DBA_TABLESPACES (Answer A) and the V\$TABLESPACE (Answer B) views have one row per tablespace, they cannot possibly show each user's temporary tablespace. The DBA_PROFILES view (Answer D) shows information on profiles that exist in the database, not users. For more information, see Chapter 17, "Getting Information about Users."
- 59. D.** A user needs a quota in order to create objects and have space allocated to them. Since the user does not own the rollback segments, the user does not need a quota in the rollback tablespace. For more information, see Chapter 17, "Overview of User Management" and "Quotas."
- 60. C.** The SELECT ANY TABLE is a system privilege since it allows a user to perform a specific operation, not to access a specific object. System privileges can be granted using the ADMIN OPTION, not GRANT OPTION. For more information, see Chapter 18, "Types of Privileges."
- 61. B and D.** SYSADMIN and SYSOPER privileges cannot be granted to roles or WITH ADMIN OPTION. For more information, see Chapter 18, "SYSDBA and SYSOPER Special System Privileges."
- 62. C.** In order to allow the user to grant the role to others, you had to grant it WITH ADMIN OPTION. When the role is revoked, the revoke does not cascade, so all the other users Kim may have granted it to will keep the role. This makes answer B incorrect. Answer A is incorrect because a role can be revoked from any user, whether or not that user has it WITH ADMIN OPTION. Answer D is incorrect because a role can be revoked from a user whether they have granted it to others or not. For more information, see Chapter 19, "Managing Privileges with Roles."
- 63. B.** In order to change authorization requirements for a role (password protection, for example), you use the ALTER ROLE command. To assign a default role to a user (Answer A), you use the ALTER USER command. To drop a role, you use the DROP ROLE command. To grant privileges or other roles to a role, you use the GRANT command. For more information, see Chapter 19, "Modifying Roles."

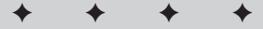
- 64. A and C.** In order to change the default sort sequence for a session, you can use the ALTER SESSION SET NLS_SORT statement or the DBMS_SESSION.SET_NLS procedure. Answer B is incorrect because ALTER SYSTEM cannot be used to modify the NLS_SORT parameter. Answer D is incorrect because NLS_SESSION_PARAMETERS is a read-only view that shows you the current values of NLS parameters. For more information, see Chapter 20, “Specifying Language-Dependent Behavior in Oracle.”
- 65. C.** The NLS_LANG parameter set as a client environment variable is used to specify the default language, territory, and character set for the client session. All other parameters listed can be set by using ALTER SESSION. For more information, see Chapter 20, “Specifying Language-Dependent Behavior in Oracle.”



Objective Mapping

The “Oracle8i: Architecture and Administration” exam has a clear set of defined objectives. This book covers all of the material that is required for the exam. This chart is intended to assist you in ensuring that you have properly prepared yourself for the exam by outlining which sections of the book cover which parts of the Oracle exam. Use this as a guideline to ensure that you have covered all the bases. It can also be used to help you determine which parts of the book to review in case you may have to take the exam again.

A P P E N D I X



<i>Exam Objective</i>	<i>Section Covering Objective</i>
Oracle Architecture Components	Chapter 1: Oracle8i Architecture
Describe the Oracle server architecture and its main components	Chapter 1: Oracle8i Architecture
List the structures involved in connecting a user to an Oracle instance	Chapter 1: Oracle8i Architecture, Processing SQL Statements
List the stages in processing queries, DML statements, COMMITS	Chapter 1: Oracle8i Architecture, Processing SQL Statements
Getting Started with the Oracle Server	Chapter 2: Getting Started with Oracle8i Server
Identify the features of the Universal Installer	Chapter 2: Getting Started with Oracle8i Server, Oracle Universal Installer
Set up operating and password file authentication	Chapter 2: Getting Started with Oracle8i Server, Authenticating Privileged Users
List the main components of Oracle Enterprise Manager and their uses	Chapter 2: Getting Started with Oracle8i Server, Oracle Tools for Administration
Managing an Oracle Instance	Chapter 3: Managing an Oracle Instance
Create the parameter file	Chapter 3: Managing an Oracle Instance, Parameter File
Start up an instance and open the database	Chapter 3: Managing an Oracle Instance, Starting Up and Shutting Down an Oracle Instance
Close the database and shut down the instance	Chapter 3: Managing an Oracle Instance, Starting Up and Shutting Down an Oracle Instance
Get and set parameter values	Chapter 3: Managing an Oracle Instance, Managing Oracle Instance Settings
Manage sessions	Chapter 3: Managing an Oracle Instance, Managing Oracle Instance Settings
Monitor the ALERT file and the trace files	Chapter 3: Managing an Oracle Instance, Managing the ALTER File and Trace Files
Creating a Database	Chapter 4: Creating a Database
Prepare the operating systems	Chapter 4: Creating a Database, Creation Prerequisites
Prepare the parameter file	Chapter 4: Creating a Database, Creation Prerequisites
Create the database	Chapter 4: Creating a Database

Exam Objective	Section Covering Objective
Creating Data Dictionary Views and Standard Packages	Chapter 5: Creating Data Dictionary Views and Standard Packages
Construct the data dictionary views	Chapter 5: Creating Data Dictionary Views and Standard Packages, Creating Data Dictionary Views
Query the data dictionary	Chapter 5: Creating Data Dictionary Views and Standard Packages, Data Dictionary View Types Querying of data dictionary views is also covered in each chapter of the book for the views appropriate to that chapter
Prepare the PL/SQL environment using the administrative scripts	Chapter 5: Creating Data Dictionary Views and Standard Packages, Creating Data Dictionary Views
Administer stored procedures and packages	Chapter 5: Creating Data Dictionary Views and Standard Packages, Stored Program Units
List the types of database event triggers	Chapter 5: Creating Data Dictionary Views and Standard Packages, Stored Program Units
Maintaining the Control File	Chapter 6: Maintaining the Control File
Explain the uses of the control file	Chapter 6: Maintaining the Control File, Overview of the Control File
List the contents of the control file	Chapter 6: Maintaining the Control File, Control File Contents
Multiplex the control file	Chapter 6: Maintaining the Control File, Multiplexing Control Files
Obtain control file information	Chapter 6: Maintaining the Control File, Displaying Information About the Control File
Maintaining Redo Log Files	Chapter 7: Maintaining Redo Log Files
Explain the uses of online redo log files	Chapter 7: Maintaining Redo Log Files
Obtain log and archive information	Chapter 7: Maintaining Redo Log Files,
Control log switches and checkpoints	Chapter 7: Maintaining Redo Log Files, Oracle Use of Redo Log Files
Multiplex and maintain online redo log files	Chapter 7: Maintaining Redo Log Files, Planning Redo Log Files
Plan online redo log files	Chapter 7: Maintaining Redo Log Files, Planning Redo Log Files

Continued

Exam Objective	Section Covering Objective
Troubleshoot common redo log file problems	Chapter 7: Maintaining Redo Log Files, Maintaining Redo Log Files, Troubleshooting LGWR Problems
Analyze online and archived redo logs	Chapter 7: Maintaining Redo Log Files, Using Log Miner
Managing Tablespaces and Datafiles	Chapter 8: Managing Tablespaces and Datafiles
Distinguish the different types of temporary segments	Chapter 8: Managing Tablespaces and Datafiles, Tablespaces, Tablespace Contents Chapter 9: Storage Structure and Relationships, Types of Segments, Temporary Segments
Create tablespaces	Chapter 8: Managing Tablespaces and Datafiles, Tablespaces, Creating Tablespaces
Change the size of tablespaces	Chapter 8: Managing Tablespaces and Datafiles, Managing Tablespaces
Allocate space for temporary segments	Chapter 8: Managing Tablespaces and Datafiles, Tablespaces, Tablespace Contents
Change the status of tablespaces	Chapter 8: Managing Tablespaces and Datafiles, Tablespaces, Maintaining Tablespaces
Changes the storage settings of tablespaces	Chapter 8: Managing Tablespaces and Datafiles, Tablespaces, Maintaining Tablespaces
Relocate tablespaces	Chapter 8: Managing Tablespaces and Datafiles, Tablespaces, Maintaining Tablespaces
Storage Structure and Relationships	Chapter 9: Storage Structure and Relationships
Describe the logical structure of the database	Chapter 9: Storage Structure and Relationships, Overview of Logical Storage Structure Components
List the segment types and their uses	Chapter 9: Storage Structure and Relationships, Types of Segments
List the keywords that control block space usage	Chapter 9: Storage Structure and Relationships, Block Space Utilization

Exam Objective	Section Covering Objective
Obtain information about storage structures from the data dictionary	Chapter 9: Storage Structure and Relationships, Getting Information About Storage Structures
List the criteria for separating segments	Chapter 9: Storage Structure and Relationships, Planning the Location of Segments
Managing Rollback Segments	Chapter 10: Managing Rollback Segments
Create rollback segments using appropriate storage settings	Chapter 10: Managing Rollback Segments, Creating Rollback Segments
Maintain rollback segments	Chapter 10: Managing Rollback Segments, Maintaining Rollback Segments
Plan the number and use of rollback segments	Chapter 10: Managing Rollback Segments, Planning Rollback Segments
Obtain rollback segment information from the data dictionary	Chapter 10: Managing Rollback Segments, Getting Information About Rollback Segments
Troubleshoot common rollback segment problems	Chapter 10: Managing Rollback Segments, Troubleshooting Rollback Segments
Managing Tables	Chapter 11: Managing Tables
Create tables using appropriate storage parameters	Chapter 11: Managing Tables, Creating Tables
Control the space used by tables	Chapter 11: Managing Tables, Creating Tables Chapter 11: Managing Tables, Modifying Tables
Analyze tables to check integrity and migration	Chapter 11: Managing Tables, Modifying Tables Chapter 11: Managing Tables, Getting Information About Tables
Retrieve information about tables from the data dictionary	Chapter 11: Managing Tables, Getting Information About Tables
Convert between different formats of ROWID	Chapter 11: Managing Tables, DBMS_ROWID Package
Managing Indexes	Chapter 12: Managing Indexes
List the different types of indexes and their uses	Chapter 12: Managing Indexes, Types of Indexes in Oracle
Create B-tree and bitmap indexes	Chapter 12: Managing Indexes, Creating Indexes

Continued

Exam Objective	Section Covering Objective
Reorganize indexes	Chapter 12: Managing Indexes, Modifying Indexes
Drop Indexes	Chapter 12: Managing Indexes, Dropping Indexes
Get index information from the data dictionary	Chapter 12: Managing Indexes, Getting Information on Indexes
Maintaining Data Integrity	Chapter 13: Maintaining Data Integrity
Implement data integrity constraints	Chapter 13: Maintaining Data Integrity, Implementing Constraints
Maintain integrity constraints	Chapter 13: Maintaining Data Integrity, Modifying Constraints
Obtain constraint information from the data dictionary	Chapter 13: Maintaining Data Integrity, Getting Information on Constraints
Loading Data	Chapter 14: Loading Data
Load data using direct-load insert	Chapter 14: Loading Data, Loading Data Using Direct Load Insert
Load data into Oracle tables using SQL*Loader: Conventional Path	Chapter 14: Loading Data, Loading Data Using SQL*Loader
Load data into Oracle tables using SQL*Loader: Direct Path	Chapter 14: Loading Data, Loading Data Using SQL*Loader
Reorganizing Data	Chapter 15: Reorganizing Data
Reorganize data using the Export and Import utilities	Chapter 15: Reorganizing Data, Moving Data Using Export and Import Utilities
Move data using transportable tablespaces	Chapter 15: Reorganizing Data, Transportable Tablespaces
Managing Password Security and Resources	Chapter 16: Managing Password Security and Resources
Manage passwords using profiles	Chapter 16: Managing Password Security and Resources, Profiles, Password Management
Administer profiles	Chapter 16: Managing Password Security and Resources, Profiles
Control the use of resources using profiles	Chapter 16: Managing Password Security and Resources, Profiles, Resource Management
Obtain information about profiles, password management, and resources	Chapter 16: Managing Password Security and Resources, Profiles, Getting Information on Profiles

Exam Objective	Section Covering Objective
Manage Users	Chapter 17: Managing Users
Create new database users	Chapter 17: Managing Users, Creating Users
Alter and drop existing database users	Chapter 17: Managing Users, Modifying Users Chapter 17: Managing Users, Dropping Users
Monitor information about existing users	Chapter 17: Managing Users, Getting Information on Users
Managing Privileges	Chapter 18: Managing Privileges
Identify system and object privileges	Chapter 18: Managing Privileges, Types of Privileges
Grant and revoke privileges	Chapter 18: Managing Privileges, Types of Privileges
Control operating system or password file authentication	Chapter 3: Getting Started with Oracle8i Server, Privileged Users Chapter 3: Getting Started with Oracle8i Server, Authenticating Privileged Users
Identify auditing capabilities	Chapter 18: Managing Privileges, Auditing Privilege Usage
Managing Roles	Chapter 19: Managing Roles
Create and modify roles	Chapter 19: Managing Roles, Implementing and Using Roles
Control availability of roles	Chapter 19: Managing Roles, Implementing and Using Roles, Managing Roles
Remove roles	Chapter 19: Managing Roles, Implementing and Using Roles
Use predefined roles	Chapter 19: Managing Roles, Implementing and Using Roles, Pre-Defined Roles in Oracle
Display role information from the data dictionary	Chapter 19: Managing Roles, Implementing and Using Roles, Getting Information on Roles
Using National Language Support	Chapter 20: Using National Language Support
Choose a character set and national character set for the database	Chapter 20: Using National Language Support, Choosing a Database and National Language Character Set for Your Database

Continued

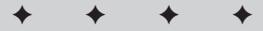
<i>Exam Objective</i>	<i>Section Covering Objective</i>
Specify the language-dependent behavior using initialization parameters, environment variables, and the ALTER SESSION command	Chapter 20: Using National Language Support, Specifying Language Dependent Behavior in Oracle
Use the different types of National Language Support (NLS) parameters	Chapter 20: Using National Language Support, Effects of NLS Parameters
Explain the influence on language-dependent application behavior	Chapter 20: Using National Language Support, Considerations for Using NLS Features
Obtain information about NLS usage	Chapter 20: Using National Language Support, Getting Information About NLS Settings



Exam Tips

Before sitting down to write the “Oracle8i: Architecture and Administration” exam, it is important to know what to expect and how to approach the exam with the highest probability of success.

In this appendix I provide some pointers to help you be better prepared to write the exam, outline the process of registering to take the exam, and explain what happens if you do not succeed on your first attempt.



Preparing to Write the Exam

While some of the points that are outlined in this section may seem common sense, they are here to remind you of what others have used to successfully prepare to pass the exam. In preparing yourself for the exam, follow these steps:

- ◆ **Read the material in this book.** This book was designed to act as a study guide to help prepare you to take and pass the exam. All of the information that you will be tested on can be found in this book. While there are no guarantees, being completely familiar with the contents of this book will go a long way to help you pass the exam.
- ◆ **Get a copy of Oracle.** Appendix A outlines where you can acquire the Oracle software if you do not currently own it. Passing the exam without working with the product is possible, but extremely difficult and quite unlikely. That is not to say it can't be done, but what will happen when you are asked to do the work on a real live database?
- ◆ **Do the labs.** The best way to prepare to perform any task well is to do it over and over again. Performing a task many times reinforces the concepts associated with the task. The labs in this book are designed to reinforce the lessons presented in each chapter.

- ◆ **Hit a few roadblocks.** When doing the labs in this book, or when working with Oracle outside of going through this material, if you hit a roadblock and do not know how to proceed, try to figure it out on your own before turning to the lab answers, the book text, or the Oracle manuals. The way most DBAs learned how to overcome a problem was by running into it at least once, and sometimes a few more times.
- ◆ **Test Yourself.** Each chapter starts with questions to test your knowledge. Each chapter has exam-style assessment questions. Each chapter also has scenarios in which you need to solve a problem. Work through each of these and test your knowledge continuously, not just before you write the exam. If you find you are weak in a certain area, review the material and test again.
- ◆ **Take the practice exams.** This book has a practice exam in Appendix B, as well as one on the CD-ROM. Self Test software also publishes practice exams for Oracle certification exams. Exposure to more questions will better expand your test taking abilities.
- ◆ **Understand the material.** The goal is not to pass the exam but rather to understand the material. Passing the exam is easy if you are comfortable with the subject matter of the exam before you walk in to write it.
- ◆ **Work with it.** Practice makes perfect so work with Oracle every chance you get. Ask your friends and colleagues to give you assignments that test your DBA knowledge of Oracle and allow you to solve problems related to the material in this book. The more you do it, the easier it gets. This works for improving your golf swing, or understanding Oracle.

Registering for the Exam

Oracle certification exams are offered through Sylvan Prometric testing centers worldwide. Consult the *Oracle Certified Professional Program Candidate Guide* available on Oracle's Web site at <http://www.oracle.com/education/certification/index.html?ocpguides.html> or the **Oracle Certification Web site** at <http://www.oracle.com/education/certification/index.html?content.html>.

Taking the Exam

When your exam day arrives, get to the testing center a few minutes early. This will give you some time to relax before going into the test room. You will need to bring two pieces of identification with you (one with a photo and signature, and the other with your signature). You will be asked to sign in and then be escorted to the computer you will use to write the test. You will also get a brief orientation to the testing process if you are new to it.

When taking the test, keep these points in mind:

- ♦ **Pace yourself.** The exam is timed to allow you about a minute or so to answer each question. Take the time to read each question completely and then select your answers. Do not rush the exam. Finish the exam in record time and failing is not as good as taking all the time allowed and passing.
- ♦ **Read the whole question and the answers.** This may seem like a common sense thing to do, but it is amazing how many people do not follow this simple rule. Before jumping on an answer to a question, make sure you have read it completely and have read each of the possible answers. Be attentive to the wording used by the question. For instance, there is a difference between “it must be done” and “it may be done” and “it should be done.” One is mandatory, one is possible, and one is preferable.
- ♦ **Mark questions.** The exam allows you to mark questions whose answers you are unsure of and come back to them later. Use this feature. Even if you are 90 percent sure of an answer, mark the question and come back to it. Who knows? The answer to the question you marked may be found later in the test or be triggered by something later in the test.
- ♦ **Don’t get bogged down.** Do not get bogged down trying to answer a single question. There are many more that you still need to answer so mark the question and move on.
- ♦ **Relax.** Stress and panic will work against you. Take a deep breath and relax.

After the Test

After you have written the test, you will know your results right away. When you confirm that you want to end the test, your score and pass or fail grade will be presented on the screen and a hardcopy printed for you to take home. If you passed, CONGRATULATIONS!

If you were not successful, don’t despair. While this book is designed to help you pass the exam, this is not a guarantee. Use the information on your test score report to determine in which areas you are weak and focus on those before retaking the exam. Don’t neglect other areas though, as the exam will still test your knowledge of the complete set of objectives outlined in Appendix C.

Oracle has a mandatory waiting period of 30 days before you can retake the exam. Use this time to make yourself even more comfortable with the material.

Good luck!



Database Schema for Labs

Throughout this book you have been working with a number of Oracle database objects. The labs and the examples within the various chapters of the book call for a number of tables, and other objects. This appendix provides you with information about the structure of the objects being used, as well as a hard copy of the scripts found in the DBSETUP folder on the CD-ROM that you can use to create this same structure in your database.

A P P E N D I X

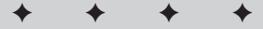


Table Structures

The tables used in the book are as follows:

Courses table

Column Name	Datatype & Length	Null?	Primary Key?	Foreign Key? Table/Column
CourseNumber	int	No	Yes	
CourseName	varchar2 (200)	No	No	
ReplacesCourse	int	Yes	No	
RetailPrice	number (9,2)	No	No	
Description	varchar2 (2000)	Yes	No	

Instructors table

Column Name	Datatype & Length	Null?	Primary Key?	Foreign Key? Table/Column
InstructorID	Int	No	Yes	
Salutation	Char (4)	Yes	No	
LastName	Varchar2 (30)	No	No	
FirstName	Varchar2 (30)	No	No	
MiddleInitial	Varchar2 (5)	Yes	No	
Address1	Varchar2 (50)	Yes	No	
Address2	Varchar2 (50)	Yes	No	
City	Varchar2 (30)	Yes	No	
State	Char (2)	Yes	No	
Country	Varchar2 (30)	Yes	No	
PostalCode	Char (10)	Yes	No	
OfficePhone	Char (15)	Yes	No	

Column Name	Datatype & Length	Null?	Primary Key?	Foreign Key? Table/Column
HomePhone	Char (15)	Yes	No	
CellPhone	Char (15)	Yes	No	
Email	Varchar2 (50)	Yes	No	
InstructorType	Char (10)	No	No	
PerDiemCost	Number (9,2)	Yes	No	
PerDiemExpenses	Number (9,2)	Yes	No	
Comments	Varchar2 (2000)	Yes	No	

Locations table

Column Name	Datatype & Length	Null?	Primary Key?	Foreign Key? Table/Column
LocationID	int	No	Yes	
LocationName	varchar2 (50)	No	No	
Address1	varchar2 (50)	Yes	No	
Address2	varchar2 (50)	Yes	No	
City	varchar2 (30)	Yes	No	
State	char (2)	Yes	No	
Country	varchar2 (30)	Yes	No	
PostalCode	char (10)	Yes	No	
Telephone	char (15)	Yes	No	
Fax	char (15)	Yes	No	
Contact	varchar2 (50)	Yes	No	
Description	varchar2 (2000)	Yes	No	

Students table

<i>Column Name</i>	<i>Datatype & Length</i>	<i>Null?</i>	<i>Primary Key?</i>	<i>Foreign Key? Table/Column</i>
StudentNumber	int	No	Yes	
Salutation	char (4)	Yes	No	
LastName	varchar2 (30)	No	No	
FirstName	varchar2 (30)	No	No	
MiddleInitial	varchar2 (5)	Yes	No	
Address1	varchar2 (50)	Yes	No	
Address2	varchar2 (50)	Yes	No	
City	varchar2 (30)	Yes	No	
State	char (2)	Yes	No	
Country	varchar2 (30)	Yes	No	
PostalCode	char (10)	Yes	No	
HomePhone	char (15)	Yes	No	
WorkPhome	char (15)	Yes	No	
Email	varchar2 (50)	Yes	No	
Comments	varchar2 (2000)	Yes	No	

Scheduled classes table

<i>Column Name</i>	<i>Datatype & Length</i>	<i>Null?</i>	<i>Primary Key?</i>	<i>Foreign Key? Table/Column</i>
ClassID	int	No	Yes	
CourseNumber	int	No	No	Courses (CourseNumber)
LocationID	int	No	No	Locations (LocationID)
ClassRoomNumber	smallint	No	No	
InstructorID	int	No	No	Instructors (InstructorID)
StartDate	date	No	No	

Column Name	Datatype & Length	Null?	Primary Key?	Foreign Key? Table/Column
DaysDuration	smallint	No	No	
Status	char (10)	No	No	
Comments	varchar2 (2000)	Yes	No	

Class enrollment table

Column Name	Datatype & Length	Null?	Primary Key?	Foreign Key? Table/Column
ClassID	int	No	Yes	ScheduledClasses (ClassID)
StudentNumber	int	No	Yes	Students (StudentNumber)
Status	char (10)	No	No	
EnrollmentDate	date	No	No	
Price	number (9,2)	No	No	
Grade	char (40)	Yes	No	
Comments	varchar2 (2000)	Yes	No	

Batch jobs table

Column Name	Datatype & Length	Null?	Primary Key?	Foreign Key? Table/Column
JobId	number (6)	No	Yes	
JobName	varchar2 (30)	No	No	
Status	varchar2 (30)	No	No	
LastUpdated	date	No	No	

Course audit table

<i>Column Name</i>	<i>Datatype & Length</i>	<i>Null?</i>	<i>Primary Key?</i>	<i>Foreign Key? Table/Column</i>
CourseNumber	int	No	Yes	Courses (CourseNumber)
Change	varchar2 (30)	No	Yes	
DateChanged	date	No	Yes	
Price	number (9,2)	Yes	No	
ChangedBy	varchar2 (15)	Yes	No	

Scripts used to create database objects

The CD-ROM accompanying this book has a number of scripts in the DBSETUP folder that can be used to create the tables and load sample data into them. The scripts, and their purpose, are:

- ◆ CREATEUSER.SQL — Creates the STUDENT user and the CERTDB tablespace.
- ◆ CERTDBOBJ.SQL — Creates the tables and adds constraints.
- ◆ INSERT_DATA.SQL — Adds sample data to the database.

