Lecture Notes in Computer Science 1846
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Hongjun Lu   Aoying Zhou (Eds.)

Web-Age Information Management

First International Conference, WAIM 2000
Shanghai, China, June 21–23, 2000
Proceedings

Springer

Volume Editors

Hongjun Lu
Hong Kong University of Science and Technology
Department of Computer Science
Clear Water Bay, Kowloon, Hong Kong, China
E-mail: luhj@cs.ust.hk

Aoying Zhou
Fudan University
Department of Computer Science
220 Handan Road, Shanghai, China
E-mail:ayzhou@fudan.edu.cn

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Web age information management : first international conference ;
proceedings / WAIM 2000, Shanghai, China, June 21 - 23, 2000. Hongjun
Lu ; Aoying Zhou (ed.) - Berlin ; Heidelberg ; New York ; Barcelona ;
Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1846)
ISBN 3-540-67627-9

CR Subject Classification (1991): H.2, H.3, H.4, I.2, H.5, J.1

ISSN 0302-9743
ISBN 3-540-67627-9 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
Printed on acid-free paper SPIN 10722078 06/3142 5 4 3 2 1 0

Preface

Database research and development has been remarkably successful over the past three decades. Now the field is facing new challenges posted by the rapid advances of technology, especially the penetration of the Web and Internet into everyone's daily life. The economical and financial environment where database systems are used has been changing dramatically. In addition to being able to efficiently manage a large volume of operational data generated internally, the ability to manage data in cyberspace, extract relevant information, and discover knowledge to support decision making is critical to the success of any organization. In order to provide researchers and practitioners with a forum to share their experiences in tackling problems in managing and using data, information, and knowledge in the age of the Internet and Web, the First International Conference on Web-Age Information Management (WAIM 2000) was held in Shanghai, China, June 21-23.

The inaugural conference in its series was well received. Researchers from 17 countries and regions, including Austria, Australia, Bahrain, Canada, China, France, Germany, Japan, Korea, Malaysia, The Netherlands, Poland, Singapore, Spain, Taiwan, UK, and USA submitted their recent work. Twenty-seven regular and 14 short papers contained in these proceedings were presented during the two-day conference. These papers cover a large spectrum of issues, from classical data management such as object-oriented modeling, spatial and temporal databases to recent hits like data mining, data warehousing, semi-structured data, and XML. More importantly, there are also sections of papers devoted to related research areas, such as information retrieval, artificial intelligent agents, and electronic commerce, which broaden the scope of the conference to cover the entire process of data management, information retrieval and dissemination, knowledge discovery, and applications with the Internet as the media.

Any successful conference requires tremendous efforts from numerous people. In addition to the people in various conference organization committees, we would like to thank ACM SIGMOD who supported the conference from its planning stage and granted it the in-cooperation status. We are grateful to our sponsors, in particular the K. C. Wong Education Foundation, Hong Kong, for their generous support.

June 2000 *Hongjun Lu, Aoying Zhou*

Conference Organization

General Chairs:

Rakesh Agrawal	*IBM Almaden Research Center, USA*
Baile Shi	*Fudan University, China*

Program Committee Chairs:

Hongjun Lu	*Hong Kong University of Science and Technology, China*
Aoying Zhou	*Fudan University, China*

Organizing Committee Chairs:

Shoucai He	*Shanghai No.2 University of Technology, China*
Jinhai Chen	*Fudan University, China*

Publicity Chairs:

X. Sean Wang	*George Mason University, USA*
Yujun Wang	*Fudan University, China*

Local Arrangement:

Jiajin Le	*China Textile University*
Yangyong Zhu	*Shanghai Computer Society, China*

Program Committee:

Arbee Chen	*National Tsing Hua University, Taiwan*
Chen Guoqing	*Tsinghua University, China*
Ming-Syan Chen	*National Taiwan University, Taiwan*
David W. Cheung	*Hong Kong University, China*
Umeshwar Dayal	*Hewlett-Packard Laboratories, USA*
Guozhu Dong	*Wright State University, USA*
Martin Ester	*University of Munich, Germany*
Feng Yucai	*Huazhong University of Science and Technology, China*
Johannes Gehrke	*Cornell University, USA*
Stephane Grumbach	*INRIA, France*
Jiawei Han	*Simon Fraser University, Canada*
He Xingui	*Beijing Institute of System Engineering, China*
Kamal Karlapalem	*Hong Kong University of Science and Technology, China*
Masaru Kitsuregawa	*University of Tokyo, Japan*
Chiang Lee	*National Cheng-Kung University, Taiwan*
Qing Li	*City University of Hong Kong, China*
Jianzhong Li	*Harbin Institute of Technology, China*
Li Zhanhuai	*Northwestern Polytechnical University, China*
Tok Wang Ling	*National University of Singapore, Singapore*
Alberto Mendelzon	*University of Toronto, Canada*
Mukesh Mohania	*University of South Australia, Australia*

Shojiro Nishio	*Osaka University, Japan*
Ooi Beng Chin	*National University of Singapore, Singapore*
Maria E. Orlowska	*University of Queensland, Australia*
Michael Papazoglou	*Tilburg University, The Netherlands*
Qu Zhaorong	*East China Institute of Computer Technology, China*
Ming-Chien Shan	*Hewlett-Packard Laboratories, USA*
Jianwen Su	*University of California at Santa Barbara, USA*
Zhao-Hui Tang	*Microsoft, USA*
Tang Changjie	*Sichuan University, China*
Tang Shiwei	*Peking University, China*
Tian Zengping	*Fudan University, China*
Tong Fu	*Shanghai University, China*
X. Sean Wang	*George Mason University, USA*
Wang Shan	*Renmin University of China, China*
Kyu-Young Whang	*KAIST, Korea*
Wong Lim Soon	*Kent Ridge Digital Laboratories, Singapore*
Xu Jiepan	*Nanjing University, China*
Jian Yang	*CSIRO, Australia*
Jeffrey X. Yu	*Chinese University of Hong Kong, China*
Philip S. Yu	*IBM T.J. Watson Research Center, USA*
Yu Ge	*Northeastern University, China*
Yanchun Zhang	*University of Southern Queensland, Australia*
J. Leon Zhao	*University of Arizona, USA*
Zhou Lizhu	*Tsinghua University, China*
Zhou Longxiang	*China Academy of Science, China*

The First International Conference on Web-Age Information Management was organized by

Fudan University, China

in cooperation with

ACM SIGMOD
Shanghai Computer Society

and sponsored by

K.C. Wong Education Foundation, Hong Kong

Table of Contents

Visual Data Mining for Business Intelligence Applications

Ming Hao, Umeshwar Dayal, Meichun Hsu

Hewlett-Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94304, USA
{mhao, dayal, mhsu}@hpl.hp.com

Abstract. Business intelligence applications require the analysis and mining of large volumes of transaction data to support business managers in making informed decisions. A key dimension of data mining for human decision making is information visualization: the presentation of information in such a way that humans can perceive interesting patterns. Often, such visual data mining is a powerful prelude to using other, algorithmic, data mining techniques. Additionally, visualization is often important to presenting the results of data mining tasks, such as clustering or association rules. There are several challenges to providing useful visualization for business intelligence applications. First, these applications typically involve the navigation of large volumes of data. Quite often, users can get lost, confused, and overwhelmed with displays that contain too much information. Second, the data is usually of high dimensionality, and visualizing it often involves a series of inter-related displays. Third, different visual metaphors may be useful for different types of data and for different applications. This paper discusses VisMine, a content-driven visual mining infrastructure that we are developing at HP Laboratories. VisMine uses several innovative techniques: (1) hidden visual structure and relationships for uncluttering displays; (2) simultaneous, synchronized visual presentations for high-dimensional data; and (3) an open architecture that allows the plugging in of existing graphic toolkits for expanding its use in a wide variety of visual applications. We have applied this infrastructure to visual data mining for various business intelligence applications in telecommunication, e-commerce, and Web information access.

1 Introduction

Business Intelligence is the gathering, management, and analysis of large amounts of data on a company's customers, products, services, operations, suppliers, and partners and all the transactions in between. Examples of business intelligence applications include traffic analysis, fraud detection, and customer loyalty (churn) analysis in the telecommunications industry, which require analyzing large volumes of call detail records [1,2]; and target marketing, market basket analysis, customer profiling, and fraud detection in the e-commerce industry, which require analyzing large volumes of shopping transaction data from electronic storefront sites [3].

Typically, business intelligence applications involve extracting data from operational databases, transforming the data and loading it into data warehouses. The data in the warehouse is then input to a variety of reporting, querying, on-line analytical processing (OLAP) and data mining tools, which generate results used by business managers for making decisions. An important requirement of these applications is information visualization. Visualization is important because presenting information visually to analysts often enables them to perceive patterns in the data, and to select the subsets of data on which to focus further detailed analysis. For instance, a human can visually discern outliers or interesting regions that he can select for more detailed analysis. Furthermore, patterns and rules discovered as a result of using OLAP and data mining tools (such as clustering or association rules for market basket analysis) are more readily comprehended by business managers when presented visually.

Recent research efforts have focused on visual mining in many different areas, such as telecom switch data, World Wide Web traffic, company organization charts, and file systems [4,5,6,7,8,9,15]. There are several information visualization products available today. IBM's Intelligent Miner organizes data so as to make maximum use of significant pattern recognition. SGI's MineSet uses 3D animation to represent knowledge extracted from large data sets. AT&T Bell Laboratory's SeeNet uses 3D layout and direct user graphic interfaces to visualize telecom network activities. MindMan helps individuals organize, generate and learn ideas and information with multiple displays.

There are many challenges to be addressed in building visualization systems for business intelligence applications. Current information visualization systems are designed to handle moderate amounts of structured data. New information visualization systems will be built around the navigation of, and interaction with, massive volumes of unstructured information. The challenge is to find methods for presenting valuable information from large volumes of data so as to enable a user to quickly identify exceptions and to distinguish interesting patterns visually. The following are some issues in today's visual mining of massive volumes of data:

1. Cluttered displays.
2. Disjoint displays (display-after-display presentations).
3. Limited visual representations and lack of extensibility.

Presenting a large amount of information on a single complex view often causes display clutter and visual confusion. Furthermore, single view visualization does not allow users to visualize complex inter-relationships among different sets of high-dimensional data. A common solution is to provide multiple views using many displays. However, now users have to click through display after display to find the information. Often, users have great difficulty to analyze and correlate information from these disjoint displays. For example, in a telecom traffic analysis application, suppose a user wants to selectively monitor overloaded telephone links in the United States. Starting with a display of a United States map, the user would need to click though each display of progressively greater detail (at the state level, at the city level, etc) until he finds the overloaded links. With multiple views, the user can see presentations at different levels of detail simultaneously to identify the problems in real time.

Most recent graphic toolkits are designed for visualization of certain types of information. They provide a fixed set of visual representations (metaphors or templates). For example, hyperbolic trees are excellent for representing hierarchical relationships among a large number of data objects, not for graphs or networks that are necessary for telecom traffic analysis or web site navigation analysis [6]. Similarly, other templates such as 3-D histograms, bar charts, and maps are designed for specific uses. To meet the different requirements of different applications (or even to support multiple views for a single application), we need an extensible architecture.

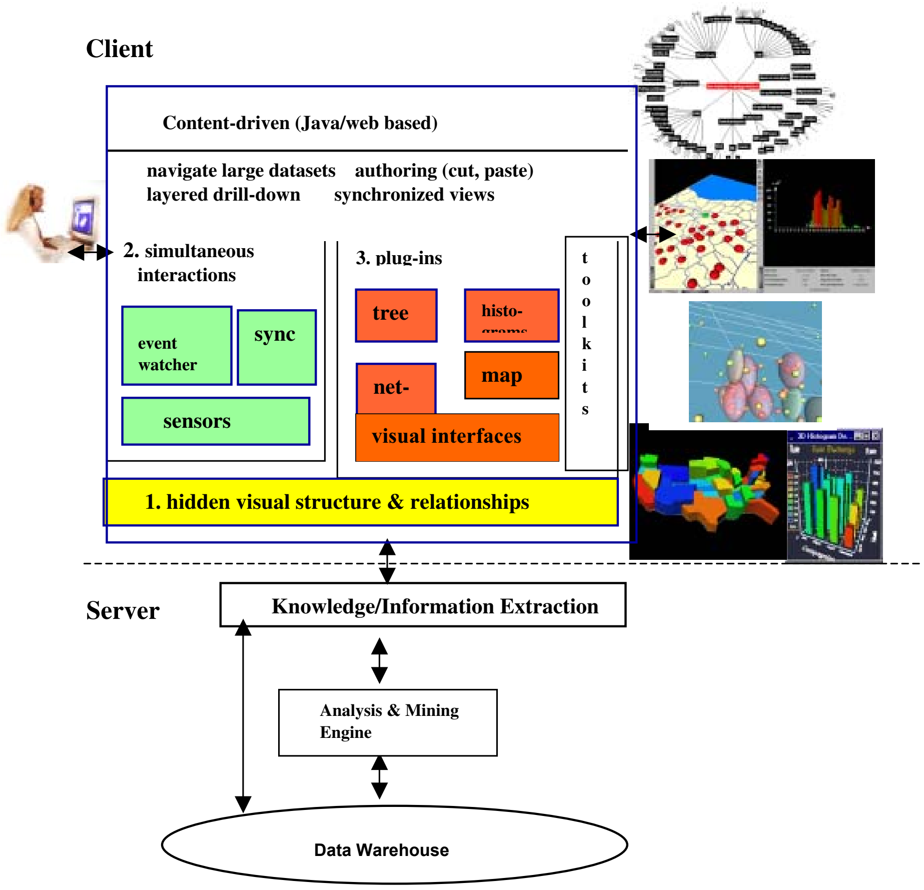


Fig. 1. Architecture of VisMine

In Figure 1, we show the architecture of VisMine, a visualization system that we are developing at Hewlett-Packard Laboratories, and that is designed to overcome the limitations described above [10]. The main features of VisMine are that it supports hidden structure and relationships to avoid clutter; it supports multiple synchronized views so that the user does not have to click through multiple disjoint displays; and it is based on an extensible architecture with well-defined visual interfaces that allow graphics toolkits to be wrapped and plugged in.

In the next section, we describe the architecture of VisMine and our approach to providing the capabilities outlined above. In Section 3, we describe a number of visual data mining applications for business intelligence in Web information access, telecommunication, and e-commerce that we have built using this platform.

2 Architecture

VisMine is built on a Java-based client-server model. Java allows the creation of visual component applets that can be automatically downloaded and executed on the local client. To achieve rapid display, VisMine separates the visualization from the data mining computation engine. The data mining computation executes on the server. The visualization construction and rendering are done locally in the client sites. As a result, VisMine is able to provide fast response needed for real-time visual mining.

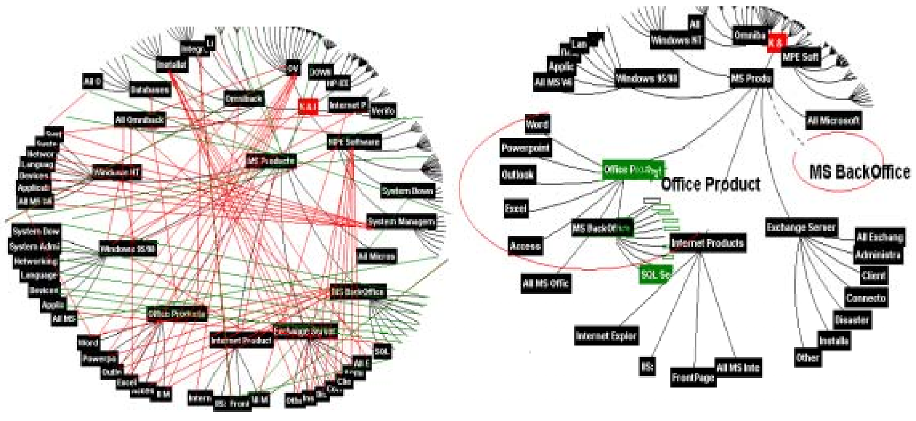
VisMine provides three types of interfaces. One interface is for extracting and loading information from the data sources into the VisMine client. The data sources might be databases, data warehouses, or files exported from analysis and mining tools. The second interface is a Web browser interface to the user. This supports user interactions with the visual views, and captures various screen and mouse events necessary for synchronizing multiple views. The interactions include the typical functions of zoom, rotate, point and click; in addition, functionality is provided for users to navigate and author (edit) visual representations, and to perform layered drill down for information that is represented at different levels of detail. The third interface is to plug in graphics toolkits.