Before running the scripts

In order to successfully run the scripts, please ensure that the following have been performed:

1. You have installed, according to the installation instructions provided by Oracle, Oracle8i version 8.1.6 or later, on the computer that you will use to perform the lab exercises and other steps in the book.



Caution

The book assumes you are running Oracle 8.1.6 or later Enterprise Edition, since this is what the Oracle exam is based on. Although most labs will work with Oracle 8.1.6 or later Server, it is recommended that you download the latest version of Oracle8i Enterprise Edition for your platform by joining the Oracle Technology Network at <http://otn.oracle.com> and going to the download section. Oracle8i Personal Edition is not recommended or supported.

2. You have created a database to hold the objects and data that the scripts will create, and configured Net8 to be able to connect to the server.

3. You have been granted the DBA role in the database, or you know the password for the SYSTEM user in the database.
4. You have created a directory on your C: drive (for Windows NT/2000/9x) or off your root (for Linux) to hold the Oracle datafile that will be created by the scripts. (See the scripts for the actual folder name.)
5. You are not running this on a production database used for other critical purposes in your organization.

The scripts themselves are simply ASCII files that may be modified. To ensure that they will work properly in your environment, you should verify their contents and make any necessary changes. It is recommended that you copy the script files from the DBSETUP folder of the CD-ROM to a new folder you create called CERTDB on your hard drive (this is the folder where the datafile will be created by default). Before running the scripts, check their contents for the following:

1. The CREATEUSER.SQL script will connect to the default instance as the user SYSTEM with a password of “manager.” If this is not correct, or you wish to connect as another user who has been assigned the DBA role, or you wish to connect to a different instance, change the appropriate line in the script.
2. The CREATEUSER.SQL script will first delete and then create a tablespace called CERTDB and place the datafile in a folder called CERTDB on your hard disk. If you want to place the datafile in a different location, change the appropriate lines in the script — the line that deletes the data file, as well as the one that creates the tablespace.
3. The CREATEUSER.SQL script will create an Oracle user called STUDENT with a password of “oracle.” If you already have a user called STUDENT created in the database, that user will be dropped by the script before being re-created. Modify the appropriate lines of the script to drop and create a user with a different name.
4. The CERTDBOBJ.SQL script will connect to the database as the user STUDENT by default. If you changed the username in the CREATEUSER.SQL script, modify the appropriate line of CERTDBOBJ.SQL to have the objects created by the user you specified.

Running the scripts

To run the scripts, perform the following tasks:

1. Logon to the computer that you will be running Oracle from as an administrator (preferred).
2. If you are running WindowsNT/2000/9x, create a folder off of the root of your C: drive called **CERTDB**. If you are running Linux or another UNIX variant, create a folder off the root called **CERTDB**. In the Linux/UNIX world, the folder must be created as all uppercase.

3. Insert the CD into your CD-ROM drive and locate the **DBSETUP** folder. Copy the contents of the **DBSETUP** folder to the **CERTDB** folder you created.
4. Invoke Server Manager line mode from the command line and execute the scripts in the following order:
 - CREATEUSER.SQL — To create the user and tablespace
 - CERTDBOBJ.SQL — To create the database objects
 - INSERT_DATA.SQL — To load the sample data

To execute the scripts, at the command prompt, issue the following commands:

```
C:\CERTDB> svrmgr1

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SVRMGRL> @createuser.sql;
...
SVRMGRL> @certdbobj.sql;
...
SVRMGRL> @insert_data.sql;
...
SVRMGRL> quit
Server Manager Complete

C:\CERTDB>
```

Text of scripts used to create the database objects

The scripts mentioned previously can be found on the CD-ROM. If you modify them or do not have the CD handy, you can type them in from the code listings that follow:

CREATEUSER.SQL

```
# This script creates the STUDENT user and CERTDB tablespace.
#
# The script first connects as system/manager and then drops the objects,
# after which it creates them. If you have errors on dropping the objects,
# these are normal the first time you run the script because the objects
# do not yet exist.
#
# REVISION HISTORY:
```

```

#
# 25-jan-2001      Initial Creation      Damir Bersinic
#
#
# Connect to the default instance
connect system/manager;
#
# Drop the user STUDENT, if it exists
DROP USER Student CASCADE;
#
# Drop the CERTDB tablespace, if it exists
DROP TABLESPACE CERTDB INCLUDING CONTENTS;
#
# Delete the file from the hard drive (for Windows).
# Comment out for running on Unix.
# Change the path if required.
HOST "DEL C:\CERTDB\CERTDB01.DBF";
#
# Delete the file from the hard drive (for Linux/Unix).
# Uncomment the HOST command to run on Unix.
# Change the path if required.
# HOST "rm /CERTDB/CERTDB01.DBF";
#
# Create the tablespace (for Windows)
# Comment out for running on Unix.
# Change the path if required.
CREATE TABLESPACE CERTDB DATAFILE 'C:\CERTDB\CERTDB01.DBF' SIZE 10M;
#
# Create the tablespace (for Linux/Unix)
# Uncomment the CREATE TABLESPACE command to run on Unix.
# Change the path if required.
# CREATE TABLESPACE CERTDB DATAFILE '/CERTDB/CERTDB01.DBF' SIZE 10M;
#
# Create the USER and grant a quota on the CERTDB tablespace.
# Make sure a TEMP tablespace already exists in your database.
CREATE USER Student IDENTIFIED BY oracle
        DEFAULT TABLESPACE CERTDB
        TEMPORARY TABLESPACE TEMP
        QUOTA UNLIMITED ON CERTDB;
#
# Allow the STUDENT user to connect and logon to the instance.
GRANT CONNECT,RESOURCE TO Student;

```

CERTDBOBJ.SQL

```

# This script creates the tables in the STUDENT user schema.
#
# The script first connects as student/oracle and then drops the objects,
# after which it creates them. If you have errors on dropping the objects,
# these are normal the first time you run the script because the objects
# do not yet exist.
#

```

```

# REVISION HISTORY:
#
# 25-jan-2001      Initial Creation      Damir Bersinic
#
#
# Connect to the default instance as user STUDENT.
connect student/oracle;
#
#
# Drop all tables and cascade constraints
DROP TABLE Courses CASCADE CONSTRAINTS;
DROP TABLE Instructors CASCADE CONSTRAINTS;
DROP TABLE Locations CASCADE CONSTRAINTS;
DROP TABLE Students CASCADE CONSTRAINTS;
DROP TABLE ScheduledClasses CASCADE CONSTRAINTS;
DROP TABLE ClassEnrollment CASCADE CONSTRAINTS;
DROP TABLE BatchJobs CASCADE CONSTRAINTS;
DROP TABLE CourseAudit CASCADE CONSTRAINTS;
#
#
# Create each table in the appropriate order on the CERTDB
# tablespace. All tables will be owned by STUDENT.
#
CREATE TABLE Courses (
    CourseNumber int NOT NULL
        CONSTRAINT PK_CourseNumber PRIMARY KEY ,
    CourseName varchar2 (200) NOT NULL ,
    ReplacesCourse int NULL ,
    RetailPrice number (9,2) NOT NULL ,
    Description varchar2 (2000) NULL
) TABLESPACE CERTDB;

CREATE TABLE Instructors (
    InstructorID int NOT NULL
        CONSTRAINT PK_InstructorID PRIMARY KEY,
    Salutation char (4) NULL ,
    LastName varchar2 (30) NOT NULL ,
    FirstName varchar2 (30) NOT NULL ,
    MiddleInitial varchar2 (5) NULL ,
    Address1 varchar2 (50) NULL ,
    Address2 varchar2 (50) NULL ,
    City varchar2 (30) NULL ,
    State char (2) NULL ,
    Country varchar2 (30) NULL ,
    PostalCode char (10) NULL ,
    OfficePhone char (15) NULL ,
    HomePhone char (15) NULL ,
    CellPhone char (15) NULL ,
    EMail varchar2 (50) NULL ,

```

```

    InstructorType char(10) NOT NULL,
    PerDiemCost number (9,2) NULL ,
    PerDiemExpenses number (9,2) NULL ,
    Comments varchar2 (2000) NULL
) TABLESPACE CERTDB;

CREATE TABLE Locations (
    LocationID int NOT NULL
        CONSTRAINT PK_LocationID PRIMARY KEY ,
    LocationName varchar2 (50) NOT NULL ,
    Address1 varchar2 (50) NULL ,
    Address2 varchar2 (50) NULL ,
    City varchar2 (30) NULL ,
    State char (2) NULL ,
    Country varchar2 (30) NULL ,
    PostalCode char (10) NULL ,
    Telephone char (15) NULL ,
    Fax char (15) NULL ,
    Contact varchar2 (50) NULL ,
    Description varchar2 (2000) NULL
) TABLESPACE CERTDB;

CREATE TABLE Students (
    StudentNumber int NOT NULL
        CONSTRAINT PK_StudentNumber PRIMARY KEY,
    Salutation char (4) NULL ,
    LastName varchar2 (30) NOT NULL ,
    FirstName varchar2 (30) NOT NULL ,
    MiddleInitial varchar2 (5) NULL ,
    Address1 varchar2 (50) NULL ,
    Address2 varchar2 (50) NULL ,
    City varchar2 (30) NULL ,
    State char (2) NULL ,
    Country varchar2 (30) NULL ,
    PostalCode char (10) NULL ,
    HomePhone char (15) NULL ,
    WorkPhone char (15) NULL ,
    EMail varchar2 (50) NULL ,
    Comments varchar2 (2000) NULL
) TABLESPACE CERTDB;

CREATE TABLE ScheduledClasses (
    ClassID int NOT NULL
        CONSTRAINT PK_ClassID PRIMARY KEY,
    CourseNumber int NOT NULL ,
    LocationID int NOT NULL ,
    ClassroomNumber smallint NOT NULL ,
    InstructorID int NOT NULL ,
    StartDate date NOT NULL ,
    DaysDuration smallint NOT NULL ,
    Status char (10) NOT NULL ,

```

```

        Comments varchar2 (2000) NULL
    ) TABLESPACE CERTDB;

CREATE TABLE ClassEnrollment (
    ClassID int NOT NULL ,
    StudentNumber int NOT NULL ,
    Status char (10) NOT NULL ,
    EnrollmentDate date NOT NULL,
    Price number (9,2) NOT NULL ,
    Grade char (4) NULL ,
    Comments varchar2 (2000) NULL
) TABLESPACE CERTDB;

CREATE TABLE BatchJobs (
    JobId number (6) NOT NULL,
    JobName  varchar2 (30) NOT NULL,
    Status varchar2 (30) NOT NULL,
    LastUpdated date NOT NULL
) TABLESPACE CERTDB;

CREATE TABLE CourseAudit (
    CourseNumber int NOT NULL,
    Change varchar2 (30) NOT NULL,
    DateChanged date NOT NULL,
    Price number (9,2),
    ChangedBy varchar2 (15)
) TABLESPACE CERTDB;
#
#
#
# Alter the tables to include foreign keys and composite primary keys

ALTER TABLE ClassEnrollment ADD
    (CONSTRAINT PK_ClassID_StudentNumber
    PRIMARY KEY (ClassID, StudentNumber)) ;

ALTER TABLE ScheduledClasses ADD
    (CONSTRAINT FK_SchedClass_CourseNum
    FOREIGN KEY (CourseNumber) REFERENCES Courses (CourseNumber));

ALTER TABLE ScheduledClasses ADD
    (CONSTRAINT FK_SchedClasses_LocationID
    FOREIGN KEY (LocationID) REFERENCES Locations (LocationID));

ALTER TABLE ScheduledClasses ADD
    (CONSTRAINT FK_SchedClasses_InstID
    FOREIGN KEY (InstructorID)
    REFERENCES Instructors (InstructorID));

ALTER TABLE ClassEnrollment ADD
    (CONSTRAINT FK_ClassEnrollment_ClassID
    FOREIGN KEY (ClassID) REFERENCES ScheduledClasses (ClassID));

```

```
ALTER TABLE ClassEnrollment ADD
  (CONSTRAINT FK_ClassEnrollment_StudentNum
   FOREIGN KEY (StudentNumber) REFERENCES Students (StudentNumber));
```

```
ALTER TABLE BatchJobs ADD
  (CONSTRAINT BatchJobs_JobId_pk PRIMARY KEY(JobId));
```

```
ALTER TABLE CourseAudit ADD
  (CONSTRAINT CourseAudit_PK
   PRIMARY KEY(CourseNumber, Change, DateChanged));
```

```
ALTER TABLE CourseAudit ADD
  (CONSTRAINT FK_CourseAudit_CourseNumber
   FOREIGN KEY (CourseNumber) REFERENCES Courses(CourseNumber));
```

INSERT_DATA.SQL

```
# This script populates the tables in the STUDENT user schema.
#
# This script assumes you are connected as student/oracle.
#
# The script first deletes data from each of the tables and then performs
# inserts
# to load the sample data.
#
#
# REVISION HISTORY:
#
# 25-jan-2001      Initial Creation      Damir Bersinic
#
#
#
# Disable foreign key constraints to ensure TRUNCATE works.
#
ALTER TABLE ScheduledClasses DISABLE CONSTRAINT FK_SchedClass_CourseNum;

ALTER TABLE ScheduledClasses DISABLE CONSTRAINT FK_SchedClasses_LocationID;

ALTER TABLE ScheduledClasses DISABLE CONSTRAINT FK_SchedClasses_InstID;

ALTER TABLE ClassEnrollment DISABLE CONSTRAINT FK_ClassEnrollment_ClassID;

ALTER TABLE ClassEnrollment DISABLE CONSTRAINT FK_ClassEnrollment_StudentNum;

ALTER TABLE CourseAudit DISABLE CONSTRAINT FK_CourseAudit_CourseNumber;
#
#
# Truncate all tables in the appropriate order.
# This is required to ensure primary keys are not violated.
```

```

#
TRUNCATE TABLE BatchJobs;
TRUNCATE TABLE ClassEnrollment;
TRUNCATE TABLE ScheduledClasses;
TRUNCATE TABLE CourseAudit;
TRUNCATE TABLE Locations;
TRUNCATE TABLE Courses;
TRUNCATE TABLE Instructors;
TRUNCATE TABLE Students;
#
#
# Enable foreign key constraints to ensure proper data load
#
ALTER TABLE ScheduledClasses ENABLE CONSTRAINT FK_SchedClass_CourseNum;

ALTER TABLE ScheduledClasses ENABLE CONSTRAINT FK_SchedClasses_LocationID;

ALTER TABLE ScheduledClasses ENABLE CONSTRAINT FK_SchedClasses_InstID;

ALTER TABLE ClassEnrollment ENABLE CONSTRAINT FK_ClassEnrollment_ClassID;

ALTER TABLE ClassEnrollment ENABLE CONSTRAINT FK_ClassEnrollment_StudentNum;

ALTER TABLE CourseAudit ENABLE CONSTRAINT FK_CourseAudit_CourseNumber;
#
#
# Insert sample data into the Students table.
#
INSERT INTO STUDENTS VALUES
(1000,'Mr','Smith','John','H','34 Anystreet',null, 'Victoria','BC','Canada',
'V5F 3E8','904-567-8889','904-787-8888','james@emailrus.com',null);

INSERT INTO STUDENTS VALUES
(1001,'Mr','Jones','Davey',null,'10 Main St',null,
'New York','NY','USA','87653','312-334-8889',
'312-642-5134','djones@hitech.com',null);

INSERT INTO STUDENTS VALUES
(1002,'Mrs','Massey','Jane','S','723 Church St',null,
'New York','NY','USA','87654','412-324-0880',
'412-887-7489','jmassey@hitech.com',null);

INSERT INTO STUDENTS VALUES
(1003,'Mr','Smith','Trevor','J','13 Crosswood Cres',null,
'Toronto','ON','Canada','M5T 5F6',
'416-456-7890',null,'trevorsmtih@comptel.com',null);

INSERT INTO STUDENTS VALUES
(1004,null,'Hogan','Mike',null,'49 Bentbrook Cres',null,
'Ottawa','ON','Canada','K4M 1Y5',
'613-765-4321','613-567-1234','mhogan@consulters.com',null);

```

```
INSERT INTO STUDENTS VALUES
(1005,'Mr','Hee','John','K','Apt 7','90th Street','New
York','NY','USA','76990','412-567-8673',
'412-747-6543','johnhee@emailrus.com',null);

INSERT INTO STUDENTS VALUES
(1006,'Mrs','Andrew','Susan','M','15 King St',null,'Dallas','TX','USA','87654',
'492-667-8889','492-875-9876','sandrew@bigtime.com',null);

INSERT INTO STUDENTS VALUES
(1007,'Mrs','Holland','Roxanne',null,'212 Lorne St',null,
'San Francisco','CA','USA','77765',
'721-557-8567','721-787-5538','rholland@bigtime.com',null);

INSERT INTO STUDENTS VALUES
(1008,'Mr','Jones','Gordon',null,'17 Nisku',null,'Toronto','ON','Canada',
'T2L 4R8','416-663-5689','416-645-5246','gordonjones@wesell.ca',null);

INSERT INTO STUDENTS VALUES
(1009,'Mrs','Colter','Sue','J','1112 Queen St',null,
'San Francisco','CA','USA','56443','721-566-8645',
'721-744-4756','suecolter@compstore.com',null);

INSERT INTO STUDENTS VALUES
(1010,'Mr','Patterson','Chris','M','72 Regent St',null,
'San Fransisco','CA','USA','57572','721-445-5239',
'721-547-3256','cpatterson@emailrus.com',null);
#
#
# Insert sample data into the Courses table.
#
INSERT INTO COURSES VALUES
(100,'Basic SQL',null,2000,
'An introduction to basic SQL statements and commands');

INSERT INTO COURSES VALUES
(110,'Advanced SQL',null,2000,
'Advanced SQL statements and commands for exerienced users');

INSERT INTO COURSES VALUES
(201,'Performance Tuning your Database',200,4000,
'Concepts and tricks to tune your database for optimum performance');

INSERT INTO COURSES VALUES
(200,'Database Performance Basics',null,4000,
'How to tune your database for maximum performance');

INSERT INTO COURSES VALUES
(210,'Database Administration',null,4500,
'Everything the DBA needs to know to start building a database');
```

```

INSERT INTO COURSES VALUES
(220,'Backing up your database',null,3000,
'The essentials for backing up and recovering the database after a failure');

INSERT INTO COURSES VALUES
(300,'Basic PL/SQL',null,2500,
'An introduction to the PL/SQL programming language');

INSERT INTO COURSES VALUES
(310,'Advanced PL/SQL',null,2000,
'A follow-up to the basic PL/SQL course that introduces complicated PL/SQL
programming techniques');

INSERT INTO COURSES VALUES
(320,'Using your PL/SQLskills',null,1750,
'Introduces database triggers and database packages');
#
#
# Insert sample data into the Locations table.
#
INSERT INTO locations VALUES
(100,'New York Park Ave','80 Park Ave',null,'New York','NY','USA','66578',
'412-389-8889','412-389-8859','Charlene Moore',
'Beautiful location overlooking fabulous central park, easy access to subway');

INSERT INTO locations VALUES
(200,'San Francisco Downtown','40 Bay St',null,
'San Francisco','CA','USA','85763',
'721-765-0987','721-765-9421','James Madison',
'Located in downtown San Francisco, you can see Alcatraz on a clear day');

INSERT INTO locations VALUES
(300,'Downtown Toronto','40 Yonge Street',null,'Toronto','ON','Canada',
'M6H 5K8','416-543-8768','416-544-3965','Joanne Matthews',
'Convenient downtown location, easy access to subway');
#
#
# Insert sample data into the Instructors table.
#
INSERT INTO instructors VALUES
(300,'Mr','Harrison','Michael','H','8899 Eglinton Ave',null,
'Toronto','ON','Canada','M7H 6H5','416-543-8769',
'416-778-5366',null,'michaelharrison@trainers.com','ORACLE',500,200,null);

INSERT INTO instructors VALUES
(310,'Mrs','Keele','Susan','J','42 Bloor St',null,'Toronto','ON','Canada',
'M5T 5F7','416-543-8775','416-857-9876',null,
'susankeele@trainers.com','UNIX',450,200,null);

```

```
INSERT INTO instructors VALUES
(100,'Mr','Ungar','David','J','995 White Plains Ave',null,'New York','NY','USA',
'98750','412-389-6557','412-345-6543',null,'davidungar@trainers.com',
'ORACLE',600,200,null);
```

```
INSERT INTO instructors VALUES
(110,'Mr','Jamieson','Kyle','L','Apt 86','95 Cornerbrook St','New York',
'NY','USA','87653','412-389-7683','412-889-0987','412-987-0423',
'kylejamieson@trainers.com','ORACLE',500,200,null);
```

```
INSERT INTO instructors VALUES
(200,'Miss','Cross','Lisa','M','45 Sunny Drive',null,'Palo Alto','CA','USA',
'89075','721-765-9985','721-649-0944',null,
'lisacross@trainers.com','UNIX',750,250,null);
```

```
#
```

```
#
```

```
# Insert sample data into the ScheduledClasses table.
```

```
#
```

```
INSERT INTO scheduledclasses VALUES
(50,100,100,4,100,'06-jan-2001',4,'Confirmed',null);
```

```
INSERT INTO scheduledclasses VALUES
(51,200,300,1,200,'13-jan-2001',5,'Confirmed',null);
```

```
INSERT INTO scheduledclasses VALUES
(53,100,300,2,110,'14-feb-2001',4,'Hold',null);
```

```
#
```

```
#
```

```
# Insert sample data into the ClassEnrollment table.
```

```
#
```

```
INSERT INTO classenrollment
(ClassId, StudentNumber, Status, EnrollmentDate, Price, Grade, Comments)
VALUES (50, 1001, 'Confirmed','01-JAN-2001',2000, 'B', null);
```

```
INSERT INTO classenrollment
(ClassId, StudentNumber, Status, EnrollmentDate, Price, Grade, Comments)
VALUES (50, 1002, 'Confirmed','12-DEC-2000',1750, 'A', null);
```

```
INSERT INTO classenrollment
(ClassId, StudentNumber, Status, EnrollmentDate, Price, Grade, Comments)
VALUES (50, 1005, 'Confirmed','21-DEC-2000',2000, 'F',
'Missed last two days of class - will resit in March');
```

```
INSERT INTO classenrollment
(ClassId, StudentNumber, Status, EnrollmentDate, Price, Grade, Comments)
VALUES (51, 1003, 'Cancelled','01-JAN-2001',4000, null, null);
```

```
INSERT INTO classenrollment
(ClassId, StudentNumber, Status, EnrollmentDate, Price, Grade, Comments)
VALUES (51, 1004, 'Confirmed','5-JAN-2001',4000, 'A', null);
```

```
INSERT INTO classenrollment
    (ClassId, StudentNumber, Status, EnrollmentDate, Price, Grade, Comments)
VALUES (51, 1008, 'Confirmed','02-DEC-2000',3500, 'A', null);

INSERT INTO classenrollment
    (ClassId, StudentNumber, Status, EnrollmentDate, Price, Grade, Comments)
VALUES (53, 1003, 'Hold','02-JAN-2001',1500, null, null);
#
#
# Insert sample data into the BatchJobs table.
#
INSERT INTO BatchJobs (JobId, JobName, Status, LastUpdated)
VALUES (100, 'CLASS_STATUS','RUNNING','01-MAR-2001');

INSERT INTO BatchJobs (JobId, JobName, Status, LastUpdated)
VALUES (101, 'PRINT_REGISTRATION','COMPLETED','12-MAR-2001');

INSERT INTO BatchJobs (JobId, JobName, Status, LastUpdated)
VALUES (102, 'CALCULATE_REVENUE','COMPLETED','01-MAR-2001');
#
#
# Commit the changes to the database.
COMMIT;
```



Suggested Readings, Web Sites, and Other Resources

This appendix provides information on books, Web sites, and other resources, such as periodicals, that can be used to better prepare yourself for the “Oracle8i: Architecture and Administration” exam. Although you are not expected or required to read all the books and visit every Web site, working with Oracle and being completely familiar with the contents of this book are necessary to prepare you for taking the exam.