2.1 Hidden Visual Structure and Relationships

A common method for visualizing information from applications such as fault diagnosis, document topic hierarchy navigation, or market segment analysis is to lay out all the structure and relationships on the screen, using a graph layout with nodes and links, or a matrix layout with cells. However, for large amounts of information with rich structure and complex relationships, these static techniques do not work. There are just too many lines, nodes, and cells to draw. As a result, the display becomes cluttered and causes visual confusion.

To reduce display cluttering and visual confusion, our approach is to hide irrelevant visual structure and relationships. The principle is to hide all non-primary relationships, and to show objects only when the user focuses on them. All other structures and relationships are hidden in the properties of each object.

We have applied this principle to visualizing large, complex graphs. For strictly hierarchical (tree-structured) data, the Hyperbolic Tree is an attractive and efficient visual representation [6]. Laying out a tree on a hyperbolic surface (which is then mapped to the 2-D screen) allows more nodes and edges to be drawn than on a flat or spherical surface. Extending this metaphor to general graphs, however, is a problem, as illustrated in Figure 2a. The multi-path hyperbolic graph becomes very cluttered if you display lines corresponding to all edges connecting nodes. This results in too many lines, intersections, and broken lines. VisMine exploits the fact that in many such applications, there is usually a primary relationship that is more or less hierarchical. For example, a topic hierarchy consists of a tree structure representing the primary topic-subtopic relationship, plus many cross links representing other secondary relationships among topics. For other applications such as traffic monitoring, the tree can be organized according to geographical relationships or service categories. At the time the visual space is initialized, all the non-primary structure and paths are hidden in each object's property. They become accessible and interactive only at the time of focus. Thus, the user can easily navigate through all possible paths without tracing many lines and intersections. This is shown in Figure 2b, where the multi-path hyperbolic space retains the simplicity of the original hyperbolic space, without using lines to display all edges. The user can easily navigate through all possible paths without tracing many lines and intersections. For example, the “Office Product” node contains a hidden-path to “MS BackOffice” (indicated as an empty circle). No extra line is needed to connect “MS BackOffice” node to the “Office Product” node. The “MS BackOffice” child node will be automatically mapped from its parent node, “MS Product” to the “Office Product” node when the user clicks on the “Office Product” node. At the conclusion of navigation, the path is again suppressed.



a. Traditional Method: without hidden links b. With hidden links

Fig. 2. A Multi-Path Hyperbolic Space

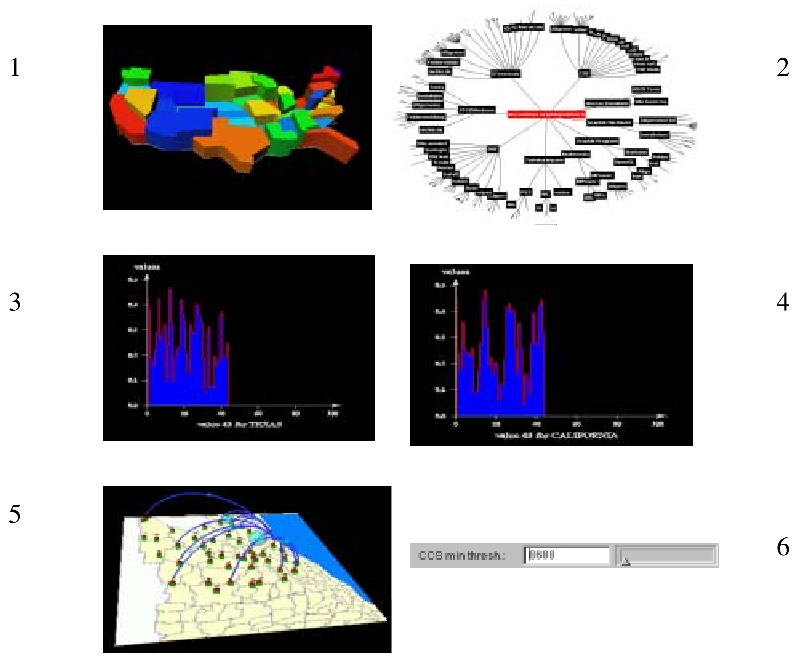
The idea of hidden links in dense graph layouts is different from Munzner’s hyperbolic graph viewer H3 [5]. In VisMine, only the primary directed links are shown. All other types of links (non-tree/secondary links) are made invisible and hidden in the property of a node. They only appear when the user navigates/clicks on a node. This technique allows a user to freely focus on the hierarchy of interest without tracing many lines.

2.2 Simultaneous Multiple Presentations

A second approach to dealing with large, complex data sets is to support multiple simultaneous presentations of the data. Each presentation can show either a different view over the same data, or one view may show more detail than another (e.g., traffic overlaid on a map, and a detailed 3-D histogram view showing hourly distribution of traffic during a day). These different views are synchronized in response to events.

There are two types of event interactions. One is direct from the Web user interface. For example, the user moves the slider on the display to select the telephone service units to visualize. The other type of event is from the mining engines; for example, an unexpected condition occurred in the middle of monitoring traffic data. VisMine assigns each visual group a sensor. The sensor will be automatically awakened to handle these incoming events and execute certain functions, such as setting up an alarm or activating a spotlight. For visual consistency, a change in one presentation will be propagated in real-time to the other presentations. An event synchronizer ensures that all changes occur simultaneously. For example, a user can set a new threshold during visualization, and the results will be reflected in all the presentations synchronously.

Figure 3 illustrates animated telephone traffic data. The six views are synchronously presented at the same time. Any change in one view will be automatically made in the other five views simultaneously. The Texas and California histograms are presented simultaneously, enabling a user to visually compare the traffic patterns, and their relationship with the rest of the country.



Presentation 1: uses a United States map to visualize nation-wide telephone traffic.
Presentation 2: uses a hyperbolic tree to represent the logical telephone switch connections.
Presentations 3 & 4: uses two histograms to represent Texas and California traffic data.
Presentation 5: uses animated presentation of 24-hour phone traffic overlaid on a city map.
Presentation 6: use a text field to enter min threshold of customer service seconds.

Fig. 3. Multiple simultaneous views of telephone traffic

2.3 Plug-in Capability

Different types of application require different visual representations. Sometimes, even within the same application, there is a need to use different graphic techniques for different visual representations, as we see in Figure 3. VisMine allows various existing toolkits to be plugged in. For example, VisMine interfaces with Inxight's hyperbolic tree toolkit [11] to represent the service classification hierarchy, and it interfaces with TGS's toolkits [12] to construct 3D-network visualizations for displaying maps and histograms of traffic service data.

To interface to different existing toolkits, VisMine defines a common visual interface, a class library, and a suite of visual protocols, such as VisConnect, VisSend, VisRecv, VisSync, Vis-Author, details of which we omit from this paper. The VisMine class library contains visual interfaces to the APIs of the existing graphic toolkits.

3 Visual Data Mining Applications

There are many data mining applications with large information structures that can employ the VisMine content-driven visualization model. We have prototyped several applications, which we will use to illustrate these techniques.

3.1 Topic Hierarchy for Web Document Navigation

The first example is to visualize a topic hierarchy for browsing a large collection of documents on the Web [13]. A hyperbolic space is constructed to present a topic hierarchy for some 200,000 documents. The topic hierarchy is constructed by mining the content of the documents and session logs that record accesses to these documents. The user can navigate the tree and read a document on demand. There are many non-hierarchical relationships among nodes. These relationships are constructed and displayed only when the user focuses on them. Using the hidden structure and relationship capabilities of VisMine, the user is able to navigate a large, highly connected topic hierarchy in a simple, uncluttered hyperbolic space on a single Web browser screen (as shown in Figure 4).