Suggested Readings

Additional research and preparation before taking the “Oracle8i: Architecture and Administration” can always be beneficial. The books listed here provide additional information on database administration of Oracle8i, as well as information that may make it easier for you to become a DBA of an Oracle database if you have experience in other relational database management systems.

Books

Stephen Chelack, *Oracle8i and Microsoft SQL Server 2000 Integration* (IDG Books Worldwide, 2000)

Jason S. Couchman, *Oracle8i Certified Professional DBA Practice Exams* (Osborne/McGraw-Hill, 2001)

David Ensor and Tim Stevenson, *Oracle Design* (O'Reilly & Associates, 1997)

Ralph Kimball, et al., *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses* (John Wiley & Sons, 1998)

David C. Kreines and Ken Jacobs, *Oracle SQL: The Essential Reference* (O'Reilly & Associates, 2000)

Kevin Loney and George Koch, *Oracle8i: The Complete Reference* (Osborne/McGraw-Hill, 2000)

Oracle documentation manuals

Oracle8i Administrator's Guide

Oracle8i Administrator's Guide for Windows NT

Oracle8i Concepts

Oracle8i Reference

Oracle8i SQL Reference

Oracle8i Utilities

Oracle8i Data Warehousing Guide

Oracle8i National Language Support Guide

*Oracle8i Documentation Addendum
Release 2 (8.1.6)*

*Oracle8i Documentation Addendum
Release 3 (8.1.7)*

Oracle Enterprise Manager Administrator's Guide

Oracle Enterprise Manager Concepts Guide

Oracle Enterprise Manager Configuration Guide

*SQL*Plus User's Guide and Reference*

Web Sites

The Web sites in the table below are useful in acquiring Oracle software and getting information about Oracle and the Oracle Certified Professional (OCP) program.

URL	Description
http://www.oracle.com	Oracle Corporation home page
http://education.oracle.com	Oracle Education home page. Includes information on the Oracle Certified Professional program, as well as Oracle course offerings.
http://technet.oracle.com	Oracle Technology Network home page. Provides information on Oracle products, as well as copies of Oracle database and other tools. Membership is free.
http://technet.oracle.com/docs/content.html	Oracle documentation on TechNet. This URL provides links to all the Oracle documentation sets for Oracle8i—in case you don't have the hard copies or CD handy.
http://www.certcities.com	CertCities home page. This site offers information on certifications, as well as columns and feature articles on Oracle and the Oracle Certified Professional (OCP) program.
http://www.orafaq.com/faq.htm	Underground Oracle FAQ. This site provides links and information on Oracle products and features.
http://www.oracle.com/oramag	<i>Oracle Magazine</i> Web site. You can order a free subscription to Oracle's official magazine or review technical and other articles online.



Index

Symbols and Numerics

- # (pound sign) comment prefix, 95, 753
- \$ (dollar sign) base table name component, 197
- (dash) SQL*Plus parameter display command, 56
- /*+ (slash, asterisk, plus sign) optimizer hint prefix, 690
- @ (at sign) SQL*Plus parameter prefix, 57
- _ (underscore) parameter prefix, 102

A

- ABORT shutdown mode, 94, 110–111
- ACCOUNT LOCK | UNLOCK CREATE USER parameter, 855
- account lockout
 - passwords, on invalid, 805, 806
 - status, viewing, 826–827
 - user creation, during, 853, 855, 856, 858
- Acrobat Reader (on the CD), 1068
- ADMINISTER DATABASE TRIGGER privilege, 223
- Administration Assistant snap-in, 53
- Adobe Acrobat Reader (on the CD), 1068
- AFTER events, 215
- ALERT file, 100, 122–125, 301
- alert support, setting up, 235
- ALL_ data dictionary views, 198, 200, 204
- ALL_DEF_AUDIT_OPTS view, 946–949
- ALL_TAB_PRIVS view, 930
- ALL_TAB_PRIVS_MADE view, 930
- ALL_TAB_PRIVS_RECD view, 931
- ALTER ANY INDEX privilege, 604
- ALTER DATABASE command
 - ADD LOGFILE, 295
 - ADD LOGFILE MEMBER, 296, 299, 300
 - BACKUP, 260
 - BACKUP CONTROLFILE TO, 125, 260
 - CHARACTER SET, 157, 1031
 - CLEAR LOGFILE, 292, 300
 - CLOSE, 106
 - described, 106
 - DISMOUNT, 106
 - DROP LOGFILE, 297
 - DROP LOGFILE MEMBER, 298, 299, 300
 - MOUNT state, 106
 - NATIONAL CHARACTER SET, 157, 1031
 - NOMOUNT state, 106
 - OPEN, 106
 - OPEN READ ONLY, 108
 - RENAME FILE, 261, 299, 353, 355–356, 357
 - ALTER DATABASE privileges
 - ARCHIVELOG, 47
 - BACKUP CONTROLFILE, 46
 - DISABLE RESTRICTED SESSION, 47
 - ENABLE RESTRICTED SESSION, 47
 - MOUNT, 46
 - OPEN, 46
 - RECOVER DATABASE, 47
- ALTER INDEX command
 - ALLOCATE EXTENT, 605
 - COALESCE, 608
 - DEALLOCATE UNUSED, 605
 - REBUILD, 523, 606–607
 - REBUILD ONLINE, 607–608
 - syntax, 604–605
- ALTER PROFILE command, 818–819
- ALTER PROFILE privilege, 813
- ALTER RESOURCE COST command, 812
- ALTER ROLE command, 991
- ALTER ROLLBACK SEGMENT command, 432, 459, 460
- ALTER ROLLBACK SEGMENT ONLINE command, 454
- ALTER ROLLBACK SEGMENT SHRINK command, 436
- ALTER SESSION command
 - ENABLE PARALLEL DML, 691, 693
 - FORCE PARALLEL DML, 693
 - NLS settings, 1032, 1036–1038
 - SET CONSTRAINT[S], 647
 - SET log_checkpoint_timeout, 117
 - SET QUERY_REWRITE_ENABLED, 691, 693
 - SET SORT_AREA_SIZE, 117, 599
- ALTER SESSION privilege, 117
- ALTER SYSTEM command
 - ARCHIVE LOG ALL, 289
 - ARCHIVE LOG CURRENT, 289
 - CHECKPOINT, 15, 16, 283
 - KILL SESSION, 465
 - SET RESOURCE_LIMIT, 805
 - SWITCH LOG FILE, 284, 298, 299
- ALTER SYSTEM privilege, 117
- ALTER TABLE command
 - ADD CONSTRAINT, 656–657
 - ALLOCATE EXTENT, 402, 533
 - BUFFER POOL, 531
 - CACHE, 531
 - CHECKPOINT, 542
 - CONTINUE option, 543
 - DEALLOCATE UNUSED, 540

Continued

- ALTER TABLE command (*continued*)
 - DISABLE ALL TRIGGERS, 224
 - DISABLE CONSTRAINT, 644
 - DROP COLUMN, 542–543
 - DROP CONSTRAINT, 657
 - DROP UNUSED COLUMNS, 543
 - ENABLE ALL TRIGGERS, 224
 - ENABLE CONSTRAINT, 644, 656
 - EXCEPTIONS, 662
 - INTRANS, 531
 - LOCATIONS, 532
 - LOGGING, 531, 691
 - MAXEXTENTS, 531
 - MAXTRANS, 531
 - MINEXTENTS, 531
 - MODIFY, 657–658, 659, 667
 - MOVE, 524, 534, 606, 737, 743–744
 - NEXT, 531
 - NOCACHE, 531
 - NOLOGGING, 531, 691
 - PCTFREE, 531, 532
 - PCTINCREASE, 531, 532
 - PCTUSED, 531, 532
 - RENAME, 736
 - RENAME TO LOCATIONS, 736
 - SET UNUSED COLUMN, 543
- ALTER TABLESPACE BEGIN BACKUP privilege, 47
- ALTER TABLESPACE command, 346, 347–348, 351
- ALTER TABLESPACE END BACKUP privilege, 47
- ALTER TABLESPACE RENAME DATAFILE
 - command, 353–354
- ALTER TABLESPACE TOOLS READ ONLY
 - command, 777
- ALTER TABLESPACE TOOLS READ WRITE
 - command, 778
- ALTER TRIGGER command, 224
- ALTER USER command, 804, 815, 863–864, 994
- ALTER USER privilege, 813
- ANALYZE Import utility parameter, 756
- ANALYZE INDEX command, 606
- ANALYZE privilege, 897
- ANALYZE TABLE command, 529, 532–533, 537–538, 547
- AQ_ADMINISTRATOR_ROLE role, 981
- AQ_USER_ROLE role, 981
- ARC0. *See* archiver process (ARC0)
- ARCHIVE LOG LIST command, 54, 294
- ARCHIVELOG mode, 21–22, 161, 176, 289–291
- archiver process (ARC0), 6, 17, 99, 294
- archiving. *See also* logs; redo log files
 - ARCHIVELOG mode, 21–22, 161, 176, 289–291
 - archiver process (ARC0), 6, 17, 99, 294
 - destination, 99, 289, 294
 - file information, retrieving, 263, 294
 - log sequence numbers (LSNs), retrieving, 294
 - mode, viewing, 293–294
 - NOARCHIVELOG mode, 21, 176, 256, 287–289
 - redo log files, automatic, 21–22, 161, 277
 - status, retrieving, 263, 294
- ARIAZIM (on the CD), 1068–1069
- ASC/DESC parameter, 597
- AUDIT command
 - ALL, 943
 - BY ACCESS, 943
 - BY SESSION, 943
 - CLUSTER, 938
 - CONTEXT, 938
 - DATABASE LINK, 939
 - described, 894, 938
 - DIMENSION, 939
 - DIRECTORY, 939
 - INDEX, 939
 - NOT EXISTS, 939
 - ON DEFAULT, 943
 - PROCEDURE, 939
 - PROFILE, 939
 - PUBLIC DATABASE LINK, 939
 - PUBLIC SYNONYM, 939
 - ROLE, 940
 - ROLLBACK SEGMENT, 940
 - SEQUENCE, 940
 - SESSION, 940
 - SYNONYM, 940
 - syntax, 941, 952
 - SYSTEM AUDIT, 940
 - SYSTEM GRANT, 940
 - TABLE, 940, 941
 - TABLESPACE, 940
 - TRIGGER, 940
 - TYPE, 940
 - USER, 941
 - VIEW, 941
- auditing
 - access records, by, 935
 - ALTER operations, 944, 945
 - AUDIT operations, 944, 945
 - audit records, adding to trail, 942
 - audit records, generating at user reconnection, 938, 941
 - audit trail, backing up, 935
 - audit trail deletion, roles for, 979
 - audit trail location, 935, 937
 - audit trail monitoring, 935
 - audit trail, moving, 935
 - audit trail security, 936
 - audit trail size, 935, 936

- audit trail, storing at database level, 935, 937
 - audit trail, storing at operating system level, 935–936, 937
 - audit trail, viewing, 949–952
 - clusters, 938
 - COMMENT operations, 944
 - connections, 933, 940
 - connections, viewing audit options present, 950
 - contexts, 938
 - data dictionary involvement, 197, 232
 - database, 933
 - database audit options, viewing, 946–949
 - database, disabling, 945–946
 - database, enabling, 937, 938–941
 - database links, 939
 - default, on, 943
 - DELETE operations, 944, 945
 - detail, determining level of, 934–935
 - dimensions, 939
 - directories, 939
 - disabling, 894, 945–946
 - disk space considerations, 935
 - enabling, 894, 937–941
 - Event Viewer application log, 933
 - EXECUTE operations, 944
 - exporting auditing information, 739, 740, 742
 - failure, 934, 941, 942, 947
 - functions, 939
 - GRANT operations, 944, 945
 - importing auditing information, 739, 740, 742, 743
 - index operations, 939, 944
 - INSERT operations, 944
 - instances, enabling for, 937–938
 - libraries, 939
 - LOCK operations, 944, 945
 - objects, 935, 942–945
 - packages, 939
 - performance overhead, 893
 - PL/SQL blocks, 937–938
 - privilege usage, 933
 - privilege usage, system, 940, 941–942
 - privilege usage, viewing audit options present, 948
 - privileges, operations on, 940
 - procedures, 937–938, 939
 - profiles, 939
 - program units, 944
 - RENAME operations, 944, 945
 - responsibility for, 936
 - roles, 933, 940
 - rollback segments, 940
 - schema objects, viewing audit options present, 948–949, 950
 - SELECT operations, 944
 - sequences, 940, 944
 - session setting changes, 942
 - statement execution, 937–941
 - statement execution, viewing audit options present, 946–948, 950
 - strategy, 934–936
 - success, 934, 947
 - synonyms, 939
 - tables, 940, 941, 943–945
 - tablespaces, 940
 - triggers, 940
 - triggers, using, 933–934
 - types, 933, 940
 - UNIX systems, 933, 937
 - UPDATE operations, 944
 - users, 940, 941
 - value-based, 933–934
 - views, operations on, 940, 943–944
 - views, using to retrieve information about, 946–952
 - WindowsNT/2000 systems, 933, 937
 - AUDIT_TRAIL initialization parameter, 937
 - authentication. *See also* passwords; security
 - database authentication, 849
 - encryption, using, 46
 - groups necessary, 48–49
 - logon credentials, 57
 - method, assigning, 854, 858
 - method, choosing, 849–850
 - network authentication, 850, 870
 - operating system authentication, 45–49, 101, 144, 849–850, 860–863
 - password file, 46, 49–52, 144
 - remote connections, 45–46
 - roles, by application, 976, 984
 - roles, by operating system, 977, 982–983, 984
 - roles, by Oracle Security Server, 983–984
 - users, privileged, 45–46
 - Authentication Mechanism, 801
 - AUTOEXEC.BAT character set environment variable, 1036
- B**
- background processes, 6, 8, 13–18, 100, 162
 - BACKGROUND_DUMP_DEST initialization parameter, 100, 125, 163
 - backups
 - ARCHIVELOG mode, in, 289, 290
 - control files, 257, 260–261, 263
 - datafile backup state, retrieving, 263
 - export operations, using, 741

Continued

- backups (*continued*)
 - load operations, after, 690, 715
 - logical, 735
 - NOARCHIVELOG mode, in, 288
 - privileges needed, 260
 - redo log file groups, planning around, 278
 - segment considerations, 411
 - table backups, checking for, 547
 - tablespaces, read-only, 331, 347–348
 - time kept, setting, 257
 - .bad files, 698
 - BAD SQL*Loader parameter, 703
 - batch processes
 - constraints, disabling during, 642
 - constraints, enabling after, 645, 660
 - rollback segments, 468
 - BEFORE events, 215
 - BEFORE INSERT triggers, 632
 - BEGIN keyword, 212
 - BFILE datatype, 494
 - Bible Series Certification Test Engine (on the CD), 1068
 - BINDSIZE SQL*Loader parameter, 704, 717
 - bitmap indexes, 392, 593–596, 600
 - BITMAP_MERGE_AREA_SIZE initialization parameter, 596
 - BLOB datatype, 229, 397, 494, 498
 - blocks. *See also* storage management
 - addresses, 402–403
 - contents, 402–404
 - corrupt, skipping, 549
 - described, 333
 - export operations, changed during, 746
 - extents, relation to, 328, 401
 - fragmentation, 404
 - free lists, removing from, 405–406, 525, 531
 - free lists, retrieving number in, 547, 548
 - free lists, returning to, 406, 526–527, 531
 - headers, 403
 - high-water mark, 537–541
 - index blocks, contention, 588–589
 - index blocks per key, retrieving number of, 610
 - index blocks, space settings, 597–598, 600
 - index blocks, splitting, 587–588, 589
 - index blocks, threshold percentage, 610
 - index leaf blocks, 585, 586
 - index leaf blocks, headers, 598
 - index leaf blocks, merging, 608
 - index leaf blocks, retrieving number of, 610
 - index leaf blocks, space settings, 597–598
 - index leaf blocks, splits, 587–588
 - operating system blocks, 328, 329, 391
 - queries, blocks changed after starting, 431
 - reading multiple simultaneously, 101
 - reads per call, setting maximum, 811, 818–819
 - rollback segment block usage, 429, 433–434
 - row chaining, 404, 405, 528–530, 548
 - row migration, 525, 527–530, 548
 - row storage in, 492, 493
 - ROWID block number, 501, 554
 - segment data, 403
 - size, 15, 20, 96, 162, 333
 - size, changing, 402
 - space utilization parameters, 405–406, 508–509, 522–523, 525–527, 528–529
 - storage structure, place in, 333, 391
 - table block space deallocation, 537–541
 - table block space, sharing with other objects, 517
 - table block space utilization parameters, 508–509, 522–523, 525–527, 528–529, 531
 - table block space utilization parameters, viewing, 546–549
 - transaction slots, 403, 404–405
 - updates, reserving space for, 525
 - updates, simultaneous, 403
 - books to read, 1125–1126
 - boot loader, 397
 - bootstrap segments, 397–398
 - BUFFER parameter
 - Export utility, 745, 749
 - Import utility, 756, 760
 - BUFFER POOL parameter, 509, 531, 598
 - buffers, dirty, 14, 15, 283, 285. *See also* storage management
 - business rules, 214, 634, 640–641, 642
- ## C
- cache. *See* storage management
 - CACHE parameter, 508, 511, 531
 - calendar format, 1026, 1034
 - Capacity Planner utility, 61
 - case sensitivity, 43
 - CATALOG.SQL script, 232–234, 847
 - CATPROC.SQL script, 235–237
 - CAT*.SQL scripts, 238
 - CD-ROM with this book
 - exam resources, 1068
 - installation, 1067, 1070
 - Portable Document Format (PDF) version of this book, 1070
 - scripts, 1112–1114
 - software, 1065, 1068–1070
 - source code for examples, 1068
 - system requirements, 1066

- CertCities Web site, 1127
- CERTDDBOBJ.SQL (on the CD), 1112, 1113
- Certification Test Engine (on the CD), 1068
- chaining, row, 404, 405, 528–530, 548
- CHAR datatype, 494, 1029
- character sets. *See also* National Language Support (NLS)
 - AUTOEXEC.BAT environment variable, 1036
 - changing, 157, 1030–1031
 - Chinese, 1028
 - choosing, 1030–1031
 - client application setting, 1036
 - client National Language set, translation to/from database set, 1025, 1026
 - database character set, 156
 - database creation, setting at, 165, 177, 1026
 - database, multiple in same, 156–157
 - datatype set, retrieving, 553
 - datatypes, character, 493, 494–495, 1029–1030
 - default, 156
 - described, 156
 - encoding schemes, 1027–1029
 - environment variables related to, 168, 177, 1036
 - exporting data, considerations in, 744, 775, 1045
 - fixed-width, 1027, 1028–1029, 1031
 - GB2312-80, 1028
 - importing data, conversion in, 744, 776
 - JA16EUC, 1029
 - JA16EUCFIXED, 1028–1029, 1030
 - Japanese Extended UNIX Code (JEUC), 1028, 1030
 - languages supported, 1025
 - National Language set, 177, 1026–1031
 - Net8, conversion by, 1025, 1026
 - ROWID, 501
 - single-byte, 1027
 - Unicode, 1027, 1029
 - Universal Character Set Transformation Format character set, 1029
 - Universal Character Set two-byte form, 1029
 - US7ASCII, 168, 177, 1027, 1030, 1031
 - UTF8 character set, 1029
 - varying-width, 1027, 1028
 - WE8DEC, 1027
 - WE8ENCDEC500, 1027
 - WE8ISO8859P1, 1027–1028, 1030, 1031
- CHECK constraints, 640–642
- checkpoint process (CKPT), 6, 16–17, 255
- CHECKPOINT_INTERVAL initialization parameter, 160
- checkpoints
 - automatic, 284
 - column drops, during, 542–543
 - described, 15, 276, 283
 - dirty buffers, role in writing to disk, 15, 283
 - forcing, 283, 346
 - frequency, 98–99, 276, 284, 285
 - initiating, 15
 - log switches, occurring with, 284
 - recovery, role in, 276
 - response delay, 283
 - returning last, 263
 - timeout, 98, 284–285, 286
 - trigger events, 283, 284–285
- CHECKPOINT_TIMEOUT initialization parameter, 160
- Chinese GB2312-80 character set, 1028
- CKPT. *See* checkpoint process (CKPT)
- CLOB datatype, 229, 397, 494, 498, 1029
- Cluster Key, 491, 517–518, 520, 521
- clusters
 - auditing, 938
 - exporting cluster definitions, 741, 742
 - hash clusters, 393, 517, 519–522
 - importing cluster definitions, 742
 - index clusters, 393, 517–519, 610
 - segments, cluster, 393
 - sizing, 518, 520
 - speed, effect on, 517, 519
 - tables, creating on, 491, 517–522
 - tablespaces, 518
- COL\$ table, 334
- collections, 502–506
- columns
 - buckets, retrieving number of, 553
 - character set, retrieving, 553
 - constraint column information, retrieving, 665
 - copying, 512–513
 - data dictionary, removing definition from, 543
 - datatype, retrieving, 552
 - density, retrieving, 553
 - dropping, 542–545
 - fixed-length, 494–495
 - fixed-point number columns, 496
 - index columns, maximum combined size, 583
 - index columns, maximum number, 583
 - index columns, names, 597
 - index columns, referencing in queries, 583
 - index columns, retrieving information about, 611–615
 - index columns, value cardinality, 593, 596
 - length, 492
 - length, retrieving, 552
 - length, retrieving average, 553
 - listing, 553
 - load operations, in, 699

Continued

- columns (*continued*)
 - name, retrieving, 550
 - nulls, retrieving number of, 553
 - ownership, retrieving, 550
 - privileges, column, 542, 915
 - scale information, retrieving, 552
 - sequence number, retrieving, 552
 - statistics, retrieving, 553
 - storage order, 492
 - tables belonging to, retrieving, 552
 - UNUSED, flagging as, 543–544
 - values, retrieving default, 552
 - values, retrieving low/high, 552
 - values, retrieving number of, 552
 - variable-length, 492, 494–495
 - views, 552–553, 665, 930
- comments
 - auditing COMMENT operations, 944
 - exporting, 739, 740, 741, 742
 - importing, 739, 740, 742
 - parameter files, 95, 753
- COMMIT command, 649, 704
- COMMIT Import utility parameter, 756
- COMPATIBLE initialization parameter, 101, 223
- COMPOSITE_LIMIT resource management setting, 811–813
- COMPRESS Export utility parameter, 746, 749, 775
- COMPUTE STATISTICS command, 606
- Configuration Assistant (Oracle Enterprise Manager (OEM)), 65–69
- configurations, stored, 102–103
- CONNECT INTERNAL syntax, 45
- CONNECT role, 980, 994–995
- connections
 - auditing, 933, 940, 950
 - pooling, enabling, 162
 - remote, 45–46
 - time, setting maximum per session, 810, 814
- CONNECT_TIME resource management setting, 810, 811, 814
- CONSISTENT Export utility parameter, 746, 775
- Console, Oracle Enterprise Manager (OEM), 62, 70–76
- constraints. *See* integrity constraints
- CONSTRAINTS parameter
 - Export utility, 746, 777
 - Import utility, 757
- contexts
 - auditing, 938
 - importing/exporting, 742
- control files
 - active, displaying, 261–262
 - backing up, 260–261
 - backup information stored in, 257, 263
 - checkpoint process (CKPT), updating by, 255
 - copying in binary form, 260
 - creating, 257–258
 - datafile information stored in, 256, 257, 263
 - described, 7, 20, 255
 - entries, reusable/nonreusable, 257
 - instance mounting, reading during, 255
 - location, 20, 42, 145, 158, 259
 - moving, 260
 - multiplexing, 145, 259–260
 - number of, 20
 - record information, displaying, 262
 - recovery, role in, 255
 - redo log file information stored in, 256, 257, 258, 263
 - size, 158, 178, 256
 - SQL*Loader, 697, 698–700, 714
 - views displaying control file data, 261–263
 - views using control file data, 263
- CONTROL SQL*Loader parameter, 703
- CONTROL_FILE_RECORD_KEEP_TIME initialization parameter, 257
- CONTROL_FILES initialization parameter, 96, 145, 171, 258, 259–260
- counters, 848
- CPU time, setting maximum
 - per session, 809, 811
 - per Structured Query Language (SQL) statement, 810, 819
- CPU_PER_CALL resource management setting, 810, 819
- CPU_PER_SESSION resource management setting, 809, 811
- CREATE ANY INDEX privilege, 597, 897
- CREATE ANY TABLE privilege, 507
- CREATE_BITMAP_AREA_SIZE initialization parameter, 596, 600
- CREATE CLUSTER command, 518–519
- CREATE CONTROLFILE command, 257–258
- CREATE DATABASE command
 - ARCHIVELOG clause, 176
 - authentication needed, 173
 - base tables creation, 197
 - CHARACTER SET clause, 177, 1029–1030, 1031
 - control file creation, 257–258
 - CONTROLFILE REUSE clause, 175
 - DATAFILE clause, 175
 - Dbname clause, 175
 - file structure necessary for, 142–143
 - LOGFILE GROUP clause, 175–176
 - MAXDATAFILES clause, 176

- MAXINSTANCES clause, 176
 - MAXLOGFILES clause, 176, 280
 - MAXLOGHISTORY clause, 176
 - MAXLOGMEMBERS clause, 176, 280
 - NATIONAL CHARACTER SET clause, 177, 1029, 1030, 1031
 - NOARCHIVELOG clause, 176
 - operating system preparation, 167–170
 - parameter file creation, 170
 - parameters, 175–177
 - permissions needed, 46, 144, 179
 - program unit creation, 209
 - redo log file creation, 279–280
 - scripts to be run after, 232
 - syntax, 174, 178–179, 280
 - SYSTEM tablespace automatically created with, 409
 - troubleshooting, 178–179
 - CREATE GLOBAL TEMPORARY TABLE command, 514–517
 - CREATE INDEX command, 597–601
 - CREATE INDEX privilege, 897
 - CREATE PACKAGE BODY command, 226
 - CREATE PROFILE command, 814–815
 - CREATE PROFILE privilege, 813
 - Create Repository Summary screen, 68
 - CREATE ROLE command, 982–984
 - Create Role dialog box, 985–986
 - CREATE ROLE privilege, 982
 - CREATE ROLLBACK SEGMENT privilege, 438
 - Create Rollback Segment screen, 442
 - CREATE SESSION privilege, 772, 856
 - CREATE TABLE AS SELECT (CTAS) command, 511, 512, 735–737, 743–744
 - CREATE TABLE command, 14, 507–514, 524
 - CREATE TABLE privilege, 507, 772, 897
 - Create Tablespace screen, 344
 - CREATE TEMPORARY TABLESPACE command, 338, 342, 852
 - CREATE USER command, 815, 852–856, 860
 - CREATE USER privilege, 813
 - CREATE VIEW privilege, 895, 907–909
 - createsid.bat file, 165
 - CREATEUSER.SQL (on the CD), 1112, 1114–1119
 - CTAS command. *See* CREATE TABLE AS SELECT (CTAS) command
 - currency display, 1025, 1026, 1033–1034, 1036
- D**
- Data Definition Language (DDL), 197
 - data dictionaries
 - adding, 197
 - auditing, involvement in, 197, 232
 - column definitions, removing from, 543
 - DBA Studio, viewing dictionary information using, 230
 - deleting, 197
 - dynamic performance views, 111–115, 195, 206–208
 - editing, 197
 - export operations, recording in, 748
 - imports, recording in, 759
 - information stored, 196–197
 - location, 302
 - LogMiner dictionary, 302–303
 - ownership, 44
 - packages, data dictionary, 228–230
 - privileges, restricting access using, 913
 - program unit storage in, 196, 210
 - queries against, 198, 205
 - scripts related to, 232–238
 - SYSTEM tablespace reserving for, 334, 362, 409
 - tables, base, 195, 197–199
 - tables, internal, 178
 - tablespaces, dictionary-managed, 334, 337, 344, 360, 401
 - user info, storage of, 196
 - uses, 196–197, 199
 - views, data dictionary, 195
 - Data Management ↔ Load, 706
 - Data Manipulation Language (DML), 197, 587–593, 607, 691–694
 - data, moving between databases. *See* databases, moving data between
 - Data Object Number, ROWID, 501
 - DATA SQL*Loader parameter, 703
 - Data tablespaces, 330, 334, 335, 410, 411
 - data validation using constraints. *See* integrity constraints
 - Data Viewer utility, 61
 - data warehousing, 468
 - database blocks. *See* blocks
 - Database Configuration Assistant. *See* Oracle Database Configuration Assistant
 - database writer (DBW0), 6, 14–16
 - databases. *See also* databases, moving data between; datafiles; tables
 - archiving mode, 21–22, 161, 176, 256
 - compatible parameter, 156
 - creating, character set options at, 165, 177, 1026
 - creating, packages created during, 229–230
 - creating, privileges needed, 44, 46, 144, 179
 - creating, scripts to be run after, 232
 - creating, steps in, 142, 165–166

Continued

- databases (*continued*)
 - creating using CREATE DATABASE command, 165–166, 174–177
 - creating using Oracle Database Configuration Assistant, 149
 - date information, 256
 - deleting, 150, 261
 - disk space requirements, 143, 179
 - entity relational model, 327–328
 - environment types, 152
 - file structure, 7, 18–20, 42–43, 142–143, 144–149
 - Global Database Name, 155, 166
 - identifier, internal, 256, 263
 - instances, association with, 8
 - migration from previous versions, 77–78
 - mounting/dismounting, 104–106
 - name, 96, 155, 175, 256
 - name conflicts, 258
 - name, retrieving, 263
 - objects, database, 848–849
 - opening, 93–94, 103–108
 - operating system, preparing for, 142–143, 167–170
 - optimization utilities, 61
 - read-only mode, 107–108
 - script files used in creating, 164, 177
 - shutting down, 93–94, 103–104, 108–111
 - STAGING databases, 776
 - state, changing, 105–106
 - storage hierarchy, 327–328
 - storage structure, place in, 330, 391
 - structure, logical, 329–333, 391–392
 - structure, physical, 328–329
 - system change number (SCN), retrieving, 263
 - system identifier (SID), 155, 166–167
 - tables, internal, 334
 - tablespaces, relation to, 328, 330–331
 - time information, 256
 - transitional, 776
 - triggers, database event, 222–224
 - users, specifying maximum simultaneous, 153, 169
- databases, moving data between. *See also* Export utility; Import utility
 - ALTER TABLE command, using, 737–738, 743–744
 - constraints during, 736
 - create table as select (CTAS) command, using, 735–737, 743–744
 - datafiles between identical databases, 776
 - disk space requirements, 737
 - Export utility, using, 738, 743–744
 - Import utility, using, 738
 - indexes, rebuilding, 737
 - method, choosing, 743–744
 - reasons for, 734–735
 - scenarios, 734–735, 743–744
 - STAGING databases, using, 776
 - storage options, 737
 - tablespaces, transportable, 776–782
 - time needed, 734
- datafiles
 - backup state, retrieving, 263
 - control file, information stored in, 256, 257, 263
 - database block size, 20
 - date information, 256
 - deleting, 261
 - described, 7, 19–20, 328–329
 - extents, relation to, 328, 329
 - filename, 175
 - headers, 329
 - identifier, retrieving, 360, 361
 - information about, retrieving, 360–361, 407
 - load operations, specifying datafiles written to, 698, 703, 717
 - location, 42, 146–147, 256
 - Logwriter (LGWR) writing to, 15
 - missing, 346
 - moving, 352–358
 - name, 256
 - name, retrieving, 263, 360, 361
 - number of, 98, 158
 - number of, maximum, 98, 176, 256, 257
 - offline, 329
 - operating system blocks, 329
 - operating system path, retrieving, 360
 - Oracle Storage Manager, setup using, 344
 - path, retrieving, 361
 - recovery, 346
 - resizing, 329
 - reusing existing, 340
 - rollback segments datafile location, 146
 - size, autoextend option, 19, 339–340, 350–351
 - size, changing manually, 351
 - size, retrieving, 360, 407
 - size, retrieving settings, 360–361
 - size, specifying, 19, 175
 - space usable, retrieving, 360
 - status, retrieving, 360, 361, 407
 - storage structure, place in, 328–329, 391
 - system change number (SCN), 256
 - SYSTEM datafiles, moving, 356–358
 - tablespace creation, configuring during, 339–340
 - tablespace name, retrieving, 360, 407
 - tablespace number, retrieving, 263

- tablespaces, adding to, 351
- tablespaces, listing datafiles in, 778
- tablespaces, multiple per, 340
- tablespaces, relation to, 328
- tablespaces, sizing appropriately for, 159
- time information, 256
- views, 360–361, 407
- datatypes. *See also specific datatypes*
 - binary data storage, 497, 498
 - built-in, 493–494
 - character set, 493, 494–495, 1029–1030
 - character set, retrieving, 553
 - character types, 494–495
 - fixed-length, 494, 497
 - large object (LOB) datatypes, 229, 397, 498–499
 - numeric types, 496
 - retrieving column datatype, 552
 - scalar, 493–494
 - time, 497
 - user-defined, 849
 - variable-length, 494, 496
 - VARRAYs, 502–504
- DATE datatype, 493, 497
- dates
 - National Language Support (NLS) settings, 1025, 1026, 1033, 1034, 1035
 - object creation date, retrieving, 545
 - time storage with, 497
- DBA_data dictionary views, 198, 200–202
- DBA role, 43, 44, 978
- DBA Studio
 - described, 77
 - General page, 602
 - index creation using, 601–604
 - index information, retrieving using, 615–616
 - interface, 77
 - Management Server connection, 63
 - object information, viewing using, 230
 - Options page, 603–604
 - profile management using, 823
 - rollback segment operations, 442, 455, 460
 - Storage page, 603
 - table creation using, 514
- DBA_AUDIT_EXISTS view, 950
- DBA_AUDIT_OBJECT view, 950
- DBA_AUDIT_SESSION view, 950
- DBA_AUDIT_STATEMENT view, 950
- DBA_AUDIT_TRAIL view, 950–952
- DBA_COL_PRIVS view, 930
- DBA_CONS_COLUMNS view, 665
- DBA_CONSTRAINTS view, 663–665
- DBA_DATA_FILES view, 360–361, 407
- DBA_DEPENDENCIES view, 866
- DBA_EXTENTS view, 407, 551–552
- DBA_FREE_SPACE view, 407
- DBA_IND_COLUMNS view, 611–615, 666
- DBA_INDEXES view, 605, 609–611, 666
- DBAKNOW (on the CD), 1069
- DBA_OBJECTS view, 231, 545–546, 774
- DBA_PROFILES view, 824–826
- DBA_ROLE_PRIVS view, 1000
- DBA_ROLES view, 999–1000
- DBA_ROLLBACK_SEGS view, 443–445
- DBArtisan (on the CD), 1069
- DBA_SEGMENTS view, 407, 549–550
- DBA_SYS_PRIVS view, 910–912, 1001
- DBA_TAB_COLUMNS view, 552–553
- DBA_TABLES view, 529, 532, 537, 546–549
- DBA_TABLESPACES view, 360, 407
- DBA_TAB_PRIVS view, 929
- DBA_TS_QUOTAS view, 870–871
- DBA_USERS view, 336, 869
- DB_BLOCK_BUFFERS initialization parameter, 97, 162
- DB_BLOCK_SIZE initialization parameter, 11, 96, 172, 333
- DB_FILE_MULTIBLOCK_READ_COUNT initialization parameter, 101, 511, 522–523
- DB_FILES initialization parameter, 98
- DBMS_APPLICATION_INFO errors, 235
- DBMS_LOB package, 229
- DBMS_LOGMNR.ADD_LOGFILE procedure, 303
- DBMS_LOGMNR_D.BUILD procedure, 302
- DBMS_LOGMNR.START_LOGMNR package, 303
- DBMS_PIPE, 979
- DBMS_RLS access, role needed, 979
- DBMS_ROWID package, 229, 501, 553–554
- DBMS_SESSION package, 229
- DBMS_SESSION.SET_ROLE procedure, 996–997
- DBMS_SHARED_POOL package, 230, 238
- DBMS_SPACE package, 229
- DBMS_SPACE.UNUSED_SPACE procedure, 537, 538–539
- DBMS*.SQL scripts, 238
- DBMS_TTS.ISSELFCONTAINED function, 782
- DBMS_TTS.TRANSPORT_SET_CHECK procedure, 780
- DBMS_UTILITY package, 229
- DB_NAME initialization parameter, 96, 171
- DBSNMP user, 847, 981
- DBW0. *See* database writer (DBW0)
- DB_WRITER_PROCESSES parameter, 15
- DDL statements, 428
- DEALLOCATE command, 540–541
- Decision Support System (DSS) environment, 152

Dedicated Server mode, 154
 DEFAULT TABLESPACE CREATE USER parameter, 855
 DELETE operations
 auditing, 944, 945
 rollback segments, information recorded in, 429, 467
 DELETE_CATALOG_ROLE role, 936, 979
 DELETING trigger predicate, 219
 dependencies
 retrieving, 866–867
 tracking, 232
 Dependencies dialog box, 867
 DESCRIBE command, 112, 810
 DESTROY Import utility parameter, 757
 diagnostic tools, 61
 dictionaries. *See* data dictionaries
 DICTIONARY view, 205–206
 dimensions
 auditing, 939
 exporting, 331
 importing, 740, 743
 DIRECT parameter
 Export utility, 746
 SQL*Loader, 705
 directory aliases, importing/exporting, 742
 DISABLE CONSTRAINT command, 659
 DISCARD SQL*Loader parameter, 703
 DISCARDMAX SQL*Loader parameter, 703, 717
 Discovery Results dialog box, 75
 Discovery Service, 63
 Discovery Wizard, 73–75
 dispatcher process (Dnnn), 17
 DISTRIBUTED_TRANSACTIONS initialization parameter, 18
 DLL. *See* Data Definition Language (DLL)
 DML. *See* Data Manipulation Language (DML)
 Dnnn. *See* dispatcher process (Dnnn)
 documentation manuals listed, 1126–1127
 downloading Oracle trial version, 1065–1066
 DROP ANY COLUMN privilege, 542
 DROP ANY TABLE privilege, 542
 DROP COLUMN command, 542–545
 DROP INDEX command, 608
 DROP PROFILE privilege, 813
 DROP TABLE command, 514, 542
 DROP TABLESPACE command, 358–359
 DROP UNUSED COLUMNS command, 544
 DROP USER command, 866–867
 DSS environment. *See* Decision Support System (DSS) environment
 dynamic performance views, 111–115, 195, 206–208

E

Edit role dialog box, 990, 991
 Edit User dialog box
 introduced, 865
 Object Privileges tab, 917, 928
 Role tab, 993, 995–996
 System Privileges tab, 903, 909
 Embarcadero Technologies, 1069
 ENABLE CONSTRAINT command, 659
 ENABLE_QUERY_REWRITE initialization parameter, 584
 encoding, character set, 1027–1029
 encryption, authentication using, 46
 Enterprise Manager. *See* Oracle Enterprise Manager
 entity relational model, 327–328
 environment variables
 NLS_CREDIT, 1036
 NLS_DEBIT, 1036
 NLS_LANG, 1035–1036, 1045
 NLS_LIST_SEPARATOR, 1036
 NLS_MONETARY_CHARACTERS, 1036
 NLS_NCHAR, 1036
 NLS_TIME_FORMAT, 1038
 NLS_TIMESTAMP_FORMAT, 1038
 NLS_TIMESTAMP_TZ_FORMAT, 1038
 NLS_TIME_TZ_FORMAT, 1038
 ORACLE_BASE, 148, 168
 ORACLE_HOME, 50, 148, 168
 ORACLE_SID, 50, 168, 233, 860
 ORA_NLS33, 168, 177
 PATH, 168
 UNIX systems variables, 167–169
 ERRORS SQL*Loader parameter, 704, 717
 ER/Studio (on the CD), 1069
 Event Management Service, 64
 Event Viewer, 933
 events, trigger, 214–215
 exam
 Candidate Guide, 1104
 CD-ROM, resources on, 1068, 1104
 objectives, 1095–1102
 preparing for, 1103–1104
 registering for, 1104
 retaking, 1105
 writing, 1104–1105
 EXCEPTION keyword, 213
 EXECUTE_CATALOG_ROLE role, 979
 EXP. *See* Export utility (EXP)
 EXP command, 749
 EXP USERID command, 749
 EXP_FULL_DATABASE role, 739, 747, 753, 979

- Export utility (EXP). *See also* exporting data
 - Command Line mode, 745–749, 751–753
 - COMPRESS parameter, 746, 749, 775
 - CONSISTENT parameter, 746, 775
 - CONSTRAINTS parameter, 746, 777
 - Database mode, 741–743
 - export file creation process, 738
 - FEEDBACK parameter, 747
 - FILESIZE parameter, 747
 - help window display, 747
 - Interactive mode, 745, 749–751
 - introduced, 58–59
 - National Language Support (NLS) settings,
 - using, 1045
 - Oracle Enterprise Manager, using through, 753–755
 - OWNER parameter, 747
 - parameters, 745–749
 - parameters, storing in file, 748, 751–753, 775
 - passwords, 753
 - privileges needed, 739, 979
 - QUERY parameter, 748
 - STATISTICS parameter, 748
 - Table mode, 739, 748
 - Tablespace mode, 741
 - TRIGGERS parameter, 777
 - User mode, 739–741, 747
- exporting data. *See also* Export utility (EXP)
 - application contexts, 742
 - auditing information, 739, 740, 742
 - backups, using for, 741
 - block data conversions, 744
 - blocks changed during, 746
 - character set considerations, 744, 775, 1045
 - cluster definitions, 741, 742
 - comments, 739, 740, 741, 742
 - compression, 524, 746, 775
 - constraints, 739, 740, 741, 742, 746
 - constraints in transportable tablespace exports,
 - 780–781
 - conventional path, 744, 749
 - data dictionary, recording incremental
 - exports in, 748
 - database links, 742
 - databases, full, 741–743, 747, 753
 - default method, 744
 - dimensions, 740, 743
 - direct path, 744, 746, 1045
 - directory aliases, 742
 - evaluation buffer, data transfer to, 744
 - export file filename, 747
 - export file location, 775
 - export file, setting maximum size, 747, 749
 - export file, setting record length, 748
 - Export utility, using, 738, 745–753
 - functions, 740, 742
 - incremental, 747, 748
 - indexes, 739, 740, 741, 742, 747
 - indextypes, 740, 743
 - job queues, 740, 743
 - libraries, foreign function, 742
 - log file name, 747
 - memory requirements, 775
 - mode, specifying, 747, 748
 - modes, 738–744
 - National Language Support (NLS) considerations,
 - 775–776
 - object type definitions, 739, 740, 741
 - objects, procedural, 740
 - operators, 740, 742
 - Oracle Enterprise Manager, using, 753–755
 - packages, 740, 742
 - password history, 743
 - privileges needed, 739, 740, 741, 745, 747
 - procedures, 740, 742
 - progress meter option, 747
 - queries, applying in table-level exports, 748
 - READ ONLY transactions, running as, 746
 - refresh groups, 740, 743
 - resource costs, 742
 - role grants, 742
 - roles, 742
 - roles needed for, 740, 741, 745, 747, 979
 - rollback segment definitions, 742
 - rows changed during, 746
 - rows processed per array fetch, setting, 745
 - scheduling, 745, 755
 - security policies, 739, 740
 - sequence numbers, 742
 - snapshot logs, 740, 743
 - snapshot too old errors, 746, 775
 - snapshots, 740, 743
 - SQL COMMAND processing layer, 744
 - synonyms, 740, 742
 - SYS user objects, 741
 - system auditing, 743
 - system privilege grants, 742
 - table data, 739, 740, 742, 748
 - table definitions, 739, 740, 741, 742, 746
 - table grants, 739, 740, 741, 742
 - tables, 517, 524
 - tablespace definitions, 742, 748–749

Continued

exporting data (*continued*)
 tablespace quotas, 742
 transportable tablespace exports, 776–782
 triggers, 739, 740, 741, 743
 type, specifying, 746
 types of export, 744–745
 user definitions, 742
 views, 740, 742, 743

extents
 allocation, automatic, 401–402
 allocation, manual, 401–402, 532–534, 605
 blocks, relation to, 328, 401
 datafiles, relation to, 328, 329
 deallocation, 437, 460–462, 605
 described, 332–333
 dictionary management, 334, 337, 344, 360, 401
 dropping, 401
 file ID, retrieving, 551
 free, returning information about, 408
 indexes, 599, 605, 610
 INSERT operations, management during,
 689, 691, 695
 INSERT operations, space allocation in,
 591, 689, 695
 load operations from existing tables, management
 during, 689, 691, 695
 load operations of external data, management
 during, 713, 714, 716
 local management, 334, 337, 344, 360, 401
 location, 332
 mixing large and small, 335
 name, retrieving, 551
 ownership, retrieving, 551
 rollback segment extents, 429, 434–436
 rollback segment extents, deallocation,
 437, 460–462
 rollback segment extents, maximum, 436, 439,
 441, 463
 rollback segment extents, minimum, 400, 436, 438,
 440, 467
 rollback segment extents, minimum possible, 460
 rollback segment extents, retrieving information
 about, 444, 445, 448–450
 segments, automatic allocation to, 401–402
 segments, manual allocation to, 401–402
 segments, maximum allocated to, 341
 segments, number created with, 341
 segments, relation to, 328, 333
 segments, retrieving number in, 550
 size, returning, 551

sizing, 332, 335, 337, 341–342, 348–349
 space, contiguous, 333
 storage clause settings, 341, 398, 400
 storage structure, place in, 332–333, 391
 table extents, listing, 551–552
 table extents parameters, 509, 510–511, 522,
 531–534
 tablespace information, returning, 360, 551
 tablespaces, in temporary, 338
 tablespaces, relation to, 332
 truncating, 401
 type, retrieving, 551
 views, 408, 551–552

EXTERNAL_NAME view, 870

F

FAILED_LOGIN_ATTEMPTS profile setting, 806
 FAST_START_IO_TARGET initialization parameter,
 16, 99, 283, 285

FEEDBACK parameter
 Export utility, 747
 Import utility, 757

FET\$ table, 334

FILE parameter
 Export utility, 747, 749
 Import utility, 757, 760
 SQL*Loader, 705, 715

file structure, 7, 18–20, 42–43, 142–143, 144–149

FILESIZE parameter
 Export utility, 747
 Import utility, 757

FOREIGN KEY constraints, 542, 633, 638–640, 651–654

fragmentation
 blocks, 404
 free space, 403–404
 rollback segments, 440
 SYSTEM tablespace, 337, 411
 tablespaces, 411

free lists. *See also* storage management
 blocks, removing from, 405–406, 525, 531
 blocks, retrieving number in, 547, 548
 blocks, returning to, 406, 526–527, 531
 described, 407
 groups, 547, 550
 index free list information, retrieving, 610
 INSERT operations, relation to, 407, 689
 PCTFREE parameter, relation to, 405–406, 525, 531
 PCTUSED parameter, relation to, 406, 526–527, 531
 tables, assigning to, 509, 524, 534
 tables, retrieving number assigned to, 547, 550