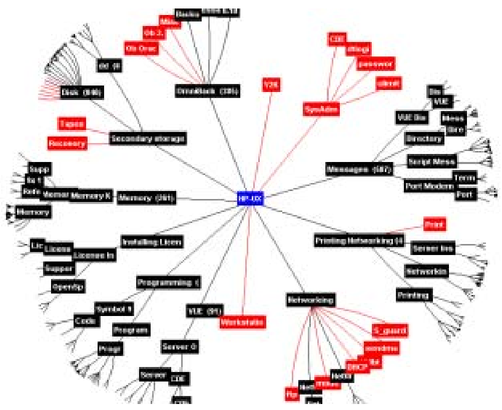
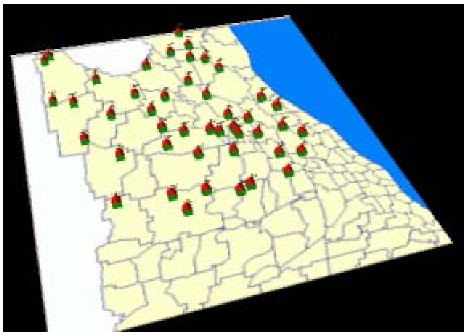


Fig. 4. A Web Document Topic Hierarchy

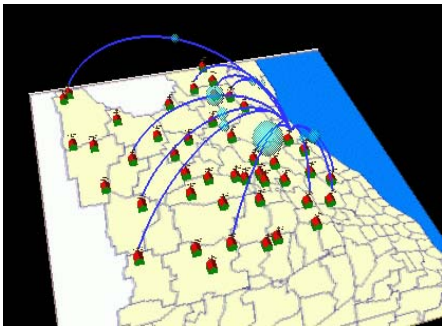
3.2 Telecom Traffic Analysis

Our next example is visual mining of telecom network traffic data. The geographical location of switches and the monitoring policy are organized into different group levels, such as country, state, service units, and busy hours for mining visually. Buttons, sliders, color scales, thresholds, and other visual parameters permit a user to dynamically select the information of interest to be displayed. VisMine hides non-primary structures and relationships and only displays objects when in focus. For example, initially all the phone switches located in a city are hidden on the map. This city map is displayed as a result of clicking on the desired city on the United States Map. This map shows a house and pole to represent the geographical position of each switch, and a directed arc to represent links between the caller and the callee (Figure 5a)**.**

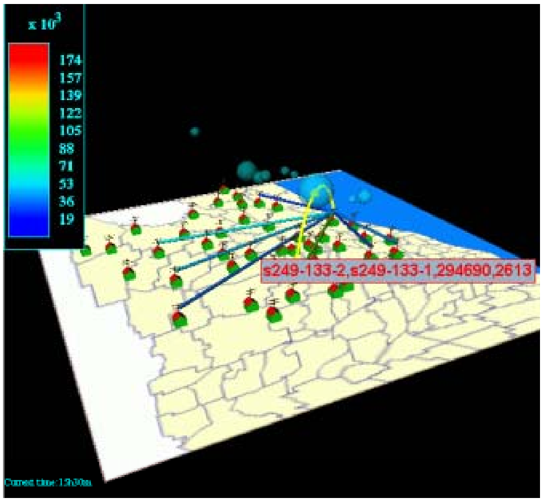
Upon a user's request, the current traffic can be monitored for switches where the activity exceeds some threshold set by the user. A slider is provided to control the threshold. Using 3D animation, changes of calling behavior over a 24-hour period can be monitored. By selecting a proper threshold, a user can easily identify which switches and links are being overloaded. The calling activity in call seconds is represented by spheres of different sizes. Figure 5b shows one such heavily loaded switch on the link from s249-133-2 to s249-133-1; the heavy activity occurred at 11:30, and the amount of the activity and number of calls on the link can be displayed by selecting the link. When a heavily loaded switch is discovered, the user can select it (by clicking on the house) to display a color histogram showing traffic distribution on that switch over a 24-hour period (Figures 5c,d).



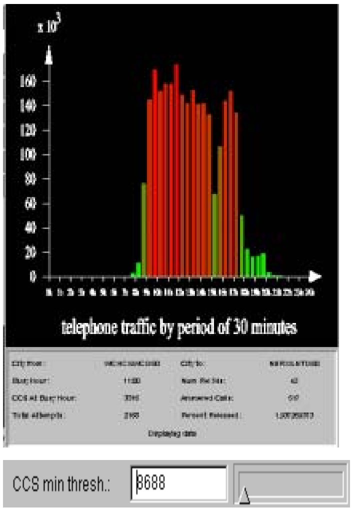
a. A city map displaying switches and links



b. A heavily loaded switch



c. Properties of the heavily loaded switch



d. 24-hour traffic on the switch

Fig. 5 Telecom Traffic Analysis

3.3 E-Commerce Applications

One of the common problems electronic store managers want to solve is how to target advertisements and increase product sales. We used a hyperbolic space graph visualization of the navigation paths followed by customers, i.e., the pages that a customer clicked through from referral sites and on the electronic store's web site. The nodes of the graph are web pages and the edges are URL links. We incorporate the invisible links technique to hide cross-links and cycles, displaying them only as they come into focus (Figure 6).

For example, the store manager can easily follow the paths from Yahoo, BuyOnline, and other referring sites to find out how many customers visited the ads on the referring sites and how many of these visits resulted in sales. The nodes "Yahoo" and "Buy Online" are defined as invisible links to the node "HighPCs". To graphically represent the contents of the node, each node includes attributes such as visiting time, access statistics, etc., that are displayed when the user clicks on the node. Color-coding is used to show the sales over a period of time.

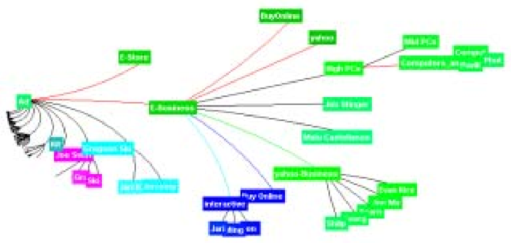


Fig. 6. Web Site Navigation Paths

Another common data mining task for e-commerce is market basket analysis, i.e., analyzing purchase transaction data to find products that are usually purchased together. This information is often used for target marketing or making product recommendations to customers. Figure 7 shows a visualization of market basket data, displayed using a mass-spring physical modeling system that we plugged into VisMine [14]. Input to the modeling system is a frequency matrix that shows how often each product is bought, and how often pairs of products are bought together. In the resulting presentation, the vertices represent products, and each vertex provides attribute information about the product, including product name, price, etc. Color-coding is used to show how often the product appears in the transaction database over a period of time. The distance in 3-D space between a pair of vertices is a measure of how often the two products are bought together (their joint support. The arcs represent directed product associations (there is an arc from product A to product B if the transaction data shows that with high confidence, when A is bought, B is bought as well).

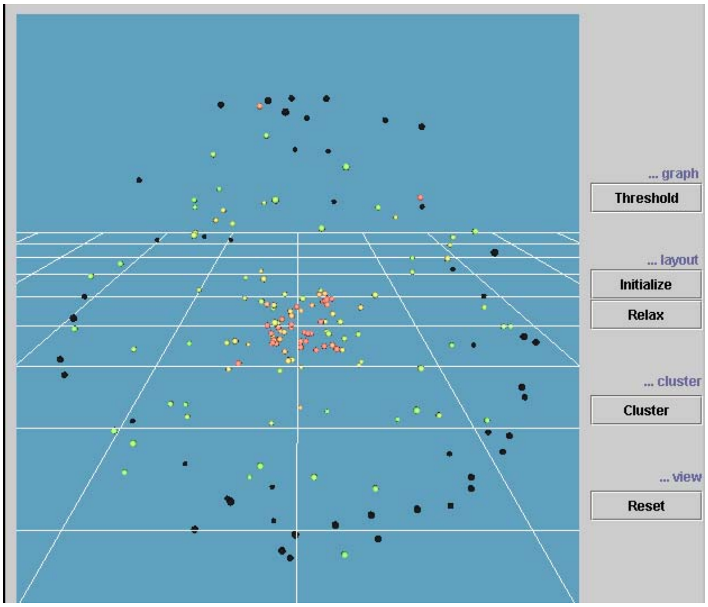
4 Conclusions

Data mining applications for business intelligence face challenges in the visual mining of massive, highly connected data sets.

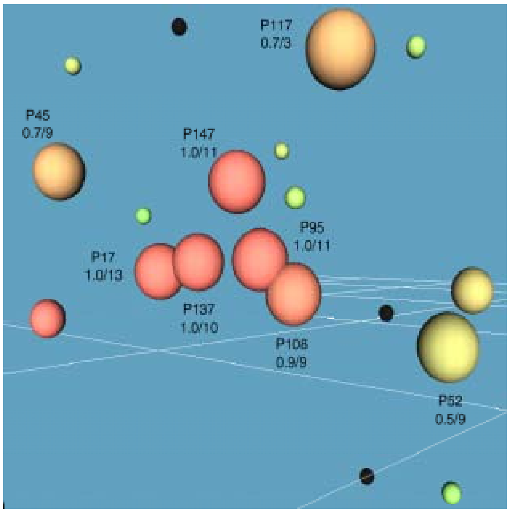
In this paper, we describe a visual mining infrastructure called VisMine. VisMine employs a hidden structure and relationship method to unclutter the display for massive data visualization. In addition, VisMine allows users to easily navigate through different synchronized views without being overwhelmed by a large number of disjoint presentations. VisMine is extensible to allow interfacing with various existing graphical toolkits that supply visual metaphors useful for different . These techniques have been successfully prototyped at Hewlett Packard Laboratories, and we showed several visual mining applications in telecommunication and e-commerce that we have developed using VisMine.