FROMUSER Import utility parameter, 757, 767, 779

FULL parameter

Export utility, 747

Import utility, 758, 760

functions

auditing, 939

exporting, 740, 742

importing, 740, 742

indexes, function-based, 584, 591–593, 601, 611, 1041–1042

libraries, importing/exporting foreign function, 742

National Language Support (NLS) parameters, using in, 1042–1044

objects, as, 849

password complexity rules, using to enforce, 805, 808–809

PL/SQL functions, 213–214

roles needed for execution, 979

tables, on temporary, 517

user-defined, 211

G

GB2312-80 character set, 1028

Global Database Name, 155, 166

GLOBAL TEMPORARY tables, 411

GRANT ANY ROLE privilege, 992

GRANT command

privilege assignment, 893, 914–916

role assignment, 987, 991–992

GRANTS parameter

Export utility, 747, 749, 777

Import utility, 758, 760

groups

names, 982–983

roles, granting to, 982–983

GV\$ views, 112, 208

H

hardware requirements, 143

hash addresses, 520, 521

hash buckets, 520, 521

hash clusters, 393, 517, 519–522

hash partitioning, 395–396

HASHKEYS parameter, 521, 522

headers

blocks, 403

datafiles, 329

index leaf blocks, 598

rollback segments, 427, 428, 433, 444, 467

rows, 492, 493

segment header information, retrieving, 550

triggers, 215

HELP parameter

Export utility, 747

Import utility, 758, 762

high-water mark, 537–541

home directories, multiple, 37, 43

I

IDENTIFIED CREATE USER parameter, 854

IDLE_TIME resource management setting, 810

IFILE initialization parameter, 102

IGNORE Import utility parameter, 758, 760, 774

IMMEDIATE shutdown mode, 94, 110, 111

IMP. *See* Import utility (IMP)

imp command, 760

imp userid command, 760

IMP_FULL_DATABASE role, 756, 980

Import utility (IMP). *See also* databases, moving data between; importing data

ANALYZE parameter, 756

cluster loading, 519, 521

Command Line mode, 756, 762–770

COMMIT parameter, 756

DESTROY parameter, 757

Export utility files needed, 738, 756

FROMUSER parameter, 757, 767, 779

help window display, 758, 762

IGNORE parameter, 758, 760, 774

INDEXFILE parameter, 758–759, 768

Interactive mode, 756, 760–762

introduced, 58–59

National Language Support (NLS) settings, using, 1045

Oracle Enterprise Manager, using through, 756, 770–772

parameters, 756–760

parameters, storing in file, 759, 775

RECALCULATE_STATISTICS parameter, 759

role needed to use, 980

SHOW parameter, 759, 760, 763–767

statistics, generating, 759

TTS_OWNERS parameter, 760

importing data. *See also* load operations, external data

ANALYZE command, executing during, 756

application contexts, 742

array insert buffer size, setting, 756

auditing information, 739, 740, 742

character set translation, 744, 776

cluster definitions, 742

comments, 739, 740, 742

commits, issuing, 756

constraints, 739, 740, 742, 757, 773–774

Continued

- importing data (*continued*)
 - data dictionary, recording incremental
 - import in, 759
 - database links, 742
 - databases, full, 758
 - datafiles, reusing existing, 757
 - dimensions, 740, 743
 - directory aliases, 742
 - dump file size, setting maximum, 757
 - functions, 740, 742
 - import file contents, displaying, 759, 763–767
 - import file filename, 757, 771
 - import file record length, specifying, 759
 - import file size, setting maximum, 760
 - incremental, recording in data dictionary, 759
 - incremental, specifying type, 758
 - indexes, 739, 740, 742, 758–759, 768–770
 - indextypes, 743
 - job queues, 740, 743
 - libraries, foreign function, 742
 - log file name, 759
 - memory requirements, 775
 - National Language Support (NLS) considerations, 775–776
 - object grants, 758
 - object order, 772–774
 - object placement, 774–775
 - object status, verifying after, 774
 - object type definitions, 739, 740
 - objects, procedural, 740
 - operators, 740, 742
 - order of, 772–774
 - ownership, changing, 757, 760, 767, 779
 - packages, 740, 742
 - password history, 743
 - permissions, 739, 740
 - privileges needed, 756, 772
 - procedures, 740, 742
 - progress meter option, 757
 - refresh groups, 740, 743
 - resource costs, 742
 - roles, 742
 - roles needed for, 756, 772, 980
 - rollback segment definitions, 742
 - scheduling, 772
 - security policies, 739, 740
 - sequence numbers, 742
 - snapshot logs, 740, 743
 - snapshots, 740, 743
 - statistics, generating, 759
 - statistics, loading from export file, 756
 - synonyms, 740, 742
 - system auditing, 743
 - system privilege grants, 742
 - table creation errors, ignoring, 758
 - table data, 739, 740, 742, 759
 - table definitions, 739, 740, 742
 - table grants, 742
 - tables, into existing, 774
 - tables, specifying, 759
 - tablespace conflicts, 757
 - tablespace definitions, 742, 760
 - tablespace mode imports, 760
 - tablespace quotas, 742
 - triggers, 739, 740, 743
 - user definitions, 742
 - users, from/to, 757, 760, 767, 779
 - views, 740, 742, 743
- inactivity timeout, setting, 810
- INCTYPE parameter
 - Export utility, 747
 - Import utility, 758
- INDEX privilege, 597
- indexes
 - auditing, 939, 944
 - bitmap indexes, 392, 593–596, 600
 - blocks, contention for, 588–589
 - blocks, retrieving number per key, 610
 - blocks, space settings, 597–598, 600
 - blocks, splitting, 587–588, 589
 - blocks, threshold percentage, 610
 - branch elements, 585
 - B-tree indexes, 392, 585–587
 - B-tree indexes, bitmaps compared, 593, 596
 - B-tree indexes, converting to Reverse Key, 606–607
 - B-tree indexes, creating, 597–600
 - buffer pool information, retrieving, 611
 - buffer pool setting, 598
 - cache parameters, 598
 - clustering factor, retrieving, 610
 - clusters, index, 393, 517–519
 - coalescing, 608
 - columns information, retrieving, 611–615
 - columns, maximum combined size, 583
 - columns, maximum number of, 583
 - columns, names, 597
 - columns, referencing in queries, 583
 - columns, value cardinality, 593, 596
 - compound, 583
 - compression, 596
 - compression information, retrieving, 609
 - concatenated, 582–583, 597, 598–599, 609

- constraints, creating before, 659
- on constraints, DISABLE VALIDATE, 645
- constraints, dropped when disabling, 660–661
- constraints, dropped when dropping, 608–609
- on constraints, FOREIGN KEY, 652–653
- constraints index information, retrieving, 666
- on constraints, PRIMARY KEY, 608–609, 638, 645, 649–651, 659
- on constraints, UNIQUE, 608–609, 637, 645, 649–651, 659
- creating, 597–604
- creating in other schemas, 897
- Data Manipulation Language (DML) operations
 - involving, 587–593, 607
- DBA Studio, creating using, 601–604
- DBA Studio, retrieving information using, 615–616
- deletes, 588, 606
- described, 581–582
- dropping, 608–609
- elements, logical, 582–584
- elements, physical, 585
- entries, retrieving number of, 610
- entry header element, 586, 594
- exporting, 739, 740, 741, 742, 747
- extent allocation/deallocation, 605
- extent information, retrieving, 610
- extent sizes, 599
- free list information, retrieving, 610
- function-based, 584, 591–593, 601, 611, 1041–1042
- generated, listing, 611, 614
- importing index information from export file in
 - ASCII format, 758, 768–770
- importing indexes, 739, 740, 742, 758–759
- importing indextypes, 743
- INITIAL parameter, 598, 610
- INITRANS parameter, 598, 599, 606
- input/output (I/O) operations, 588–590
- inserts, 581, 598, 688, 695
- instances scanned, retrieving number of, 610
- INVALID, 612
- Key column values, 586
- keys, retrieving number of, 610
- leaf blocks, 585, 586
- leaf blocks, headers, 598
- leaf blocks, merging, 608
- leaf blocks, retrieving number of, 610
- leaf blocks, space settings, 597–598
- leaf blocks, splits, 587–588
- leaf node elements, 585
- linguistic indexes, 1041–1042
- locking, 586, 595–596
- logging parameters, 598, 599, 603
- logging parameters, retrieving, 610
- maintenance costs, 582
- MAXTRANS parameter, 598, 606, 610
- MINEXTENTS parameter, 598, 610
- name, 597, 602
- name, retrieving, 609, 611, 614
- NEXT parameter, 598
- NOSORT option, 598, 600
- null values in, 587, 636
- objects, as, 848
- online, rebuilding, 607–608
- Oracle Enterprise Manager (OEM), creating
 - using, 601–604
- Oracle Enterprise Manager (OEM), retrieving
 - index info using, 615–616
- over indexing, 599
- ownership, 597
- ownership, retrieving, 609, 611, 612–613
- parallelism, default degree, 610
- partitioning, 585
- partitioning information, retrieving, 611
- PCTFREE parameter, 588, 597–598, 600, 606
- PCTINCREASE parameter, 598, 610
- PCTUSED parameter, 598, 605
- privileges needed for creating/altering, 597, 604, 897
- queries, 581, 582
- queries, listing index information using, 614–615
- queries, multiple predicate, 593, 594, 596
- queries, referencing columns in, 583
- queries, Structured Query Language (SQL), 587
- queries using bitmap, 593
- queries using function-based, 584, 591–592
- queries using Reverse Key, 590
- rebuilding, 536, 605–608, 737
- reorganizing, 588, 604–609
- Reverse Key, 588–591, 600–601
- Reverse Key, converting to B-tree, 606–607
- root level element, 585
- ROWID datatype, using instead of, 500
- ROWIDs, use in, 582, 586, 594, 596
- sample size, retrieving, 610
- schema, 597
- secondary, 611
- segments, as, 392, 581
- single-column, 582–583
- sorting options, 597, 598, 599, 600

Continued

- indexes (*continued*)
 - statistics, 606, 611, 613
 - status, retrieving, 610
 - storage parameters, 598, 603, 604–605
 - storage requirements, 581, 582
 - Structured Query Language (SQL)
 - statements, in, 587
 - tables, index-organized, 393, 490, 523–524, 548
 - tables, moved, 606
 - tables, on temporary, 517
 - tables, retrieving name, 614
 - tables, specifying, 597, 602
 - tablespaces, 330, 335, 599, 602, 605
 - tablespaces, relocating to different, 523
 - tablespaces, retrieving, 609
 - tablespaces, spanning, 585
 - tablespaces, specifying, 597
 - tablespaces, temporary, 599
 - temporary duration, retrieving, 611
 - temporary status, retrieving, 611
 - transaction slot parameters, 598, 599
 - transaction slot parameters, retrieving, 609–610
 - type, choosing, 596
 - type, retrieving, 609, 611
 - unique/nonunique, 583–584, 597, 649–651
 - updates to values, 588
 - uses of, 581
 - views, 605, 609–615, 666
- INDEXES parameter
 - Export utility, 747
 - Import utility, 758
- INDEXFILE Import utility parameter, 758–759, 768
- INDEX_STATS view, 606, 609
- INFILE keyword, 700
- INITIAL parameter
 - default, 341, 400
 - described, 341
 - indexes, 598, 610
 - precedence, 398
 - retrieving, 444, 547, 550, 610
 - rollback segments, 441
 - setting, 441
 - tables, 509, 510, 532, 547
- INITIALLY DEFERRED constraints option, 647, 656
- INITIALLY IMMEDIATE constraints option, 647, 656
- INITIAL_RSRC_CONSUMER_GROUP view, 870
- INIT.ORA file
 - AUDIT_TRAIL parameter, 937
 - BACKGROUND_DUMP_DEST parameter, 100, 125, 163
 - BITMAP_MERGE_AREA_SIZE parameter, 596
 - CHECKPOINT_INTERVAL parameter, 160
 - CHECKPOINT_TIMEOUT parameter, 160
 - comments, 95
 - COMPATIBLE parameter, 101, 223
 - CONTROL_FILE_RECORD_KEEP_TIME parameter, 257
 - CONTROL_FILES parameter, 96, 145, 171, 258, 259–260
 - CREATE_BITMAP_AREA_SIZE parameter, 596, 600
 - creating manually, 170–173
 - creating using Oracle Database Configuration Assistant, 149
 - DB_BLOCK_BUFFERS parameter, 97, 162
 - DB_BLOCK_SIZE parameter, 11, 96, 172, 333, 400
 - DB_FILE_MULTIBLOCK_READ_COUNT parameter, 101, 511, 522–523
 - DB_FILES parameter, 98
 - DB_NAME parameter, 96, 171
 - described, 21
 - DISTRIBUTED_TRANSACTIONS parameter, 18
 - ENABLE_QUERY_REWRITE parameter, 584
 - FAST_START_IO_TARGET parameter, 16, 99, 285
 - IFILE parameter, 102
 - language settings, 1032–1035
 - LARGE_POOL_SIZE parameter, 97, 162
 - location, 21, 171
 - LOG_ARCHIVE_DEST parameter, 22, 99, 289
 - LOG_ARCHIVE_FORMAT parameter, 99
 - LOG_ARCHIVE_MAX_PROCESSES parameter, 17
 - LOG_ARCHIVE_START parameter, 22, 99, 289
 - LOG_BUFFER initialization parameter, 98, 162
 - LOG_BUFFER parameter, 98, 162
 - LOG_CHECKPOINT_INTERVAL parameter, 16, 98–99, 284, 286
 - LOG_CHECKPOINT_TIMEOUT parameter, 16, 98, 284–285, 286
 - MAX_DUMP_FILE_SIZE parameter, 100
 - MAX_ROLLBACK_SEGMENTS parameter, 454
 - moving, 171
 - MTS_DISPATCHERS parameter, 17
 - MTS_MAX_DISPATCHERS parameter, 17
 - MTS_MAX_SERVERS parameter, 17
 - MTS_SERVERS parameter, 17
 - NLS_CALENDAR parameter, 1034, 1042
 - NLS_CHARACTERSET parameter, 1048, 1049
 - NLS_CURRENCY parameter, 1033, 1034, 1042
 - NLS_DATE_FORMAT parameter, 1033, 1034
 - NLS_DATE_LANGUAGE parameter, 1033, 1042
 - NLS_ISO_CURRENCY parameter, 1033, 1034, 1042
 - NLS_LANGUAGE parameter, 1033, 1049
 - NLS_LOWER parameter, 1042

- NLS_NCHAR_CHARACTERSET parameter,
 - 1048, 1049
- NLS_NUMERIC_CHARACTERS parameter, 1033, 1034, 1042
- NLS_SORT parameter, 1033, 1038–1041, 1049
- NLS_TERRITORY parameter, 1033, 1049
- NLS_UPPER parameter, 1042
- O7_DICTIONARY_ACCESSIBILITY parameter, 913
- OPTIMIZER_MODE parameter, 21
- OS_AUTHENT_PREFIX parameter, 101, 861–862
- OS_ROLES parameter, 982
- parameters, displaying information about, 118
- parameters, required, 96, 171–172
- PROCESSES parameter, 100, 162
- QUERY_REWRITE_ENABLED parameter, 584, 592
- REMOTE_LOGIN_PASSWORDFILE parameter, 47, 51, 52, 101, 899
- REMOTE_OS_AUTHENT parameter, 861, 862–863
- RESOURCE_LIMIT parameter, 101, 804–805, 809
- ROLLBACK_SEGMENTS parameter, 99–100, 432, 454
- SHARED_POOL_SIZE parameter, 9, 97, 162
- SORT_AREA_SIZE parameter, 98, 330, 337, 599
- SQL_TRACE parameter, 100–101
- TRANSACTIONS_PER_ROLLBACK_SEGMENT parameter, 100, 454, 455
- USER_DUMP_DEST parameter, 100, 163
- values, changing, 115
- values, changing dynamically, 115, 116–119
- values, displaying, 115–116
- values, specifying, 94
- INTRANS parameter
 - default, 404
 - indexes, 598, 599, 606
 - retrieving settings, 547
 - setting, ideal, 404–405
 - tables, 508, 531
- INIT<SID>.ORA file, 21, 165, 171
- input/output (IO) activity, setting maximum per session, 803, 804
- INSERT ANY privilege, 774
- INSERT operations
 - ALTER SESSION parameters, 693
 - APPEND, 511–512, 519, 690
 - APPEND hints, 690, 693
 - auditing, 944
 - constraint behavior, 688, 694, 695
 - extents management, 689, 691, 695
 - free list, relation to, 407, 689
 - index behavior, 581, 598, 688, 695
 - instance failure, 695
 - INTO ... SELECT statements, 687, 688
 - loads, conventional, 687, 689
 - loads, direct, 687, 688–696
 - logging options, 12, 511–512, 690–691, 695
 - parallel direct-load, 691–694
 - rows, one at a time, 688
 - simultaneous, 407
 - space allocation, 689, 691, 695
 - subqueries, using, 688
 - tables, clustered, 694
 - tables with LOB columns, 694
 - tables with object columns, 694
- INSERT triggers, 214, 215, 219, 694, 712–714
- INSERT_DATA.SQL (on the CD), 1112, 1119–1124
- INSERTING trigger predicate, 219
- installation
 - components, checking installed, 41
 - components, uninstalling, 41–42
 - dependency checking, 37
 - destination, 40
 - home directories, multiple, 37, 43
 - silent option, 38
 - source files location, 39, 40
 - unattended, 38
 - UNIX (Sun Solaris) systems, 38, 48
 - Web-based, 38
 - Windows NT/2000 systems, 38
- Instance Manager, 60, 63, 102–103
- instances
 - aborting, 105, 110–111
 - auditing, enabling for, 937–938
 - connecting to, 22–24
 - connection time, setting maximum per session, 810, 814
 - CPU time, setting maximum per session, 809
 - databases, association with, 8
 - described, 6
 - inactivity timeout, setting, 810
 - indexes, retrieving number of instances scanned, 610
 - input/output (IO) activity, setting maximum per session, 803, 804
 - INSERT operations, failure during, 695
 - integrity, 13
 - load operations, failure during, 695
 - locking, inter-instance, 18
 - logon credentials, 57
 - memory requirements, 143
 - memory structure, shared, 6, 7–8, 9
 - modes, changing, 105–106
 - mounting, control file reading during, 255

Continued

- instances (*continued*)
 - name, 8
 - number of, maximum, 258
 - Oracle Intelligent Agent connection, 847
 - privileges for starting/stopping, 46, 104–105
 - recovery, 255, 275–276, 285
 - rollback segment instance, retrieving, 444
 - service, starting Oracle with, 106–107
 - SQL*Plus, connecting to using, 57
 - starting, 93–94, 103–108, 173–174
 - stopping, 93–94, 103–104, 108–111
- Instant Message for Oracle v2.4 (on the CD), 1069
- INSTMSG (on the CD), 1069
- integrity constraints
 - batch processing, disabling during, 642
 - batch processing, enabling after, 645, 660
 - benefits of, 633–634
 - business rules, 634, 640–641, 642
 - CASCADE CONSTRAINTS option, 358–359, 542, 639–640, 652, 925
 - centralizing, 634
 - CHECK, 640–642
 - checking after Data Manipulation Language (DML) statements, 647
 - checking upon commit, 647
 - column information, retrieving, 665
 - database moving, during, 736
 - deferrable status, retrieving, 664
 - deferrable/nondeferrable, 647–649, 656, 659
 - described, 631–633
 - DISABLE VALIDATE state, 644–645, 660
 - DISABLED NOVALIDATE state, 643–644, 659–660
 - disabling/enabling, 608–609, 642–647, 656, 659, 661–663
 - dropping, 608–609
 - ENABLE NOVALIDATE state, 645–646, 660–661
 - ENABLE VALIDATE state, 646–647, 661–663
 - error messages, identification in, 655, 658–659
 - exceptions tables, enabling using, 661–663
 - exporting, 739, 740, 741, 742, 746
 - FOREIGN KEY, 542, 633, 638–640, 651–654
 - FOREIGN KEY, self-referencing, 639
 - generation by Oracle, 664
 - hierarchy relationships, enforcing using, 639
 - implementing, 654–659
 - importing, 739, 740, 742, 757, 773–774
 - indexes, creating before, 659
 - indexes dropped when disabling, 660–661
 - indexes dropped when dropping, 608–609
 - indexes information, retrieving, 665
 - indexes on DISABLE VALIDATE constraints, 645
 - indexes on FOREIGN KEY constraints, 652–653
 - indexes on PRIMARY KEY constraints, 608–609, 638, 645, 649–651, 659
 - indexes on UNIQUE constraints, 608–609, 637, 645, 649–651, 659
 - INITIALLY DEFERRED option, 647, 656
 - INITIALLY IMMEDIATE option, 647, 656
 - in-line (column-level), 654, 655–656
 - INSERT operations, during, 688, 694, 695
 - load operations from existing tables, during, 688, 694, 695
 - load operations of external data, during, 699, 712, 714, 716
 - LONG columns in, 498
 - modifying, 647, 659–663
 - name, retrieving, 663, 664, 665
 - naming, 655, 658–659
 - NOT NULL, 587, 634–636, 639, 659
 - objects, as, 848
 - ON DELETE CASCADE option, 656, 664
 - out-of-line (table-level), 654, 656
 - ownership, retrieving, 663, 665
 - PRIMARY KEY, 490, 584, 608, 637–638, 649–651
 - PRIMARY KEY / FOREIGN KEY relationships, 638–639
 - PRIMARY KEY / FOREIGN KEY relationships, retrieving, 664
 - privileges, considerations when revoking, 925
 - queries, retrieving information about using, 664–666
 - RELY flags, 664
 - row condition, retrieving, 664
 - state, 642–647
 - state, changing, 657, 659–663
 - status, retrieving, 664
 - tables, adding to existing, 656–657
 - tables, defining during creation, 654–656
 - tables, dropping constraints before truncation, 542
 - tables, dropping from existing, 657
 - tables, FOREIGN KEY violations caused by dropping, 651–652
 - tables, locking caused by, 652–654, 659
 - tables, modifying on existing, 657–658
 - tables, relationship enforcing using FOREIGN KEY, 638–639
 - tables, rendering read-only using, 644–645, 660
 - tables, retrieving information about, 664, 665
 - tablespace drops causing FOREIGN KEY violations, 652
 - transportable tablespace exports, 780–781
 - type, retrieving, 664
 - types, 634–642
 - UNIQUE, 608, 636–637, 649–651

UNIQUE / FOREIGN KEY relationships, 638–639
 UNIQUE / FOREIGN KEY relationships,
 retrieving, 664
 USING INDEX option, 656, 659
 validation by database engine, 634
 validation information, retrieving, 664
 validation states, 642–647, 659–663
 value ranges, restricting using CHECK constraints,
 640–642
 views, 663–666

integrity, data. *See also* integrity constraints
 application code, using, 631–632
 data dictionary, rule storage in, 196
 method, choosing, 631–633
 precision, numeric, 496
 scale, numeric, 496
 table analysis for, 527
 triggers, using, 214–215, 632, 633

INTERNAL privileged account
 auditing, automatic, 933
 creating, 157
 password, 50, 157, 169

Internet resources, 1127

INTPWD ORADIM Utility parameter, 169

IOT. *See* tables, index-organized (IOT)

J

JA16EUC character set, 1029
 JA16EUCFIXED character set, 1028–1029, 1030
 Japanese Extended UNIX Code (JEUC) character set,
 1028, 1030

Java program units, calling, 209

JAVA_POOL_SIZE parameter, 13

Job Information screen, 708, 710

job queues, importing/exporting, 740, 743

Job Scheduling Service, 64

JServer, 154

K

kernel resource limits, managing using profiles,
 804, 805

Knowledge Base for Oracle Administration
 (on the CD), 1069

L

language settings. *See* character set; National
 Language Support (NLS)

large object (LOB) datatypes, 229, 397, 498–499

large object (LOB) segments, 397

LARGE_POOL_SIZE initialization parameter, 97, 162

LCK0. *See* lock process (LCK0)

leaf blocks. *See* indexes, leaf blocks

Least Recently Used (LRU) list, 511

LGWR. *See* Logwriter (LGWR)

list separator character setting, 1036

LISTENER.ORA file, 149

load operations, external data. *See also* load
 operations from existing tables; SQL*Loader
 backup after, 715
 column values, generating when data absent, 699
 column values, padding, 699
 column values, trimming, 699
 constraints behavior during, 699, 712, 714, 716
 control file instructions, 697, 698–700, 714
 conventional loads, 687, 696, 705
 conventional loads, direct-path compared, 711–714
 data manipulations during, 697, 699
 data replacement by, 696
 data selection criteria, applying, 699
 database buffer cache, using, 712
 datafiles, writing directly to, 687, 711
 datafiles written to, specifying, 698, 703, 717
 described, 687
 direct loads, 687, 696, 705
 direct loads, conventional compared, 711–714
 direct loads, parallel, 713–714
 error defaults, 704
 extents management, 713, 714, 716
 fields, fixed length, 696
 fields, mapping, 699
 fields, variable length, 696
 file formats supported, 687, 696
 Hash Clusters, slowed by, 521
 indexes unusable after, 716
 INSERT trigger firing, 712, 714
 instance failure, 716
 Load Wizard, using, 706–711
 logging, 698, 700–702, 712
 Oracle databases, external, 687
 parallel, 705, 712, 713–714
 record terminator characters, 696
 records, loading selectively, 699
 records, presorting, 715
 records rejected, sending to discard file, 698, 703
 records, skipping, 699, 704, 715
 records, specifying maximum bad, 704, 717
 records, specifying number to load, 704
 rollback segment management, 693, 704, 715
 rows, memory setting for arrays of, 704
 rows, overwriting, 699
 rows rejected, sending to bad file, 698, 703

Continued

- load operations, external data (*continued*)
 - rows, specifying number loaded before
 - COMMIT, 704
 - space management, 713, 714, 716
 - statistics, 701, 715
 - tables, appending to, 699
 - tables, clustered, 712
 - tables, Data Manipulation Language (DML)
 - operations on target, 712
 - tables, overwriting, 699
 - type, specifying, 705, 708
 - types, 687
- load operations from existing tables. *See also* load operations, external data
 - backup after, 690
 - Data Manipulation Language (DML) statements, parallel, 691–694
 - database buffer cache, bypassing, 687, 689
 - database buffer cache, using, 689
 - datafiles, writing directly to, 687, 689
 - described, 687
 - extents management, 689, 691, 695
 - INSERT APPEND operations, 511–512, 519, 690
 - INSERT INTO ... SELECT statements, 687, 688
 - INSERT operations, ALTER SESSION parameters, 693
 - INSERT operations, APPEND hints, 690, 693
 - INSERT operations, clustered tables, 694
 - INSERT operations, constraint behavior during, 688, 694, 695
 - INSERT operations, conventional, 687, 689
 - INSERT operations, direct-load, 687, 688–696
 - INSERT operations, index behavior during, 581, 598, 688, 695
 - INSERT operations, instance failure during, 695
 - INSERT operations, logging options, 12, 511–512, 690–691, 695
 - INSERT operations, one row at a time, 688
 - INSERT operations, parallel direct-load, 691–694
 - INSERT operations, relation to free list, 407, 689
 - INSERT operations, simultaneous, 407
 - INSERT operations, space allocation in, 689, 691, 695
 - INSERT operations, tables with LOB columns, 694
 - INSERT operations, tables with object columns, 694
 - INSERT operations, trigger firing, 214, 215, 219, 694
 - INSERT operations using subqueries, 688
 - instance failure, 695
 - optimizer hints, 689, 690
 - speed, 689, 695
- LOAD SQL*Loader parameter, 704
- Load Wizard, 706–711
- LOB datatypes. *See* large object (LOB) datatypes
- LOB segments. *See* large object (LOB) segments
- lock process (LCK0), 18
- .log files, 698, 700
- LOG parameter
 - Export utility, 747
 - Import utility, 759
 - SQL*Loader, 703
- log sequence numbers (LSNs), 277, 282, 288, 294
- log switches, 282–284
- LOG_ARCHIVE_DEST initialization parameter, 22, 99, 289
- LOG_ARCHIVE_FORMAT initialization parameter, 99
- LOG_ARCHIVE_MAX_PROCESSES initialization parameter, 17
- LOG_ARCHIVE_START initialization parameter, 22, 99, 289
- LOG_BUFFER initialization parameter, 98, 162
- LOG_CHECKPOINT_INTERVAL initialization parameter, 16, 98–99, 284, 286
- LOG_CHECKPOINT_TIMEOUT initialization parameter, 16, 98, 284–285, 286
- LOGGING parameter, 508, 511–512, 531
- LOGICAL_READS_PER_CALL resource management setting, 811, 818
- LOGICAL_READS_PER_SESSION resource management setting, 811
- login attempts, setting maximum, 803, 806
- LogMiner utility, 235, 302–306
- logon credentials, 57
- logs. *See also* archiving; redo log files
 - ALERT file, 100, 122–125, 301
 - analyzing using LogMiner, 302–306
 - archiving, 6, 17, 21–22
 - buffer size, 98
 - control files, log file information stored in, 257
 - dump settings, 100, 163
 - file groups, returning, 263
 - INSERT operations, 12, 511–512, 690–691, 695
 - load operations, external data, 698, 700–702, 712
 - load operations from existing tables, 12, 511–512, 690–691, 695
 - names, 99
 - ORADIM.LOG, 170
 - table logging parameters, 508, 511–512, 531, 547
 - trace files, 100–101, 122–125, 163, 260–261
- Logwriter (LGWR)
 - datafiles, writing to, 15
 - described, 16
 - redo log buffer flushing, 12
 - redo log file, writing to, 16, 278, 279, 282, 300–301
- LONG datatype, 493, 498–499, 1029

LONG RAW datatype, 494, 498
 LRU list. *See* Least Recently Used (LRU) list
 LSNs. *See* log sequence numbers (LSNs)

M

masks, number format, 1043–1044
 MAX_DUMP_FILE_SIZE initialization parameter, 100
 MAXEXTENTS parameter
 DB_BLOCK_SIZE, relation to, 400
 defaults, 341, 400
 described, 341
 precedence, 398
 rollback segments, 441, 468
 setting, 441
 tables, 509, 531
 UNLIMITED, 341, 441
 values, retrieving, 444
 MAX_ROLLBACK_SEGMENTS initialization parameter, 454
 MAXSIZE parameter, 339–340, 350–351
 MAXTRANS parameter
 default, 405
 described, 405
 indexes, 598, 606, 610
 retrieving, 547
 tables, 508, 531, 547
 value, maximum, 405
 MAXUSERS ORADIM Utility parameter, 169
 memory management. *See* storage management
 migration, row, 525, 527–530, 548
 MINEXTENTS parameter
 default, 341, 400
 described, 341
 indexes, 598, 610
 precedence, 398
 retrieving, 444
 rollback segments, 438, 440, 459
 tables, 509, 531
 MOUNT state, 93, 104, 114
 moving data between databases. *See* databases,
 moving data between
 MTS connections. *See* multithreaded server (MTS)
 environment
 MTS_DISPATCHERS initialization parameter, 17
 MTS_MAX_DISPATCHERS initialization parameter, 17
 MTS_MAX_SERVERS initialization parameter, 17
 MTS_SERVERS initialization parameter, 17
 multiplexing
 control files, 145, 259–260
 enabling, 162
 redo log files, 146, 176, 276, 278, 281

Multipurpose environment, 152
 multithreaded server (MTS) environment, 17, 154,
 161–162, 804, 811

N

National Language Support (NLS). *See also* character
 sets
 ALTER SESSION command, changing using, 1032,
 1036–1038
 calendar format, 1026, 1034
 character set, changing, 1030–1031
 character set, choosing, 1026–1031
 client application character set setting, 1036
 client settings, translation to/from server set,
 1025, 1026
 comparison method, 1034
 country-specific values, 1033
 currency display, 1025, 1026, 1033–1034, 1036
 database creation, setting at, 165
 date display, 1025, 1026, 1033, 1034, 1035
 DBMS_SESSION.SET_NLS package, changing using,
 229, 1037–1038
 defaults, retrieving current, 1046
 described, 1025–1026
 environment variables used in configuring, 168,
 177, 1035–1036, 1038
 Export utility use of NLS settings, 1045
 exporting data, considerations in, 775–776
 features, 1025–1026
 format masks, 1034, 1036, 1037, 1043–1044
 functions, using NLS parameters in, 1042–1044
 Import utility use of NLS settings, 1045
 importing data, considerations in, 775–776
 INIT.ORA parameters, configuring using, 1032–1035
 instance level settings, retrieving, 1046–1047, 1049
 language settings, 1032–1035
 list separator character setting, 1036
 location of NLS files, default, 168
 masking characters, 1034, 1036, 1037, 1043–1044
 message display parameters, 1033
 number format, 1026, 1034, 1036
 parameters, retrieving current, 1046
 Server Manager Line Mode retrieving info about
 using, 1047
 sessions, changing settings for, 1036–1038
 sort settings, 1026, 1033, 1038–1041
 SQL*Loader utility use of NLS settings, 1045
 SQL*Plus retrieving info about using, 1047
 time display, 1025, 1026, 1038
 timestamp display, 1038

Continued

National Language Support (NLS) (*continued*)

- Unix systems, 1036
- users, changing by, 1025
- views, 1045–1049
- NCHAR datatype, 494
- NCLOB datatype, 229, 397, 494, 498
- Net8 Assistant, 77
- network authentication, 850, 870
- NEW ORADIM Utility parameter, 169
- :new trigger prefix, 217
- NEXT parameter
 - datafile autoextension, in, 339–340, 350–351
 - default, 341, 400
 - described, 341
 - indexes, 598
 - precedence, 398
 - retrieving, 444, 547, 550
 - rollback segments, 441
 - tables, 509, 531
- NLS. *See* National Language Support (NLS)
- NLS_CALENDAR initialization parameter, 1034, 1042
- NLS_CHARACTERSET initialization parameter, 1048, 1049
- NLS_COMP initialization parameter, 1034
- NLS_CREDIT environment variable, 1036
- NLS_CURRENCY initialization parameter, 1033, 1034, 1042
- NLS_DATABASE_PARAMETERS view, 1046
- NLS_DATE_FORMAT initialization parameter, 1033, 1034
- NLS_DATE_LANGUAGE initialization parameter, 1033, 1042
- NLS_DEBIT environment variable, 1036
- NLS_DUAL_CURRENCY initialization parameter, 1034
- NLS_INSTANCE_PARAMETERS view, 1046–1047
- NLS_ISO_CURRENCY initialization parameter, 1033, 1034, 1042
- NLS_LANG environment variable, 1035–1036, 1045
- NLS_LANGUAGE initialization parameter, 1033, 1049
- NLS_LIST_SEPARATOR environment variable, 1036
- NLS_LOWER initialization parameter, 1042
- NLS_MONETARY_CHARACTERS environment variable, 1036
- NLS_NCHAR environment variable, 1036
- NLS_NCHAR_CHARACTERSET initialization parameter, 1048, 1049
- NLS_NUMERIC_CHARACTERS initialization parameter, 1033, 1034, 1042
- NLS_SESSION_PARAMETERS view, 1046
- NLSSORT function, 1040–1042, 1043

- NLS_SORT initialization parameter, 1033, 1038–1041, 1049
- NLS_TERRITORY initialization parameter, 1033, 1049
- NLS_TIME_FORMAT environment variable, 1038
- NLS_TIMESTAMP_FORMAT environment variable, 1038
- NLS_TIMESTAMP_TZ_FORMAT environment variable, 1038
- NLS_TIME_TZ_FORMAT environment variable, 1038
- NLS_UPPER initialization parameter, 1042
- NOARCHIVELOG mode, 21, 176, 256, 287–289
- NOAUDIT command, 894
- NOCACHE parameter, 508, 511, 531
- NO_DATA_FOUND exceptions, 213
- node discovery, 63, 73
- NOLOGGING parameter, 508, 511–512, 531, 598, 690
- NOMOUNT state, 93, 103, 113
- NORMAL shutdown mode, 93, 109, 111
- NOSORT parameter, 598, 600
- NOT NULL constraints, 587, 634–636, 639, 659
- nulls
 - conditional, 641
 - constraints controlling, 587, 634–636, 641
 - defined, 634
 - indexes, in, 587, 636
 - mathematical expressions, in, 634–636
 - number of, retrieving, 553
- NUMBER datatype, 493, 496
- NVARCHAR2 datatype, 493

O

- O7_DICTIONARY_ACCESSIBILITY initialization parameter, 913
- Object Relational Model, 502
- object-oriented programming languages, interaction with, 502
- objects
 - auditing, 935, 942–945
 - comparing using Change Management Pack, 61
 - Constraints objects, 848
 - creation date, retrieving, 545
 - database objects, 848–849
 - DBA Studio, viewing object information using, 230
 - DBA_OBJECTS view, 231, 545–546, 774
 - dependencies, retrieving, 866–867
 - dependencies, tracking, 232
 - Functions objects, 849
 - Indexes objects, 848
 - listing all available, 204
 - ownership, listing by, 201–203
 - Packages objects, 849

- privileges for creating, 851
- privileges, granting using GRANT command, 914–916
- privileges, granting using Oracle Enterprise Manager, 916–924
- privileges listed, 914
- privileges, revoking, 925–928
- privileges, views, 929–932
- schemas, 847–848
- secondary, 546
- Sequences objects, 848
- status, retrieving, 231, 546
- Stored Procedures objects, 848
- Synonyms objects, 848
- Table objects, 848
- temporary, 336–338
- timestamp, retrieving, 546
- Triggers objects, 848
- type, retrieving, 545
- User-Defined Datatypes objects, 849
- views, 204, 230–232, 545–546, 774
- Views objects, 848
- OBJ_INFO.SQL file, 768–770
- OEM. *See* Oracle Enterprise Manager (OEM)
- OEM_MONITOR role, 981
- OFA. *See* Optimal Flexible Architecture (OFA)
- OID. *See* Oracle Internet Directory (OID)
- :old trigger prefix, 217
- OLTP environment. *See* Online Transaction Processing (OLTP) environment
- OMS. *See* Oracle Management Server (OMS)
- ON DATABASE keyword, 222, 223
- ON DELETE CASCADE constraint option, 656, 664
- Online Transaction Processing (OLTP) environment, 152
- OPEN state, 93, 104
- operating system
 - authentication, operating system, 45–49, 101, 144, 849–850, 860–863
 - blocks, operating system, 328, 329, 391
 - database creation, preparing for, 142–143, 167–170
 - username, mapping to, 861
- operators, importing/exporting, 740, 742
- OPS\$ user ID prefix, 861
- Optimal Flexible Architecture (OFA), 42–43, 147–149
- OPTIMAL parameter, 436–438, 441, 460
- optimizer hints, 689, 690
- OPTIMIZER_MODE initialization parameter, 21
- ORA-01432 errors, 234, 236
- Oracle Administration Assistant snap-in, 53
- Oracle Advanced Queuing, 235, 981
- Oracle Certified Professional Program Candidate Guide*, 1104
- Oracle Corporation Web sites, 1104, 1127
- Oracle Data Migration Assistant, 77–78
- Oracle Database Assistant, 861
- Oracle Database Configuration Assistant
 - accessing, 150
 - Custom configuration, 152
 - described, 76
 - startup screen, 150
 - tasks accomplished by, 149
 - Typical configuration, 150–152
- Oracle Enterprise Manager (OEM). *See also* Oracle Storage Manager
 - Application Management Pack, 61
 - Change Management Pack, 61
 - client component, 62–63
 - Configuration Assistant, 65–69
 - Console, 62, 70–76
 - Database Management Pack, 60
 - described, 60
 - Diagnostic Pack, 61
 - Event Management Service, 64
 - Export utility, using through, 753–755
 - Import utility, using through, 756, 770–772
 - index creation using, 601–604
 - index information, retrieving using, 615–616
 - installation, 60
 - Instance Manager, 60, 63, 102–103
 - Job Scheduling Service, 64
 - language written in, 60
 - node discovery service, 63, 73
 - Oracle Management Server (OMS) component, 58, 63–69, 70–71
 - privileges created by, 856
 - privileges, granting using, 902–906, 916–924
 - privileges, revoking using, 909–910, 927–928
 - profiles, assigning using, 858
 - profiles, creating using, 815–818
 - profiles, dropping using, 822–824
 - profiles, modifying using, 820–821
 - redo log file maintenance using, 300
 - repository creation, 65–69
 - roles, creating using, 984–986
 - roles, dropping using, 998–999
 - roles, establishing default using, 995–996
 - roles, granting to users using, 984–986

Continued

- Oracle Enterprise Manager (OEM) (*continued*)
 - roles, granting/revoking privileges to using, 989–990
 - roles, modifying using, 991
 - roles, revoking from users using, 984–986
 - rollback segments, bringing online using, 455
 - Schema Manager, 61, 63, 514, 921–924
 - Security Manager, 60, 63, 905
 - Security Service, 64
 - SQL*Plus Worksheet, using with, 58, 63, 357–358
 - SYSTEM datafiles, moving using, 356–358
 - Tuning Pack, 61
 - users, creating using, 857–860
 - users, dropping using, 868–869
 - users, modifying using, 864–865
 - Oracle Expert utility, 61
 - Oracle Home ⇄ Database Administration ⇄ Database Configuration Assistant, 150
 - Oracle Intelligent Agent, 69, 847, 981
 - Oracle Internet Directory (OID), 850
 - Oracle Magazine Web site, 1127
 - Oracle Management Server (OMS), 58, 63–71. *See also* Oracle Enterprise Manager (OEM)
 - Oracle Parallel Server, 18, 432
 - Oracle Storage Manager
 - Create Tablespace screen, 344
 - datafiles, moving using, 354
 - datafiles, setup using, 344
 - rollback segments, creating using, 442–443
 - tablespaces, changing storage settings using, 349
 - tablespaces, creating using, 343–345
 - tablespaces, dropping using, 359
 - tablespaces, making read-only using, 348
 - tablespaces, making read-write using, 348
 - tablespaces, resizing using, 351–352
 - tablespaces, taking offline using, 347
 - Oracle Technology Network, 1066, 1112
 - Oracle Trace utility, 61
 - Oracle Universal Installer, 37–42. *See also* installation
 - ORACLE_BASE environment variable, 148, 168
 - ORACLE_HOME environment variable, 50, 148, 168
 - ORACLE_SID environment variable, 50, 168, 233, 860
 - ORA_DBA group, 48
 - ORADIM utility, 59–60, 106–107, 169–170, 173
 - ORADIM.LOG file, 170
 - ORA_NLS33 environment variable, 168, 177
 - ORA_OPER group, 48
 - ORAPWD utility, 21, 49–52, 144, 173
 - ORA_PWFILE Registry entry, 52
 - ORA_SID_AUTOSTART Registry entry, 107
 - ORA_SID_DB group, 48
 - ORA_SID_DBA group, 48
 - ORA_SID_OPER group, 48
 - OS_AUTHENT_PREFIX initialization parameter, 101, 861–862
 - OSDBA user/role
 - auditing, automatic, 933
 - groups, granting to, 48
 - privileges, 46–47, 48, 144
 - OSOPER user/role
 - auditing, automatic, 933
 - groups, granting to, 48
 - privileges, 46–47
 - OS_ROLES initialization parameter, 982
 - OUTLN user, 847
 - OWNER Export utility parameter, 747
- ## P
- packages. *See also specific packages*
 - alert support, 235
 - body, creating, 226–228
 - data dictionary packages, 228–230
 - described, 224–225
 - importing/exporting, 740, 742
 - objects, as, 849
 - performance improvement using, 224–225
 - pipe support, 235
 - programs in, calling, 228
 - programs, private, 225, 226
 - programs, public, 225
 - roles needed for execution, 979
 - specification, creating, 225–226
 - user-defined, 39
 - variables, 225, 226, 228
 - wrapping, 238
 - Parallel Server, 18, 432
 - PARALLEL SQL*Loader parameter, 705, 713
 - parallelism, 548, 610, 691–694, 712
 - PARFILE parameter
 - Export utility, 748, 751, 775
 - Import utility, 759, 775
 - SQL*Loader, 705
 - PARTITION BY RANGE keyword, 394
 - partitioning
 - composite, 395
 - hash, 395
 - indexes, 585, 611
 - names of partitions, retrieving, 549
 - range, 394–395

- segments, partition, 393–396
- sub-partitioning, 395–396
- tables, 489–490, 548, 585
- tablespace considerations, 335
- PASSWORD EXPIRE PROFILE CREATE USER
 - parameter, 855
- password file
 - authentication using, 46, 49–52, 144
 - creating using ORADIM utility, 173
 - creating using ORAPWD utility, 21, 50–52, 144, 173
 - described, 21
 - location, 50, 51–52, 144
 - name, 144
 - privileges to, 50
 - Registry entry, 52
 - setup, 101
 - system privileges, relation to, 21, 50, 899
 - UNIX systems, 50–51, 144, 173
 - users, adding, 21
 - Windows NT/2000 systems, 51–52, 144, 173
- PASSWORD_GRACE_TIME profile setting, 806–807, 814
- PASSWORD_LIFE_TIME profile setting, 806
- PASSWORD_LOCK_TIME profile setting, 806, 814
- PASSWORD_REUSE_MAX profile setting, 807
- PASSWORD_REUSE_TIME profile setting, 807
- passwords. *See also* authentication; security
 - account lockout on invalid, 805, 806
 - assigning, 852
 - case sensitivity, 43
 - changes, forcing, 805
 - changes, setting grace period, 806–807, 814
 - complexity rules, 805, 808
 - expiration settings, 805, 806, 855, 856, 858
 - functions, using to enforce complexity rules, 805, 808–809
 - history, importing/exporting, 743
 - history option, setting, 805
 - INTERNAL account, 50, 157, 169
 - length rules, 803
 - profiles, management using, 805–809, 817–818
 - reusing old, controlling, 805, 807
 - roles, passwording, 976, 982, 984, 990–991, 994
 - scripts, managing using, 808–809
 - security, role in, 802
 - SYS account, 43, 178
 - SYSTEM account, 43, 178
 - views, 869
- PASSWORD_VERIFY_FUNCTION profile setting, 808
- PATH environment variable, 168
- PCTFREE parameter
 - described, 405–406
 - indexes, 588, 597–598, 600, 606, 607
 - tables, 508, 525–526, 528–529, 531, 532
 - viewing, 546
- PCTINCREASE parameter
 - default, 341, 400
 - described, 341, 398
 - indexes, 598, 610
 - precedence, 398
 - retrieving, 547, 550
 - rollback segments, 439
 - tables, 509, 523, 531
 - values, retrieving, 444
- PCTUSED parameter
 - described, 405–406
 - indexes, 405, 598, 605
 - tables, 405, 508, 526–527, 531, 532
 - viewing, 546
- permissions. *See* privileges
- PFILE ORADIM Utility parameter, 170
- PGA. *See* Process Global Area (PGA)
- PLB files, 238
- PL/SQL
 - functions, 213–214
 - program units, 209, 211
 - support, setting up, 235
- PMON. *See* process monitor (PMON)
- Prep Exam (on the CD), 1068
- PRIMARY KEY constraints, 490, 584, 608, 637–639, 649–651
- PRIVATE_SGA resource management setting, 811
- privileged users. *See* users, privileged
- privileges. *See also* profiles; roles; security; *specific privileges and roles*
 - analysis, for, 897
 - auditing operations on, 940
 - auditing options present, viewing, 948
 - auditing system privileges, 940, 941–942
 - auditing usage, 933
 - backups, for, 260
 - column privileges, 542, 915
 - constraint considerations when revoking, 925
 - data dictionary objects access, restricting
 - using, 913
 - database administrator, 43–44, 200
 - databases, for creating, 44, 46, 144, 179
 - direct, 802