References

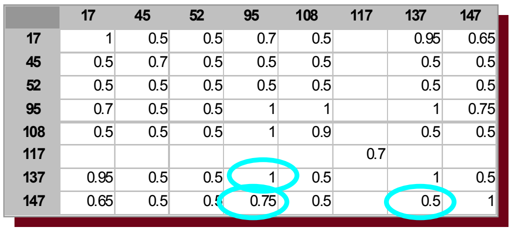
1. Qiming Chen, Umeshwar Dayal, Meichun Hsu, "OLAP-Based Scalable Profiling of Cus tomer Behavior", Proc. 1st Intl. Conf. on Data Warehousing and Knowledge Discovery (DAWAK), 1999.
2. Qiming Chen, Meichun Hsu, Umeshwar Dayal, "A Data Warehouse/OLAP Framework for Scalable Telecommunication Tandem Traffic Analysis," Proc. ICDE Conf., 2000.
3. Qiming Chen, Umeshwar Dayal, Meichun Hsu, "A Distributed OLAP Infrastructure for E-Commerce," Proc. 4th Intl. Conf. on Cooperative Information Systems, 1999.
4. Charlie Gunn, "Discrete Groups and Visualization of Three-dimensional Manifolds" ACM 1993. Various Hyperbolic Space" IEEE Computer Graphics. Vol. 18, Number 4. 1998.
5. Tamara Munzner, "Exploring Large Graphs in 3D Hyperbolic Space" IEEE Computer Graphics. Vol. 18, Number 4. 1998.
6. John Lamping and Ramana Rao, "Laying out and Visualizing Large Trees Using a Hyperbolic Space" ACM/UIST 1994.
7. Stephen G. Eick, "Aspects of Network Visualization", IEEE Computer Graphics and Applications, March 1996.
8. Joe C. Pinheiro, Don X Sun, "Methods for Linking and Mining Massive Heterogeneous Databases, KDD 1998.
9. Stephen G. Erick and Graham J. Wills, "Navigating large networks with hierarchies" IEEE Visualization, 1999.
10. Ming C. Hao, Umesh Dayal, Meichun Hsu, "A Java-based Visual Mining Infrastructure and Applications", IEEE InfoVis 1999.
11. Inxight Softwre, Hyperbolic Tree Toolkit.
12. Template Graphics Software, San Diego, CA.
13. Ming C. Hao, Meichun Hsu, Umesh Dayal, Adrian Krug, "A Technique for visualizing Large Web-based Hierarchical Hyperbolic Space with Multi-Paths", PADD, April 1999.
14. Thomas C. Sprenger, Markus H. Gross, D. Bielser, "Ivory – An Object-Oriented Framework for Physics-Based Information Visualization in Java", IEEE InfoVis, 1998.
15. Daniel A. Keim, Annemarie Herrmann, "The Gridfit Algorithms: An Efficient and Effective Approach to Visualizing Large Amounts of Spatial Data", IEEE Visualization, 1998.



a. Initialization



b. Detailed view of directed associations from a selected product



	17	45	52	95	108	117	137	147
17	1	0.5	0.5	0.7	0.5		0.95	0.65
45	0.5	0.7	0.5	0.5	0.5		0.5	0.5
52	0.5	0.5	0.5	0.5	0.5		0.5	0.5
95	0.7	0.5	0.5	1	1		1	0.75
108	0.5	0.5	0.5	1	0.9		0.5	0.5
117						0.7		
137	0.95	0.5	0.5	1	0.5		1	0.5
147	0.65	0.5	0.5	0.75	0.5		0.5	1

c. Support (frequency) matrix

Fig. 7 Market Basket Analysis

Parallel Data Mining on Large Scale PC Cluster

Masaru Kitsuregawa, Takahiko Shintani, Masahisa Tamura, and Iko Pramudiono

Institute of Industrial Science, The University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, Japan
{kitsure,shintani,masahisa,iko}@tkl.iis.u-tokyo.ac.jp

Abstract. PC cluster is recently regarded as one of the most promising platforms for heavy data intensive applications, such as decision support query processing and data mining. We proposed some new parallel algorithms to mine association rule and generalized association rule with taxonomy and showed that PC cluster can handle large scale mining with them. During development of high performance parallel mining system on PC cluster, we found that heterogeneity is inevitable to take the advantage of rapid progress of PC hardware. However we can not naively apply existing parallel algorithms since they assume homogeneity. We proposed the new dynamic load balancing methods for association rule mining, which works under heterogeneous system. Two strategies, called candidate migration and transaction migration are proposed. Initially first one is invoked. When the load imbalance cannot be resolved with the first method, the second one is employed, which is costly but more effective for strong imbalance. The experimental results confirm that the proposed approach can very effectively balance the workload among heterogeneous PCs.

1 Introduction

Recently commodity based PC cluster system is regarded as one of the most promising platforms for data intensive applications such as decision support query processing and data mining. The power of PC is superior to the workstation for integer performance and the price of PC is also much lower. So far extensive researches on parallel database processing algorithms have been done[3]. Currently parallel execution option is available for most of RDB products. Parallel engine is essential for large-scale data warehouse and is becoming popular nowadays. Thus combining the above two key trends, namely, parallel database processing on PC cluster would be a most cost-effective solution for large scale data warehousing.

We have built 100 Node PC cluster system named NEDO-100 for data base applications. We implemented parallel RDB kernel on it. TPC-D benchmark and association rule mining were run on the machine [6,16] and, it showed sufficiently high performance. We exemplified that the PC cluster can achieve considerably high cost-performance ratio.

We implemented high performance association rule mining system on the PC cluster. We have enhanced it with optimization to mine generalized association rule. In generalized association rules, application-specific knowledge in the form of taxonomies (*is-a* hierarchies) over items are used to discover more interesting rules. We introduce new parallel mining algorithms by taking the classification hierarchy into account[15]. Here we show that our system can handle large amount of transactions(1GBytes). [13]

One problem we faced in that project is "heterogeneity." The system we built[6,16] was completely uniform. However, when we planned to increase the number of nodes, it was extremely difficult to find out the same machines. Since the development period of PC is very short, configuration of machines is changing so quickly. Once six months have passed, we have to introduce different type of PCs. Thus heterogeneity is inevitable.

Most of the parallel algorithms developed so far assume the system be uniform. Very few papers address heterogeneity problem[2]. If we apply the parallel algorithm developed for uniform parallel system to the heterogeneous environment, apparently we will see significant performance deterioration. A high performance Node can process its allocated task quickly but Node with less powerful processor or with low bandwidth disk usually takes longer time to finish. We picked up data mining as a data intensive application and tried to solve the heterogeneity problem.

In [5] we propose run time load balancing algorithms for association rule mining under heterogeneous PC cluster environment. Two strategies named candidate migration and transaction migration are developed. Details on these two will be given in later sections. PCs do not have to communicate each other before the execution in order to normalize the performance among different CPUs and disks etc. During executing data mining, the workload of each Node is monitored autonomously and the system performance is controlled to be balanced by migrating candidates/transaction among nodes at runtime.

Section 2 explains the NEDO-100 PC cluster system. Section 3 briefly explains the association rule mining. Section 4 describes parallel algorithms for Apriori and for generalized association rules. Section 5 introduces the fundamental idea of load balancing for association rule mining. Section 6 discusses the future work and concludes the paper.

2 NEDO-100 PC Cluster

We have developed a large scale PC cluster consists of 128 PCs interconnected with 155 Mbps ATM and 10 Mbps Ethernet networks[6,16]. The project was launched in 1995 and the equipments came at the end of 1996. The system started at February 1997.

Initially the PC cluster was made up of 100 PCs with 200 MHz Pentium Pro only and then we have added another 8 nodes but with more powerful 333 MHz Pentium II and 20 nodes with 450 MHz Pentium II since the performance of PC hardware had improved dramatically.