Continued

privileges (*continued*)

- export operations, for, 739, 740, 741, 745, 747
- granting system privileges, 900–906
- granting using GRANT command, 893, 914–916
- granting using Oracle Enterprise Manager (OEM), 902–906, 916–924
- import operations, for, 756, 772
- importing permissions, 739, 740
- indexes, for altering, 604
- indexes, for creating, 597, 897
- instances, for starting/stopping, 46, 104–105
- instances, system privileges for, 894–895
- log switches, for controlling, 284
- managing, tasks involved, 893–894
- object privileges, granting using GRANT command, 914–916
- object privileges, granting using Oracle Enterprise Manager, 916–924
- object privileges, listed, 914
- object privileges, revoking, 925–928
- object privileges, views, 929–932
- objects (segments), for creating, 851
- Oracle Enterprise Manager (OEM), created by, 856
- password file, system privileges relation to, 21, 50, 899
- password file, to, 50
- profiles, for working with, 813
- program units, 211
- quota-related, 897
- revoking, 524, 893, 906–910, 925–928
- roles, assigning to, 900, 987, 989–990
- roles, compared, 976–977
- roles, for granting, 976, 992
- roles, removing from, 989–990
- roles, retrieving privileges granted to, 1001, 1002
- rollback segments, for creating, 438
- sessions, for altering, 117
- sessions, for creating, 772, 856
- sessions, for restricted, 105, 119–120
- sessions, retrieving currently running privileges, 912–913
- system privileges, 894, 898
- system privileges assigned to roles, retrieving, 1001
- system privileges, assigning to roles, 987
- system privileges, auditing, 940, 941–942
- system privileges, database, 894–895
- system privileges, granting, 900–906
- system privileges, instances, 894–895
- system privileges, listing all, 895–897

- system privileges, objects, 895
- system privileges, revoking, 906–910
- systems settings considerations, 895
- systems settings, for changing, 117
- tables, for creating, 507, 772, 897
- tables, for dropping, 542
- tables, for selecting, 200, 907–909
- tables, for truncating, 541
- tablespace-related, 438, 897–898
- trigger-related, 223
- types, 894
- users, allowing to grant, 914–916, 919–920, 923
- users, for altering, 813, 897
- users, for creating, 813, 897
- users, retrieving privileges granted by, 930
- users, retrieving privileges granted to, 900, 910–913, 929–932
- views, 200, 900, 910–913, 929–932
- views, for creating, 895, 907–909
- WITH ADMIN OPTION clause, 901, 907, 977
- WITH GRANT OPTION clause, 914–915, 919–920, 923, 977

procedures, stored

- auditing, 937–938, 939
- declaration section, 212
- described, 211–213
- exception handling, 213
- exporting, 740, 742
- importing, 740, 742
- objects, as, 848
- parameters, input/output, 212
- PL/SQL commands, 196, 212
- tables, on temporary, 517
- variables, 212

process, dispatcher. *See* dispatcher process (Dnnn)

Process Global Area (PGA), 24

process, lock. *See* lock process (LCK0)

process monitor (PMON), 6, 14

process, recoverer. *See* recoverer process (RECO)

process, server. *See* server process

process, user. *See* user process

processes. *See also specific processes*

- background, 6, 8, 13–18
- killing, 109
- number of, maximum, 100, 162

PROCESSES initialization parameter, 100, 162

PROFILE CREATE USER parameter, 855

profiles. *See also* privileges; security

- assigning using ALTER USER command, 804, 815
- assigning using CREATE USER command, 815

- assigning using Oracle Enterprise Manager, 858
 - auditing, 939
 - creating, 804, 813–818
 - DBA Studio, managing using, 823
 - DEFAULT, 804
 - DEFAULT, changing using script, 808–809
 - DEFAULT, inheritance from, 814, 825
 - described, 803–805
 - dropping, 821–824
 - enforcement, 804–805
 - kernel resource limits, 804, 805
 - modifying, 818–821
 - needs analysis, 804
 - Oracle Enterprise Manager (OEM), managing using, 815–818, 820–821, 822–824, 858
 - password management using, 805–809, 817–818
 - privileges needed for working with, 813
 - resource management using, 804–805, 809–813
 - users assigned to, retrieving, 825–826
 - views, 824–826
- program units
- auditing, 944
 - calling, 210
 - compiling, 210
 - data dictionary, storage in, 210
 - dynamic link libraries (DLLs), accessing as, 210
 - external, 210
 - functions, 213–214
 - functions, user-defined, 211
 - introduced, 196
 - Java program units, calling, 209
 - loading, 210
 - packages, 224–228
 - permissions, 211
 - PL/SQL program units, 209, 211
 - privileges, 211
 - procedures, 211–213
 - session parameters, changing in, 229
 - triggers, 214–224
 - types, 209–210
 - UNIX shared libraries, accessing as, 210
- PRVT*.SQL scripts, 238
- PUBLIC role, 900, 906–907
- Q**
- queries
- blocks changed after query starts, 431
 - data dictionaries, in, 198
 - data dictionary views, using in place of, 199
 - export operations, applying in table-level, 748
 - index information, listing using, 614–615
 - on indexes, 581, 582
 - on indexes, bitmap, 593
 - on indexes, function-based, 584, 591–592
 - on indexes, multiple predicate, 593, 594, 596
 - on indexes, referencing columns, 583
 - on indexes, Reverse Key, 590
 - on indexes, Structured Query Language (SQL), 587
 - Parallel Query support, setting up, 235
 - read consistency errors, 464
 - roles for data dictionary queries, 979
 - rows, migrated, 528
 - statistics, 21
 - tables, against temporary, 517
 - tables, storing results in temporary, 517
- QUERY Export utility parameter, 748
- QUERY_REWRITE_ENABLED initialization parameter, 584, 592
- QUOTA PROFILE CREATE USER parameter, 855
- QUOTA UNLIMITED privilege, 897
- quotas. *See* tablespaces, quotas
- R**
- RAW datatype, 494, 497
- read consistency errors, 464
- read-only access
- constraints, using, 644–645
 - database mode, 107–108
 - tablespaces, read-only, 331, 347–348
- RECALCULATE _STATISTICS Import utility parameter, 759
- RECO. *See* recoverer process (RECO)
- RECORD parameter
- Export utility, 748
 - Import utility, 759
- RECORDLENGTH parameter
- Export utility, 748
 - Import utility, 759
- RECOVER DATABASE UNTIL privilege, 47
- recoverer process (RECO), 18
- Recovery Manager (RMAN), 257, 981
- RECOVERY_CATALOG_OWNER role, 981
- redo log buffer, 11–12, 16, 162
- redo log files
- analyzing using LogMiner, 302–306
 - ARCHIVELOG mode, 21–22, 161, 176, 289–291
 - archiving destination, 99, 289, 294

Continued

- redo log files (*continued*)
 - backup strategy, planning around, 278
 - clearing, 300
 - control files, information stored in, 256, 257, 258, 263
 - creation with database, 279–280
 - described, 7, 20, 275–277
 - groups, 145
 - groups, adding, 295–296
 - groups, creating, 158–159, 175–176, 279–281
 - groups, dropping, 296–297
 - groups, initial, 279–280
 - groups, maximum number of, 280
 - groups, minimum number of, 278
 - groups, optimal file size, 278
 - groups, optimal number of, 278, 280–281
 - groups, returning, 263
 - index logging options, 598, 599, 603
 - information about, returning, 263, 291–293
 - information recorded in, 275
 - INSERT operations, 12, 511–512, 690–691, 695
 - listing, 263
 - location, 42, 145–146
 - log sequence numbers (LSNs), 277, 282, 288, 294
 - log switch number, 256
 - Logwriter (LGWR) writing process, 16, 278, 279, 282, 300–301
 - members, 278–279
 - members, adding, 296
 - members, dropping, 298
 - members, invalid, 301
 - members, maximum/minimum number per group, 280
 - members, unavailable, 301
 - members, viewing status, 298
 - moving, 299
 - multiplexing (multiple copies), 146, 176, 276, 278, 281
 - NOARCHIVELOG mode, 21, 176, 256, 287–289
 - online, 281–287, 295–300
 - Oracle Enterprise Manager (OEM), maintaining using, 300
 - recovery, role in, 275–276
 - Redundant Arrays of Independent Disks (RAID) considerations, 147
 - renaming, 299
 - roles needed to access, 979
 - size, recommended, 284
 - state, returning, 263
 - switches, 282–284
 - switches, forcing, 297
 - system change numbers (SCNs), returning, 263
 - table logging parameters, 508, 511–512
 - tables, on temporary, 517
 - troubleshooting, 300–301
 - views, 291–293
 - write process, 145–146
 - writing pattern, circular, 276, 279, 282
- referential integrity, 638
- Registry, editing for automatic startup, 106–107
- Relative File Number, ROWID, 501, 502
- RELY flags, 664
- REMOTE_LOGIN_PASSWORDFILE initialization parameter, 47, 51, 52, 101, 899
- REMOTE_OS_AUTHENT initialization parameter, 861, 862–863
- replication support, setting up, 235
- repositories, Oracle Enterprise Manager (OEM), 63, 64, 65–69
- RESOURCE role, 898, 980–981
- resource usage, tracking, 101
- RESOURCE_LIMIT initialization parameter, 101, 804–805, 809
- RESTRICTED mode, 347
- RESTRICTED SESSION privilege, 105, 119–120
- REUSE keyword, 340, 351
- RevealNet's Knowledge Base for Oracle Administration (on the CD), 1069
- REVOKE command
 - privileges, 893, 906–907, 925–927
 - roles, 989, 992
- RMAN. *See* Recovery Manager (RMAN)
- ROLE_ROLE_PRIVS view, 1000–1001
- roles. *See also* privileges; *specific roles*
 - auditing, 933, 940
 - authentication by application, 976, 984
 - authentication by operating system, 977, 982–983, 984
 - authentication by Oracle Security Server, 983–984
 - backward compatibility, 980–981
 - creating, 982–986, 1004
 - data dictionary object deletion, for, 979
 - data dictionary queries, for, 979
 - database administrators, 978
 - database audit trail deletion, for, 979
 - database export, for, 979
 - database import, for, 756, 772, 980
 - database, retrieving roles available in, 999–1000
 - DBMS_PIPE, for, 979
 - DBMS_RLS access, for, 979

- DBMS_SESSION.SET_ROLE procedure, disabling using, 996–997
- default user roles, establishing, 983, 994–996
- described, 975–976
- disabling, 994, 996–997
- dropping, 998–999
- enabling, 994, 996–997
- enabling selectively, 977
- enabling through applications, 982
- export operations, for, 740, 741, 745, 747, 979
- exporting, 742
- function execution, for, 979
- granting by users, 976, 992
- granting multiple to same user, 976
- granting, privileges needed, 976, 992
- granting to operating system groups, 982–983
- granting to other roles, 987, 991–992
- granting to users, 976, 991–993
- granting using GRANT command, 987, 991–992
- groups, granting to, 982–983
- IDENTIFIED EXTERNALLY option, 982–983
- IDENTIFIED GLOBALLY option, 983–984, 987
- import operations, for, 756, 772, 980
- importing, 742
- log file maintenance, for, 979
- messages between sessions, for sending, 979
- modifying, 990–991
- naming, 976, 983
- object privileges, assigning to, 987
- object privileges granted, retrieving, 1002
- object privileges, removing from, 989
- objects, as, 976
- Oracle Advanced Queuing, for, 981
- Oracle Enterprise Manager (OEM), managing using, 984–986, 989–990, 991, 995–996, 998–999
- Oracle Intelligent Agent, 981
- ownership, 992
- package execution, for, 979
- passwording, 976, 982, 984, 990–991, 994
- passwords, determining if required, 999–1000
- PL/SQL blocks, enabling/disabling from, 997
- pre-defined, 978–981
- privileges assigned directly compared, 976–977
- privileges, assigning to, 900, 987, 989–990
- privileges needed to grant, 976, 992
- privileges, removing from, 989–990
- privileges, role, 43, 803
- procedure execution, for, 979
- procedures, managing from within, 997
- queue privileges, for, 981
- Recovery Manager (RMAN), 981
- redo log file access, for, 979
- revoking, 992–993
- revoking, cascade effect, 977
- roles granted to other roles, retrieving, 1000–1001
- roles, granting to other roles, 987, 991–992
- sessions, retrieving roles enabled for, 1002–1003
- speed advantages, 977
- system privileges, assigning to, 987
- system privileges granted, retrieving, 1001
- task roles, 1003, 1004
- user roles, 1003–1004
- users, granting by, 976, 992
- users, granting multiple to same, 976
- users, granting to, 976, 991–993
- users, retrieving roles enabled, 1002–1003
- users, retrieving roles granted to, 1000
- views, 999–1003
- WindowsNT/2000 environments, 983
- WITH ADMIN OPTION, 977, 983, 987, 992
- WITH ADMIN OPTION, retrieving roles granted with, 1000
- ROLE_SYS_PRIVS view, 1001
- ROLE_TAB_PRIVS view, 1002
- rollback segments
 - ALERT log entries, 124
 - auditing, 940
 - available, 428
 - batch processes, 468
 - block usage, 429, 433–434
 - blocking sessions, 464–466
 - creating, 438–443
 - data warehousing environments, 468
 - datafiles location, 146
 - DEFERRED, 433
 - DELETE operations, information recorded in, 429, 467
 - described, 396, 427
 - dropping, 462–463
 - exporting rollback segment definitions, 742
 - extension, 436
 - extension, dynamic, 439, 441, 467
 - extents, deallocation, 437, 460–462
 - extents, maximum setting, 436, 439, 441, 463
 - extents, minimum possible, 460

Continued

- rollback segments (*continued*)
 - extents, minimum setting, 400, 436, 438, 440, 467
 - extents, retrieving information about, 444, 445, 448–450
 - extents usage, 429, 434–436
 - file ID, retrieving, 444
 - fragmentation, 440
 - growth, 435–436, 439–440
 - headers, 427, 428, 433, 444, 467
 - ID number, retrieving, 444, 445
 - importing rollback segment definitions, 742
 - INITIAL parameter, 441
 - INSERT operations, information recorded in, 429
 - instance, retrieving, 444
 - listing all, 444
 - load operations, during, 693, 704, 715
 - location, 440–441
 - MAXEXTENTS parameter, 441
 - MINEXTENTS parameter, 438
 - name, retrieving, 443, 445
 - NEXT parameter, 441
 - offline, taking, 455–459, 466
 - online, bringing, 432, 453–455
 - online, maximum number, 454
 - OPTIMAL parameter, 436–438, 441, 460
 - ownership, retrieving, 443
 - PCTINCREASE parameter, 439
 - PENDING OFFLINE status, 457
 - planning, 467–468
 - private, 99, 432
 - privileges for creating, 438
 - public, 432
 - purposes of, 26
 - read consistency errors, 464
 - read consistency, role in, 429, 431
 - requesting, 433
 - shrinking, 436–438, 460–462
 - sizing, 439–440, 441, 467
 - Snapshot Too Old errors, 431, 437, 464
 - space deallocation, 437, 460–462
 - space, reclaiming, 436–437
 - space requirements, 427, 463–464
 - space usage, 434–436
 - statistics, retrieving, 445–451
 - status, determining, 444, 456–459
 - storage parameters, changing, 459–460
 - SYSTEM rollback segments, 178, 330, 334, 432
 - tablespaces, errors in taking offline, 466
 - tablespaces, retrieving name, 443
 - tablespaces, rollback, 146, 330, 335, 411, 440
 - throughput, 467
 - ties, 433
 - transaction table, 434
 - transactions, assignment to, 427, 428–429, 430, 433
 - transactions, idle, 465
 - transactions, multiple sharing, 100, 434, 467
 - transactions, recovery role, 431
 - transactions, relation to, 40, 427–428, 433–435
 - transactions, retrieving information about, 451–453, 455–457
 - transactions, simultaneous header access attempts, 467
 - transactions, slots, 404
 - transactions, space requirements, 427, 463–464
 - types, 431–433
 - Undo Segment Number (USN), 445, 446
 - views, 443–453
 - wrap, 436
 - ROLLBACK_SEGMENTS initialization parameter, 99–100, 432, 454
 - ROWID datatype, 494, 499–502
 - ROWID RAW datatype, 494, 499–502
 - ROW_LOCKING parameter, 694
 - rows
 - blocks, storage in, 492, 493
 - chaining, 404, 405, 528–530, 548
 - deleting, 541
 - Directory, 492, 493
 - headers, 492, 493
 - length, retrieving average, 548
 - migration, 525, 527–530, 548
 - number of, retrieving, 547
 - ROWID, 487, 490, 501
 - ROWID, converting between formats, 553–554
 - ROWID, data object number, 501
 - ROWID datatype, 494, 499–502
 - ROWID encoding scheme, 501
 - ROWID, indexes, 582, 586, 594, 596
 - ROWID information, retrieving, 554
 - space, free, 493
 - ROWS parameter
 - Export utility, 748, 749
 - Import utility, 759, 761
 - SQL*Loader, 704, 715
 - SIDrun1.sql file, 165
 - SIDrun.sql file, 165
- ## S
- Schedule screen, 708, 710
 - Schema Manager, 61, 63, 514, 921–924

- schemas, 847–848
- SCN. *See* System Change Number (SCN)
- scripts. *See also specific scripts*
 - CAT*.SQL scripts, 238
 - on CD-ROM with this book, 1112–1114
 - CREATE DATABASE, to be run after, 232
 - data dictionary scripts, 232–238
 - database creation scripts, 164, 177
 - password management scripts, 808–809
 - SQL*Plus, running from, 57
- security. *See also* passwords; privileges; profiles
 - account lockout at user creation, 853, 855, 856
 - account lockout on invalid passwords, 805, 806
 - account lockout, viewing status, 826–827
 - Authentication Mechanism, 801
 - connection time, setting maximum per session, 810, 814
 - CPU time, setting maximum per session, 809
 - CPU time, setting maximum per Structured Query Language (SQL) statement, 810, 819
 - domain elements, 801–803
 - inactivity timeout, setting, 810
 - input/output (IO) activity, setting maximum per session, 803, 804
 - login attempts, setting maximum, 803, 806
 - memory use in Multi-Threaded Server (MTS)
 - environment, setting maximum, 804, 811
 - overview, 801–803
 - policy creation, 801–803
 - procedures, role management from within, 997
 - reads per call, setting maximum, 811, 818–819
 - resource usage limits, setting, 802, 804–805, 809–811
 - resource usage limits, setting using parameter composite, 811–813
 - sessions, setting maximum simultaneous, 810
 - tablespace quotas, 801–802
 - tablespaces, default, 802
 - tablespaces, temporary, 802
 - user security considerations, 852–853
- Security Manager, 60, 63, 905
- Security Service, 64
- segments. *See also* rollback segments
 - backup considerations, 411
 - blocks, returning number of unused, 229
 - bootstrap segments, 397–398
 - cluster segments, 393
 - defaults setup during tablespace creation, 340–341
 - described, 331–332
 - extents, automatic allocation, 401–402
 - extents, manual allocation, 401–402
 - extents, maximum allocated, 341
 - extents, number created with, 341
 - extents, relation to, 328, 333
 - extents, retrieving number in, 550
 - header information, retrieving, 550
 - indexes as, 392, 581
 - large object (LOB) segments, 397
 - lifetime, 411
 - location, 409–412
 - locked, 401
 - name, retrieving, 549
 - ownership, retrieving, 408, 549
 - partitions, 393–396
 - ROWID Data Object Number, 501
 - size, retrieving, 550
 - sizing, 398, 411
 - sort segments, 396
 - space allocation tracking by data dictionary, 196
 - storage clause specification, segment level, 398
 - storage structure, place in, 331–332, 391
 - table segments, 392
 - tables, index-organized, 393
 - tables, nested, 397
 - tablespace, retrieving, 550
 - tablespaces, relation to, 328, 338
 - temporary, 336, 337–338, 342, 396–397
 - type, retrieving, 550
 - types, 392–398
 - types present, listing, 331–332
 - Undo Segment Number (USN), 445, 446
 - users, returning segments owned by, 408
 - views, 362, 407, 549–550
 - views *versus*, 331
- SELECT ANY TABLE privilege, 200, 907–909, 942
- SELECT privilege, 916, 926, 942
- SELECT_CATALOG_ROLE role, 979
- SELFTTEST exam (on the CD), 1068
- sequences, 848, 940, 944
- Server Manager line mode (SVRMGR), 53–56, 173–174, 233, 853, 1047
- server mode
 - Dedicated, 154
 - Shared (multithreaded), 17, 154, 161–162
- server process, 7
- server process, shared (Snnn), 17
- service management, 59–60, 106–107, 169–170
- Services MMC snap-in, 167

- session ID, retrieving, 120–121
- session management
 - access, restricting, 105, 119–120
 - blocked sessions, freeing, 464–466
 - connection time, setting maximum per session, 810, 814
 - connections, auditing, 933, 940
 - CPU time, setting maximum per session, 809
 - diagnostic tools, 61
 - inactivity timeout, setting, 810
 - input/output (IO) activity, setting maximum per session, 803, 804
 - killing sessions, 109, 120–122, 465
 - National Language Support (NLS) settings, changing, 1036–1038
 - roles enabled, retrieving, 1002–1003
 - simultaneous, setting maximum, 810
- session privileges
 - altering sessions, for, 117
 - creating sessions, for, 772, 856
 - restricted sessions, for, 105, 119–120
 - retrieving current, 912–913
- SESSION_PRIVS view, 910, 912–913
- SESSION_ROLES view, 1002–1003
- SESSIONS_PER_USER resource management setting, 810
- SET COLUMNS UNUSED command, 543–544
- SET ROLE command, 992, 994, 996–997
- SET TRANSACTION USE ROLLBACK SEGMENT command, 439
- SET TRANSACTION USER ROLLBACK SEGMENT command, 433
- SGA. *See* System Global Area (SGA)
- Shared Global Area. *See* System Global Area (SGA)
- shared server process (Snnn), 17
- SHARED_POOL_SIZE initialization parameter, 9, 97, 162
- SHOW Import utility parameter, 759, 760, 763–767
- SHOW PARAMETER command, 115–116
- show parameter NLS command, 1034, 1047
- SHUTDOWN command, 108–111
- SHUTDOWN privilege, 46
- SHUTDOWN state, 104
- shutting down, 93–94, 103–104, 108–111
- SID. *See* system identifier (SID)
- SID ORADIM Utility parameter, 169
- <SID>alerttablespace.sql file, 165
- SKIP SQL*Loader parameter, 704, 715
- SMON. *See* system monitor (SMON)
- Snapshot Too Old errors, 431, 437, 464, 746, 775
- snapshots, importing/exporting, 740, 743
- SNMPAGENT role, 981
- Snnn. *See* shared server process (Snnn)
- SORT_AREA_SIZE initialization parameter, 98, 330, 337, 599
- sorting
 - index sorting options, 597, 598, 599, 600
 - information about, returning, 362
 - language settings, 1026, 1033, 1038–1041
 - memory parameters, 98
 - National Language Support (NLS) sort settings, 1026, 1033, 1038–1041
 - segments, sort, 396
 - tablespaces used in, 337
 - views related to, 362
- Specify Node screen, 73
- SQL statements. *See* Structured Query Language (SQL) statements
- SQL.BSQ script, 177, 197, 334, 397, 847
- SQL*Loader. *See also* load operations, external data
 - .bad file, 698, 703
 - BAD parameter, 703
 - BINDSIZE parameter, 704, 717
 - command line parameters, 702–705
 - control file, 697, 698–700, 703, 714
 - CONTROL parameter, 703
 - DATA parameter, 703
 - datafiles, 697, 714, 715
 - discard file, 698, 703
 - DIRECT parameter, 705
 - DISCARD parameter, 703
 - DISCARDMAX parameter, 703, 717
 - ERRORS parameter, 704, 717
 - features, 696–697
 - file formats supported, 687, 696
 - FILE parameter, 705, 715
 - interactive mode, 705
 - LOAD parameter, 704
 - log file, 698, 700–702, 703
 - LOG parameter, 703
 - National Language Support (NLS) settings, use of, 1045
 - Oracle Enterprise Manager, invoking using, 705–711
 - PARALLEL parameter, 705, 713
 - parameter file, 697, 705, 714
 - PARFILE parameter, 705
 - ROWS parameter, 704, 715
 - sessions, multiple, 713
 - SKIP parameter, 704, 715
 - troubleshooting, 715–717
 - USERID parameter, 703
- SQL*Plus, 53, 56–58, 154, 862, 1047
- SQL*Plus Worksheet, 58, 63, 357–358
- SQL_TRACE initialization parameter, 100–101