The configuration of each PC showed in Table 1. The details of the development of this system has been written elsewhere. [6,16] We have performed

Table 1. Configuration of each PC

Node#	$1 \sim 100$	$101 \sim 119$	$120 \sim 127$
CPU	Pentium Pro 200MHz	PentiumII 450 Mhz	PentiumII 333 MHz
Main Memory	64MB	256MB	64MB
Disk Drive for databases	Seagate Barracuda (Ultra SCSI, 4.3GB) Seagate Cheetah (Ultra SCSI, 9.1GB)	IBM DTTA-371440 (EIDE, 14.4GB)	Seagate Cheetah (Ultra SCSI, 9.1GB)
ATM NIC	Interphase 5515 PCI ATM Adapter		
OS	Solaris2.5.1 for x86	Solaris2.6 for x86	Solaris2.6 for x86

numerous experiments of data intensive applications on the system that prove the applicability of this system. Remarkably, since the system use low-cost commodity PC, it offers extremely good cost performance compared to mainframe based database system currently in the market.

Some basic performance measurement of NEDO-100 are: point-to-point throughput using TCP/IP protocol over ATM exceeds 110Mbps with message size 8KB-16KB, roundtrip latency measured to be 448μs and disk read throughput about 8 MB per second.

3 Association Rule Mining

3.1 Association Rule

An example of an association rule is $!Hif$ a customer buys A and B then 90% of them buy also $C!I$. Here 90% is called the $confidence$ of the rule. Another measure of a rule is called the *support* of the rule that represents the percentage of transactions that contain the rule.

The problem of mining association rules is to find all the rules that satisfy a user-specified minimum support and minimum confidence, which can be decomposed into two subproblems:

1. Find all combinations of items, called large itemsets, whose support is greater than minimum support.
2. Use the large itemsets to generate the rules.

Here we briefly explain the Apriori algorithm for finding all large itemsets, proposed in [9].

In the first pass, support count for each item is incremented by scanning the transaction database. All items that satisfy the minimum support are picked out. These items are called large 1-itemset. Here k-itemset is defined as a set of k items. In the k-th pass, the candidate k-itemsets are generated using set of large $k-1$-itemsets. Then the support count of the candidate k-itemsets is incremented by scanning the transaction database. At the end of scanning the transaction data, the large k-itemsets which satisfy minimum support are determined. The process is repeated until no candidate itemsets generated.

3.2 Generalized Association Rule with Taxonomy

In most cases, items can be classified according to some kind of "is a" hierarchies. [11] For example "Sushi is a Japanese Food" and also "Sushi is a Food" can be expressed as taxonomy. Here we categorize sushi as descendant and Japanese food and food are its ancestors. By including taxonomy as application specific knowledge more interesting rules can be discovered.

Cumulate algorithm [11] is the first algorithm to mine generalized association rule mining. It is based on Apriori algorithm, and it is extended with optimizations that make use of the characteristics of generalized association rule such as pruning itemsets containing an item and its ancestors at second pass and pre-computing the ancestors of each item.

4 Highly Parallel Data Mining

4.1 Parallel Association Rule Mining

J.S.Park, et.al proposed bit vector filtering for association rule mining and naive parallelization of Apriori [10,8], where every Node keeps the whole candidate itemsets and scans the database independently. Communication is necessary only at the end of each pass. Although this method is very simple and communication overhead is very small, memory utilization efficiency is terribly bad. Since all the nodes have the copy of all the candidate itemsets, it wastes memory space a lot.

In [14] Hash Partitioned Apriori(HPA) was proposed in 1996. The candidate itemsets are not copied over all the nodes but are partitioned using hash function. Then each Node builds hash table of candidate itemsets. The number of itemsets at second pass is usually extremely high, sometimes three orders of magnitude larger than the first pass in a certain retail transaction database which we examined. When the user-specified support is low, the candidate itemsets overflow the memory space and incur a lot of disk I/O.

While reading transaction data for support counting, HPA applies the same hash function to decide where to send the transactions and then probe the hash table of candidate itemsets to increase the count. Although it has to exchange transaction data among nodes, utilization whole memory space through partitioning the candidates over nodes instead of duplication results in better parallelization gain.

Hybrid approach between candidate duplication and candidate partitioning is proposed at [4] at 1997. The processors are divided into some number of groups. Within each group, all the candidates are duplicated and among groups, candidates are partitioned.

4.2 Parallel Algorithms for Generalized Association Rule Mining

In this subsection, we describe our parallel algorithms for finding all large itemsets on shared-nothing environment proposed in [15].

Non Partitioned Generalized Association Rule Mining : NPGM NPGM copies the candidate itemsets over all the nodes. Each Node can work independently.

Hash Partitioned Generalized Association Rule Mining : HPGM HPGM partitions the candidate itemsets among the nodes using a hash function like in the hash join, which eliminate broadcasting.

Hierarchical HPGM : H-HPGM H-HPGM partitions the candidate itemsets among the nodes taking the classification hierarchy into account so that all the candidate itemsets whose root items are identical be allocated to the identical node, which eliminates communication of the ancestor items. Thus the communication overhead can be reduced significantly compared with original HPGM.

H-HPGM with Fine Grain Duplicate: H-HPGM-FGD In the case the size of the candidate itemsets is smaller than available system memory, H-HPGM-FGD utilizes the remaining free space. H-HPGM-FGD detects the frequently occurring itemsets which consists of the any level items. It duplicates them and their all ancestor itemsets over all the nodes and counts the support count locally for those itemsets like in NPGM.

4.3 Transaction Dataset

We use synthetic transaction data generated using procedure in [9]. For large scale experiments of generalized association rules we use the following parameters: (1) the number of items is 50,000, the number of roots is 100, the number of levels is 4–5, fanout is 5, (2)the total number of transactions is 20,000,000 (1GBytes), the average size of transactions is 5, and (3)the number of potentially large itemsets is 10,000.

4.4 Performance Evaluation Results

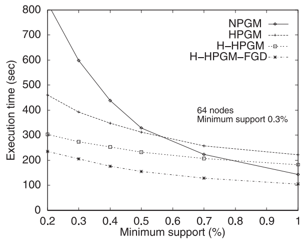


Fig. 1. Execution time

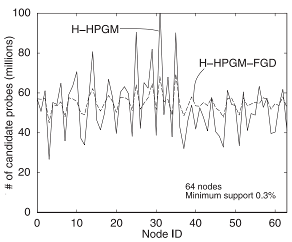


Fig. 2. Candidate probes

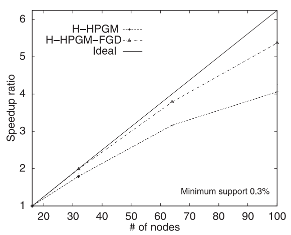


Fig. 3. Speedup ratio

We show the execution time at pass 2 of all parallel algorithms varying the minimum support in Figure 1. The execution time of all the algorithms increases when the minimum support becomes small. When the minimum support is small, the candidate partitioned methods can attain good performance. H-HPGM-FGD significantly outperforms other algorithms.

Next, the workload distribution of H-HPGM and H-HPGM-FGD is examined. Figure 2 shows the number of candidate probes to increment the support count in each Node at pass 2. In H-HPGM, the distribution of the number of probes is largely fractured, since the candidate itemsets are partitioned in the unit of hierarchy of the candidate itemsets. H-HPGM-FGD detects the frequently occurring candidate itemsets and duplicate them. The support counting process for these duplicated candidate itemsets can be locally processed, which can effectively balance the load among the nodes.

Figure 3 shows the speedup ratio with varying the number of nodes used 16, 32, 64 and 100. The curves are normalized by the execution time of 16 nodes system. H-HPGM-FGD attains higher linearity than H-HPGM. Since H-HPGM duplicates no candidate itemsets, the workload skew degrades the linearity. The skew handling methods detect the frequently occurring candidate itemsets and duplicate them so that the remaining free memory space can be utilized as much as possible. In Figure 3, H-HPGM-FGD achieves good performance on one hundred nodes system.

5 Dynamic Load Balancing on Heterogeneous PC Cluster

5.1 Run Time Load Balancing Methods

The parallel algorithms so far proposed assume homogeneous parallel processing environment. In [5], we propose dynamic load balancing algorithms for heterogeneous parallel systems, where each Node might have different type of CPU, and different kinds of disks, etc. We choose "flat" association rule mining based on HPA rather than generalized one as the application to give clearer insight on how they work.

As described in the section 4, HPA sends each node the itemsets and probes them against its own candidate itemsets hash table. [14] If a Node is assigned more candidate itemsets, it will receive more itemsets from other nodes during counting phase. This means that we can adjust the workload of each Node by adjusting the amount of candidate itemsets. If the load of a certain Node is higher than the other nodes, we take some of the candidate itemsets from that Node and give them to the other nodes. Then the itemsets that are originally directed to that Node are now redirected to the new nodes to which the removed itemsets are relocated. Thus the counting workload is migrated from the original Node to the other nodes. We name this strategy Candidate Migration.

The Candidate Migration is possible if the Node still has candidate itemsets to be migrated. itemsets. If the skew is high, there arises the case where migrating all the candidates is still not sufficient. In order to handle such situation, we need yet another strategy to migrate workload.

Let's examine the HPA algorithm again in more detail. Each node has two major task. One is to receive the itemset sent from other nodes, probe it against the hash table and increment the count corresponding to that itemset. The other task is to read the transactions from the disk, generate the itemsets and send them to the nodes determined using hash function. We use the former task for Candidate Migration.

Now we consider the latter task. Actually, the itemset generation from transactions is rather complicated process. This workload could be migrated. The Node with heavy workload reads the transactions from the disk and it does not do itemset generation itself but just sends the transactions to the light nodes. We name this strategy Transaction Migration.

Transaction Migration incurs overhead of network transfer for each transaction. Thus, we put priority to the Candidate Migration. Initially heavy Node migrates candidate itemsets only. When there are no candidate itemsets remained to migrate, then it migrates transactions.

5.2 Migration Plan Derivation

We propose dynamic load balancing methods during the execution of data mining to cope with the skew in heterogeneous system. In this approach, a coordinator Node collects necessary information from all the nodes, calculate estimated remaining processing time for that Node restT_i and controls the distribution of workload.

Since the goal is to have all nodes complete their job at the same time, our method dynamically controls the load allocated for each Node so that every Node has the same restT_i.

The skew is defined as follow,

$$skew = \frac{\max(\text{rest}T_i) - \min(\text{rest}T_i)}{\text{avg}(\text{rest}T_i)} \tag{1}$$

$$\begin{cases} skew \leq threshold : no\ skew \\ skew > threshold : skew\ exists \end{cases} \tag{2}$$

Coordinator can judge that the load control is needed if this value exceeds some certain threshold. Then it makes a plan for Candidate Migration If skew still presents, it also creates another plan for Transaction Migration. The above procedure is periodically invoked. Coordinator checks the skew condition every fixed time interval. The complete load balancing is difficult by any means. Error gradually accumulates. Once it becomes beyond the threshold, the coordinator tries to balance the workload again.

5.3 Experimental Environment and Transaction Dataset

In order to simplify the problem and to show clearly the effectiveness of our load balancing approach on heterogenous environment, we have made performance evaluation on a group of four nodes each with different CPU power, disk performance and data distribution as shown in table 2. The parameters used are described in table 3. In practice, we are forced to mine database in various situation, so data distribution is skewed. We put least amount of data to Node 4 while employing fast microprocessor in order to artificially generate skew. Apparently experiment with dataset 2 has higher skew than that with dataset 1. This is used for transaction migration experiments. And in all of the experiments, the *skew* value was set to 0.2.

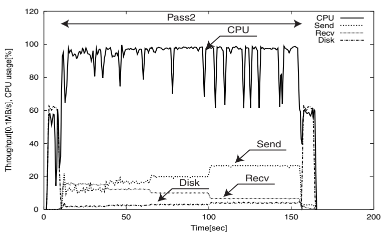
Table 2. Configuration for heterogenous experiments

	Node 1	Node 2	Node 3	Node 4
Proc.	P.Pro	P.Pro	P.Pro	P.II
Clock	200MHz	200MHz	200MHz	333MHz
Disk	SCSI	SCSI	SCSI	IDE
DataSet1	40MB	20MB	10MB	10MB
DataSet2	80MB	20MB	10MB	10MB

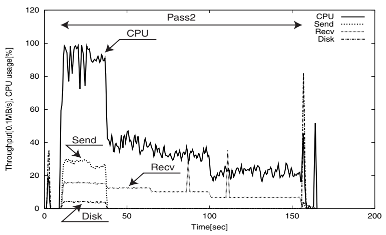
Table 3. Datasets for heterogenous experiments

	DataSet1	DataSet2
Number of transactions	1000000	1500000
Avg. size of transactions	20	20
Number of items	5000	5000

5.4 Performance Evaluation Results

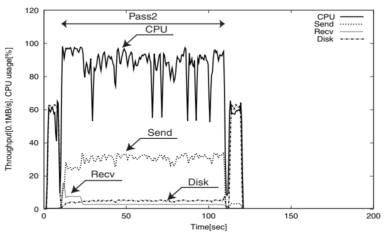


(a)Node1

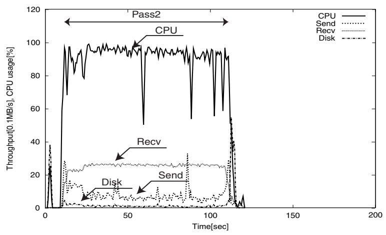


(b)Node4

Fig. 4. Execution trace without load balancing (DataSet1)



(a)Node1



(b)Node4

Fig. 5. Execution trace with Candidate Migration (DataSet1)

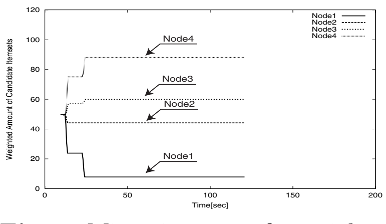


Fig. 6. Migration trace for weighted amount of candidate itemsets(DataSet1)

	DataSet1		DataSet2	
	C	L	C	L
Pass 1	5000	989	5000	982
Pass 2	488566	54	481671	51
Pass 3	42	4	38	4
Pass 4	0	0	0	0

Table 4. Number of candidate itemset and large itemset for DataSet1 and DataSet2

Heterogenous Configuration Experiment with Dataset 1 for Candidate Migration The numbers of candidate itemsets(C) and large itemsets(L) resulted from data mining of dataset 1 with 0.7% minimum support are shown in table 4. The execution traces without any load migration are shown in Figure 4. The figure shows four different resource usage: CPU, disk, interconnection network (send/receive). Horizontal axis is elapsed time and vertical axis denotes utilization ratio for CPU and data transfer throughput for disk read and interconnection network. The network throughput is divided into two parts, send throughput and receive throughput.

We only show traces for Node 1 and Node 4 since the space is limited. In the first half of second pass Node 1 is too busy receiving k-itemsets from other nodes, and could not even afford to read its own transaction data. On the other hand, Node 4 with more powerful CPU and less data finishes reading its data in first 40 seconds and idles for the rest of time. The total execution time is 164.03 seconds.

When we apply Candidate Migration strategy, candidate itemsets are reallocated as soon as skew is detected. The traces are shown in Figure 5. Every Node completes its task at almost the same time indicating the skew is eliminated and workload is evenly distributed. The processing time is also greatly improved to only 120.21 seconds.

Figure 6 shows the trace of weighted candidate itemsets of all the nodes. We can see that Node 1 and Node 2 migrate their candidate itemsets to Node 3 and Node 4. The amount of migrated itemsets gradually increases and finally converged to a certain value.

Heterogenous Configuration Experiment with Dataset 2 for Both Candidate Migration and Transaction Migration We did an experiment with more skewed environment using dataset 2. Result of data mining using dataset 2 and 0.7% minimum support is also shown in table 3. Node 1 is becoming the bottleneck of the parallelization as shown in Figure 7. The total execution time is 287.09 seconds.

By introducing the Candidate Migration, performance can be improved. The processing time is reduced to 198.36 seconds. However since the load is extremely concentrated at Node 1, as Figure 8 shows, Candidate Migration alone can not get rid of that skew completely. Node 4 finishes reading out the transactions from disk at around 125 seconds. We can see that all candidate itemsets of Node 1 has been transferred to other nodes, as shown at Figure 11. Thus Candidate Migration can not migrate workload any more.