- SRVC ORADIM Utility parameter, 169
- STAGING databases, 776
- Start Menu ⇄ Programs ⇄ Oracle Installation
 - Products ⇄ Universal Installer, 39
- Start ⇄ Programs ⇄ Oracle OEM, 300
- STARTMODE ORADIM Utility parameter, 170
- STARTUP command, 104–108, 173–174
- STARTUP RESTRICT command, 119
- statistics
 - columns, 553
 - import operations, generating during, 759
 - indexes, 606, 611, 613
 - load operations, 701, 715
 - queries, 21
 - rollback segments, 445–451
 - tables, 546, 548, 549
- STATISTICS Export utility parameter, 748
- storage hierarchy, 327–328
- storage management. *See also* blocks; free lists
 - block space utilization parameters, 405–406, 508–509, 522–523, 525–527, 528–529
 - buffers, dirty, 14, 15, 283, 285
 - cache, data dictionary, 9
 - cache, database buffer, 8, 10–11, 14–15, 162
 - cache, indexes, 598
 - cache, library, 9–10
 - cache, redo log buffer, 11–12, 16
 - cache, tables, 508, 509, 511, 531, 548
 - components, logical, 329–333, 391–392
 - components, physical, 328–329, 391
 - DEFAULT STORAGE clause, 337, 341, 399–400, 409
 - DEFAULT STORAGE clause, viewing settings, 407
 - free space, coalescing, 14, 341, 403–404
 - free space, extents, 333, 402
 - free space, retrieving information about, 229, 408, 409
 - indexes, bitmap, 596
 - instance requirements, 143
 - Least Recently Used (LRU) list, 511
 - log buffer size, 98
 - pools, shared, 9, 12–13, 97, 162
 - precedence, 398–400
 - row chaining, 404, 405, 528–530, 548
 - row migration, 525, 527–530, 548
 - segment level, specifying at, 398
 - segment size settings, 398
 - sorting memory parameters, 98
 - table storage parameters, 508–511, 522–523, 530–532
 - tables, 488–491
 - transaction slot free space allocation, 405
- Storage Manager. *See* Oracle Storage Manager
- stored procedures. *See* procedures, stored
- Structured Query Language (SQL) statements. *See also* functions; *specific scripts*
 - auditing execution, 937–941
 - auditing execution, viewing options present, 946–948, 950
 - CPU time, setting maximum, 810, 819
 - National Language Support (NLS) parameters in, 1042–1044
 - number format masks, 1043–1044
 - optimizer hints, 689, 690
 - statement processing, 22–26
 - system resource management using profiles, 810, 811
 - tracing, diagnostic, 100–101
 - transaction processing, 24–26
- SVRMGRL. *See* Server Manager line mode (SVRMGRL)
- Sylvan Prometric testing centers, 1104
- synonyms
 - auditing, 939
 - importing, 740, 742
 - objects, as, 848
- SYS user/role
 - creation with database, 43–44, 847
 - objects created in SYS schema, 44
 - password, 43, 178
 - privileges, 44, 895
- SYS.AUD\$ table, 935
- SYSDATE keyword, 497
- SYSDBA user/role, 21, 46–47, 50, 51, 898–900
 - auditing, automatic, 933
 - commands available to, 899
 - password file, authentication via, 21
 - privileges, 46–47, 898–900
 - users assigned, retrieving, 900
 - users assigned, setting maximum number, 50, 51, 169
 - users, granting to, 52, 101, 899
- SYSDBAOSDBA privilege, 47
- SYSOPER user/role
 - auditing, automatic, 933
 - commands available to, 899
 - password file, authentication via, 21
 - privileges, 46–47, 898–900
 - users assigned, retrieving, 900
 - users assigned, setting maximum number, 50, 51, 169
 - users, granting to, 52, 101, 899
- System Change Number (SCN), 256, 263, 431, 464
- SYSTEM Datafiles, moving, 356–358

- System Global Area (SGA)
 - instances, role in, 6, 7–8
 - memory structures, 9–13
 - Multi-Threaded Server (MTS) environment, setting
 - maximum memory use in, 811
 - system identifier (SID)
 - databases, 155, 166–167
 - service creation, specifying during, 169
 - sessions, 120
 - UNIX systems, on, 166–167
 - system monitor (SMON), 6, 13–14
 - SYSTEM role. *See* SYSTEM user/role
 - SYSTEM tablespace
 - CREATE DATABASE command, automatically
 - created with, 409
 - data dictionary object permissions in, 197
 - data dictionary structures, reserved for,
 - 334, 362, 409
 - datafiles belonging to, moving, 355, 356–358
 - datafiles belonging to, resizing, 159
 - datafiles belonging to, setup in CREATE DATABASE
 - operation, 175
 - described, 334
 - fragmentation, 337, 411
 - location, 146
 - offline, 345
 - rollback segments, SYSTEM, 178, 330, 334, 432
 - user temporary tablespace, as, 336–337
 - SYSTEM user/role, 43–44, 178, 847, 895
 - creation with database, 43–44
 - DBA role granted to, 44
 - password, 43
- T**
- TAB\$ table, 334
 - tables. *See also* columns; rows
 - auditing, 940, 941, 943–945
 - backups, checking for, 547
 - block space deallocation, 537–541
 - block space utilization parameters, 508–509,
 - 522–523, 525–527, 528–529, 531
 - block space utilization parameters, viewing,
 - 546–549
 - blocks, sharing with other objects, 517
 - buffer pool information, retrieving, 549, 550
 - cache parameters, 508, 509, 511, 531, 548
 - cluster name, retrieving, 546
 - clustered, INSERT operations, 694
 - constraints, defining during table creation, 654–656
 - constraints, dropping before truncation, 542
 - constraints, dropping from existing, 657
 - constraints, enabling using exceptions tables,
 - 661–663
 - constraints, FOREIGN KEY violations caused by
 - dropping, 651–652
 - constraints, locking caused by, 652–654, 659
 - constraints, modifying on existing, 657–658
 - constraints, relationship enforcing using FOREIGN
 - KEY, 638–639
 - constraints, rendering read-only using,
 - 644–645, 660
 - constraints, retrieving information about, 664, 665
 - copying, 512–514, 735
 - creating, 487–488
 - creating on clusters, 491, 517–522
 - creating using CREATE TABLE, 507–514
 - creating using create table as select (CTAS),
 - 735–736
 - creating using Oracle Schema Manager, 514
 - disks, distributed across, 490
 - dropping, 514, 542, 651–652
 - exceptions tables, 661–663
 - exporting, 517, 524
 - exporting table data, 739, 740, 742, 748
 - exporting table definitions, 739, 740, 741, 742, 746
 - exporting table grants, 739, 740, 741, 742
 - extents in, listing, 551–552
 - extents parameters, 509, 510–511, 522, 531–534
 - free lists, assigning to, 509, 524, 534
 - free lists, retrieving number assigned to, 547, 550
 - high-water mark, 537–541
 - import operations, ignoring table creation
 - errors, 758
 - importing data into existing, 774
 - importing table data, 739, 740, 742, 759
 - importing table definitions, 739, 740, 742
 - importing table grants, 742
 - indexes, on temporary, 517
 - indexes, retrieving, 614
 - indexes, specifying for, 597, 602
 - indexes, when moving, 606
 - index-organized (IOT), 393, 490, 523–524, 548
 - insert operations, 488
 - instances scanned across, retrieving
 - number of, 548
 - internal, 334
 - joined, clustering, 491
 - locking, 548, 652–654, 659

- logging parameters, 508, 511–512, 531, 547
- moving, 534–536, 606, 735–736
- name, 507
- name, retrieving, 546
- nesting, 397, 505–506, 549
- NEXT parameter, 509, 531
- objects, as, 848
- ownership, 507, 524
- ownership, retrieving, 545, 546
- parallelism, 548, 610, 691–694, 712
- partitioning, 489–490, 548, 585
- PCTINCREASE parameter, 509, 523, 531
- privileges for creating, 507, 772, 897
- privileges for dropping, 542
- privileges for selecting, 200, 907–909
- privileges for truncating, 541
- queries against temporary, 517
- query results, storing in temporary, 517
- read-only using constraints, 644–645, 660
- rebuilding, 534
- referential integrity, 638
- regular tables, 488
- relationships, maintaining using ROWID, 500
- segments, table, 392
- statistics, retrieving, 546, 548, 549
- storage parameters, 508–511, 522–523, 530–532
- storage parameters, retrieving, 546–547
- tablespaces, 507, 523–524
- tablespaces, moving between, 534–536
- tablespaces, rebuilding in same, 737
- temporary, 514–517
- temporary, determining data duration, 549
- temporary, determining if, 546, 548
- temporary, indexes on, 517
- transaction slot settings, 508
- truncating, 541–542
- types, 488–491
- views, 517, 545–553
- virtual, 206
- TABLES parameter
 - Export utility, 748
 - Import utility, 759
- TABLESPACE CREATE TABLE command parameter, 507
- Tablespace Manager utility, 61
- tablespaces. *See also* segments; *specific tablespaces*
 - administrative considerations, 335
 - assigning default, 802, 852, 855, 858
 - assigning temporary, 852, 855, 858
 - auditing operations on, 940
 - CD-ROM, moving to, 347, 355
 - clusters, for, 518
 - coalescing of free space, 14, 341
 - constraint violations when dropping, 652
 - creating offline, 342
 - creating using CREATE TABLESPACE command, 339–342
 - creating using Oracle Storage Manager, 343–345
 - databases, relation to, 328, 330–331
 - datafile tablespace name, retrieving, 360, 407
 - datafile tablespace number, retrieving, 263
 - datafiles, adding, 351
 - datafiles, configuring tablespace creation, 339–340
 - datafiles, listing, 778
 - datafiles, multiple per, 340
 - datafiles, relation to, 328
 - datafiles, sizing appropriately, 159
 - default, assigning, 802, 852, 855, 858
 - default, changing, 863–864, 865
 - dictionary-managed, 334, 337, 344, 360, 401
 - dropping, 358–359, 652
 - entity relational model, place in, 328
 - export operations, transportable tablespace, 776–782
 - exporting tablespace definitions, 742, 748–749
 - exporting tablespace quotas, 742
 - extent tablespace information, retrieving, 360, 551
 - extents, in temporary, 338
 - extents, relation to, 332
 - files belonging to, retrieving, 263
 - fragmentation, 411
 - import operations, conflicts in, 757
 - import operations, tablespace mode, 760
 - importing tablespace definitions, 742, 760
 - importing tablespace quotas, 742
 - index tablespaces, 330, 335, 599, 602, 605
 - index tablespaces, relocating, 523
 - index tablespaces, retrieving, 609
 - index tablespaces, spanning, 585
 - index tablespaces, specifying, 597
 - index tablespaces, temporary, 599
 - input/output (IO) considerations, 146–147
 - locally-managed, creating, 342, 401
 - locally-managed, for rollback segments, 440
 - locally-managed, for tables, 524
 - locally-managed, temporary, 337, 342

Continued

- tablespaces (*continued*)
 - locally-managed, UNIFORM SIZE parameter, 337, 363, 398–399
 - location, 146–147
 - log mode, retrieving, 360
 - maintenance, offline, 345
 - moving, 352–358
 - names, retrieving, 263, 360, 407
 - number of, retrieving, 263
 - objects in, permanent, 336, 852
 - objects in, temporary, 336–338
 - offline, 342, 345–347
 - offline, bringing online, 346–347
 - Oracle Enterprise Manager (OEM) repository
 - tablespaces, 67–68
 - partitioning considerations, 335
 - quota privileges, 897
 - quotas, assigning, 850–852, 855, 859, 897
 - quotas, importing/exporting, 742
 - quotas, removing, 864
 - quotas, views, 870–871
 - read-only, 331, 347–348
 - read-write, 331, 348
 - rollback segment tablespaces, 146, 330, 335, 411, 440
 - rollback segment tablespaces, errors in taking
 - offline, 466
 - rollback segment tablespaces, retrieving name, 443
 - security, 802
 - segment tablespaces, retrieving name, 550
 - segments, relation to, 328, 338
 - size, 331
 - size, changing, 350–352
 - sorting, used in, 337
 - status, retrieving, 360
 - storage hierarchy, place in, 328
 - storage settings, changing, 348–349
 - storage settings, retrieving, 360
 - storage structure, place in, 330–331, 391
 - table tablespaces, 507, 523–524
 - tables, moving between, 534–536
 - tables, rebuilding in same, 737
 - temporary, assigning, 852, 855
 - temporary objects in, 336–338
 - temporary tablespaces, 330, 331, 335, 336–337, 342
 - temporary tablespaces, converting
 - permanent to, 338
 - temporary tablespaces, security, 802
 - transportable, 776–782
 - types, 334–335, 362–363
 - User tablespaces, 330, 331, 335
 - views, 359–362
- TABLESPACES parameter
 - Export utility, 748
 - Import utility, 760
- TCL. *See* Tool Control Language (TCL)
- TechNet Web site, 1127
- Technology Tracks, 1066
- TEMP tablespaces, 362, 410, 411
- temporary files, returning information about, 263
- TEMPORARY TABLESPACE CREATE USER
 - parameter, 855
- time
 - datatypes, 497
 - dates, storage with, 497
 - National Language Support (NLS) settings, 1025, 1026, 1038
- timestamp
 - National Language Support (NLS) display settings, 1038
 - retrieving, 546
- TNSNAMES.ORA file, 149
- TO_CHAR function, 497, 1042
- TO_DATE function, 1042
- TO_NUMBER function, 1042
- Tool Control Language (TCL), 72
- Tools tablespaces, 331, 410, 411
- TOO_MANY_ROWS exceptions, 213
- TopSessions utility, 61
- TOUSER Import utility command, 757, 760, 767, 779
- trace files, 100–101, 122–125, 163, 260–261
- tracking resource usage, 101
- TRANSACTIONAL shutdown mode, 93–94, 110, 111
- transactions
 - committed, 302, 428
 - DDL statements as, 428
 - idle, 465
 - index transaction slot parameters, 598, 599, 609–610
 - information about, retrieving, 451–453, 455–456
 - recovery of committed, 302
 - rollback segments, assignment to, 427, 428–429, 430, 433
 - rollback segments, multiple sharing, 100, 434, 467
 - rollback segments, recovery role, 431
 - rollback segments, relation to, 40, 427–429, 433–435
 - rollback segments, retrieving transactions using, 455–457

- rollback segments, simultaneous header access attempts, 467
 - rollback segments, slots, 404
 - rollback segments, space requirements, 427, 463–464
 - rollback segments transaction table, 434
 - shutdown options, 93–94
 - Structured Query Language (SQL) transaction processing, 24–26
 - tables, temporary during, 514–517
 - views, 451–453
 - TRANSACTIONS_PER_ROLLBACK_SEGMENT initialization parameter, 100, 454, 455
 - TRANSPORT _TABLESPACE parameter
 - Export utility, 748, 749, 777
 - Import utility, 760
 - TRANSPORT_SET_VIOLATIONS view, 780, 782
 - trial version, downloading, 1065–1066
 - triggers
 - AFTER events, 215
 - auditing operations on, 940
 - auditing using, 933–934
 - BEFORE events, 215
 - BEFORE INSERT triggers, 632
 - business rules, planning around, 214
 - client event triggers, 223
 - described, 214
 - enabling/disabling, 224
 - events, using with, 214–215, 222–224
 - exporting, 739, 740, 741, 743
 - headers, 215
 - importing, 739, 740, 743
 - INSERT triggers, 214, 215, 219, 694, 712–714
 - INSTEAD OF triggers, 220–222
 - integrity, ensuring using, 214–215, 632, 633
 - objects, as, 848
 - order of firing, 220
 - predicates, 218–220
 - privileges, 223
 - resource manager triggers, 222
 - row-level, 216–218
 - statement-level, 215–216
 - system event triggers, 222
 - tables, on temporary, 517
 - UPDATE triggers, 632
 - user event triggers, 223
 - views, using in complex, 220–222
 - WHEN clauses, 218, 224
 - TRIGGERS Export utility parameter, 777
 - TRUNCATE ANY TABLE privilege, 541
 - TRUNCATE TABLE command, 541–542
- ## U
- UET\$ table, 334
 - Underground Oracle FAQ Web site, 1127
 - Undo Segment Number (USN), 445, 446
 - Unicode encoding, 1027, 1029
 - UNIFORM SIZE parameter, 363, 398–399, 411
 - UNIQUE constraints, 597, 608, 636–639, 649–651
 - Universal Character Set Transformation Format character set, 1029
 - Universal Character Set two-byte form, 1029
 - Universal Installer, 37–42. *See also* installation
 - UNIX systems
 - auditing, 933, 937
 - authentication, operating system, 47–48
 - background daemons, 70
 - environment variables, 167–169
 - installation, Oracle, 38, 48
 - National Language Support (NLS), 1036
 - password file, 50–51, 144, 173
 - program units, accessing external, 210
 - system identifiers (SIDs), 166–167
 - UNLIMITED TABLESPACE privilege, 438, 507, 897, 898
 - UPDATE privilege, 915
 - UPDATE triggers, 632
 - UPDATING trigger predicate, 219
 - UROWID datatype, 500
 - UROWID RAW datatype, 494, 499–500
 - US7ASCII character set, 168, 177, 1027, 1030, 1031
 - USER_ data dictionary views, 198, 200, 202–203
 - user process, 7
 - User tablespaces, 330, 331, 335
 - USER_COL_PRIVS view, 930
 - USER_COL_PRIVS_MADE view, 930
 - USER_COL_PRIVS_RECD view, 930
 - USER_DUMP_DEST initialization parameter, 100, 163
 - USERID parameter
 - Export utility, 745
 - Import utility, 756
 - SQL*Loader, 703
 - Username CREATE USER parameter, 854
 - users
 - auditing, 940, 941
 - CATALOG.SQL script, created by, 847
 - creating, 853–860
 - data dictionary storage of user info, 196

Continued

users (*continued*)

- definitions, importing/exporting, 742
- dependencies, retrieving, 866–867
- dropping, 866–869
- modifying, 863–866
- name, assigning, 854, 858
- operating system, mapping to username, 861
- Oracle Enterprise Manager (OEM), managing using, 857–860, 864–865, 868–869
- overview, 847–849
- privileges, allowing to grant, 914–916, 919–920, 923
- privileges for altering, 813, 897
- privileges for creating, 813, 897
- privileges granted by, retrieving, 930
- privileges granted to, retrieving, 900, 910–913, 929–932
- profile assignment, 804, 815, 855, 858
- roles enabled, retrieving, 1002–1003
- roles granted to, retrieving, 1000
- roles, granting by, 976, 992
- roles, granting to, 976, 991–993
- security considerations, 852–853
- simultaneous, specifying maximum, 153, 169
- SQL.BSQ script, users created by, 847
- tablespace assignment considerations, 852
- views, user, 869–871
- users, privileged. *See also* SYSDBA user/role; SYSOPER user/role
 - authentication, 45–46
 - described, 43–44
 - remote connections, 45–46
- USERS tablespace, 864
- USER_TAB_PRIVS view, 930
- USER_TAB_PRIVS_MADE view, 930
- USER_TAB_PRIVS_RECD view, 930
- USN. *See* Undo Segment Number (USN)
- UTF8 character set, 1029
- UTL_FILE_DIR parameter, 302–303
- utlpwdmg.sql file, 808
- UTL*.SQL scripts, 238

V

- V\$ views, 112. *See also* dynamic performance views
- V\$ACCESS view, 113
- VALIDATE STRUCTURE command, 609
- validation using constraints. *See* integrity constraints
- VARCHAR2 datatype, 493, 1029
- V\$ARCHIVE view, 263
- V\$ARCHIVED_LOG view, 114, 263

variables

- packages, in, 225, 226, 228
- procedures, in, 212
- VARRAY datatype, 229, 397
- VARRAYs, 502–504
- V\$BACKUP view, 263
- V\$BACKUP_SET view, 114
- V\$BGPROCESS view, 113
- V\$CONTROLFILE view, 114, 261
- V\$CONTROL_FILE_RECORD_SECTION view, 114, 262
- V\$DATABASE view, 114, 263, 293–294
- V\$DATAFILE view, 112, 114, 263, 361
- V\$DATAFILE_HEADER view, 114
- versions, database migration from previous, 77–78
- V\$FIXED_TABLE view, 113
- views. *See also* dynamic performance views; *specific views*
 - auditing information, using to retrieve, 946–952
 - auditing operations on views, 940, 943–944
 - column views, 552–553, 665, 930
 - control file data, displaying using, 261–263
 - control file data, views using, 263
 - creating, 232–238
 - data dictionary views, 195
 - database administrator (DBA_) views, 198, 200–202
 - datafile views, 360–361, 407
 - dependency views, 232
 - exporting, 740, 742, 743
 - extent views, 408, 551–552
 - importing, 740, 742, 743
 - index views, 605, 609–615, 665, 666
 - materialized, 331
 - missing, re-creating, 234
 - National Language Support (NLS) views, 1045–1049
 - network authentication view, 870
 - object views, 198, 200, 202–204, 230–232, 545–546
 - objects, as, 848
 - passwords, of, 869
 - privileges for creating, 895, 907–909
 - privileges, of, 200, 900, 910–913, 929–932
 - profile views, 824–826
 - queries, using in place of, 199
 - querying views, 234
 - quota views, 870–871
 - redo log file views, 291–293
 - rollback segment views, 443–453
 - segment views, 362, 407, 549–550
 - segments *versus*, 331

- sorting-related, 362
 - storage-related, 407–408
 - table views, 517, 545–553
 - tablespaces views, 359–362
 - transaction views, 451–453
 - types, 199–200
 - user views, 869–871
- V\$INSTANCE view, 120, 294
- V\$INSTANCE_RECOVERY view, 285–286
- virtual tables, 206
- V\$LOG view, 114, 263, 292
- V\$LOGFILE view, 114, 263, 281, 292–293
- V\$LOGHIST view, 263
- V\$LOG_HISTORY view, 114
- V\$LOGMNR_views, 304, 305
- V\$MYSTAT view, 113
- V\$NLS_PARAMETERS view, 1048–1049
- V\$NLS_VALID_VALUES view, 1049
- VOLSIZE parameter
 - Export utility, 749
 - Import utility, 760
- V\$OPTION view, 113
- V\$PARAMETER view, 113, 116, 118
- V\$PROCESS view, 113
- V\$PWFILERS view, 113, 900
- V\$ROLLNAME view, 445

- V\$ROLLSTAT view, 443, 445–451, 461
- V\$SESSION view, 113, 120–121
- V\$SGA view, 113
- V\$SORT_SEGMENT view, 361–362
- V\$SORT_USAGE view, 361–362
- V\$TABLESPACE view, 114, 263
- V\$TEMPFILE view, 263
- V\$THREAD view, 291
- V\$TRANSACTION view, 443, 451–453
- V\$VERSION view, 113

W

- WE8DEC character set, 1027
- WE8ENCDCIC500 character set, 1027
- WE8ISO8859P1 character set, 1027–1028, 1030, 1031
- Web sites, 1126–1127
- WHEN OTHERS exceptions, 213
- Windows Registry, editing for automatic startup, 106–107
- wrapper utilities, 238

X

- X\$ views, 208

Z

- ZIM Technologies Inc., 1068