When we introduce Transaction Migration in addition to Candidate Migration, we can achive almost perfect load balancing as shown at Figure 9. Node 1 sends its transaction data and delegates the generation of k-itemsets to other nodes. The elimination of skew records processing time of 182.18 seconds.

Figure 10 shows the trace of amount of weighted candidate itemset and amount of migrated transactions for Node 1 and Node 4. No candidate itemsets is left at Node 1. Node 1 also sends out transactions to the other nodes and Node 4 receives some of the transactions from Node 1.

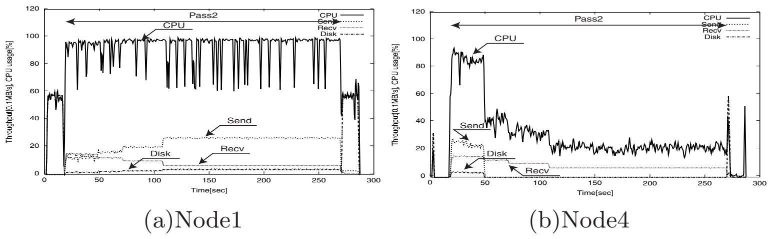


Fig. 7. Execution trace without any load balancing(DataSet2)

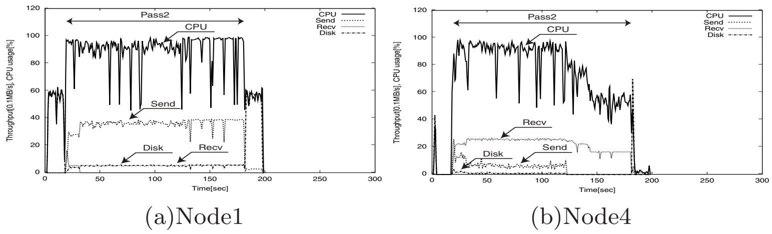


Fig. 8. Execution trace with Candidate Migration(DataSet2)

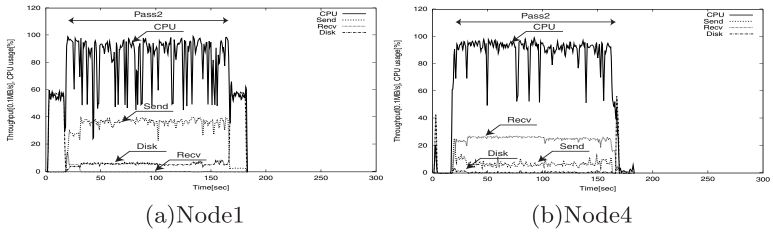


Fig. 9. Execution trace with both Candidate and Transaction Migration(DataSet2)

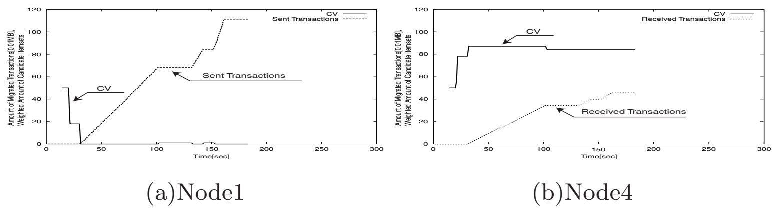


Fig. 10. Migration trace of weighted candidate itemsets and transactions(DataSet2)

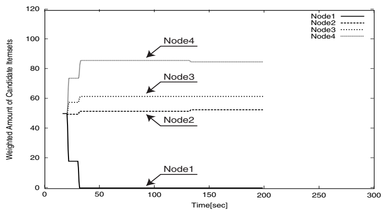


Fig. 11. Migration trace for weighted candidate itemsets (DataSet2)

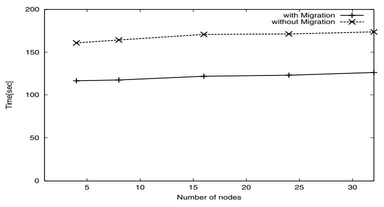


Fig. 12. Scale-up results(DataSet1)

Experiment for Scalability of Proposed Load Balancing Method We scaled up the system by multiplying the configuration we used so far. We used the configuration of a group of 4 nodes as multiplication unit and expanded the system from 4 nodes to 8, 12, 16, 24 and 32 nodes.

The results are shown in Figure 12. Execution time increases slightly as the number of nodes increases. As the number of nodes increases, the overhead time for synchronization becomes non-negligible.

6 Conclusion

We examined the effectiveness of parallel algorithms on large scale parallel computer system using the large amount of transaction dataset. Our system is consisted with one hundred of PCs. As far as the authors know, there has no research on parallel data mining over such large scale systems using a large amount of transaction dataset. Through several experiments, we showed H-HPGM-FGD could attain sufficiently high performance and achieve good workload distribution on one hundred PC cluster system.

We proposed dynamic load balancing strategies for parallel association rule mining on heterogeneous PC cluster system. Due to the short development period of recent PCs, it is inevitable that the PC cluster system becomes heterogeneous. In order to utilize all the system resources as fully as possible, we have to make the program adaptive to its runtime environment.

We adopted HPA(Hash Partitioned Apriori) algorithm for underlying parallel association rule mining. We proposed two kinds of dynamic load balancing strategies for parallel association rule mining, Candidate Migration and Transaction Migration.

In order to clearly show the effectiveness of our approach, we set up rather simple 4 Node cluster with two kinds of PCs and varied the size of dataset for each PC. We demonstrated the feasibility of our approach showing the execution trace. By examining the trace, we confirmed that the proposed scheme effectively works to remove workload inbalance. Candidate Migration works under medium skew environment. If the skew is high and candidate migration can not sufficiently help, the system automatically invokes the Transaction Migration. In addition, we also showed the scalability experiments. We increased the size of the system from 4 nodes to 32 nodes. We found sufficient scalability can be archived.

Our experiments have showed that PC cluster, with its scalable performance and high cost performance is a promising platform for data intensive applications such as data mining.

References

1. D. W. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. Fu. "A Fast Distributed Algorithms for Mining Association Rules." In *Proc. of PDIS*, pp. 31–42, Dec. 1996.
2. H. M. Dewan, M. A. Hernandez, K. W. Mok, S. J.Stolfo "Predictive Dynamic Load Balancing of Parallel Hash-Joins Over Heterogeneous Processors in the Presence of Data Skew." In *Proc. of PDIS*, pp. 40–49, 1994. 16
3. D. DeWitt and J. Gray "Parallel Database Systems: The Future of High Performance Database Systems." In *Communications of the ACM*, Vol. 35, No. 6, pp. 85–98, Jun. 1992. 15
4. E.-H.Han and G.Karypis and Vipin Kumar "Scalable Parallel Data Mining for Association Rules." In *Proc. of SIGMOD*, pp. 277–288, May. 1997 18
5. M. Tamura, M.Kitsuregawa. "Dynamic Load Balancing for Parallel Association Rule Mining on Heterogeneous PC Cluster System". In *Proc. of VLDB*, 1999. 16, 20
6. M. Kitsuregawa, T. Tamura, M. Oguchi "Parallel Database Processing/Data Mining on Large Scale ATM Connected PC Cluster." In *Euro-PDS*, pp. 313–320, Jun. 1997 15, 16
7. M. J.Zaki, S.Parthasarathy, M.Ogihara and W.Li "Parallel Algorithms for Discovery of Association Rules". Data Mining and Knowledge Discovery, Dec. 1997.
8. J. S.Park, M.-S.Chen, P. S.Yu "Efficient Parallel Algorithms for Mining Association Rules" In *Proc. of CIKM*, pp. 31–36, Nov. 1995 18
9. R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules". In *Proc. of VLDB* , pp. 487–499, Sep. 1994. 17, 19
10. R. Agrawal and J. C. Shafer. "Parallel Mining of Associaton Rules". In *IEEE TKDE*, Vol. 8, No. 6, pp. 962–969, Dec. 1996. 18
11. R. Srikant, R. Agrawal. "Mining Generalized Association Rules". In *Proc. of VLDB*, 1995. 18
12. S.Parthasarathy and M. J.Zaki and W.Li "Memory Placement Techniques for Parallel Association Mining." In *Proc. of KDD*, pp. 304–308, Aug. 1998
13. T.Shintani, M. Oguchi, M.Kitsuregawa. "Performance Analysis for Parallel Generalized Association Rule Mining on a Large Scale PC Cluster". In *Proc. of Euro-par*, 1999. 16
14. T. Shintani and M. Kitsuregawa "Hash Based Parallel Algorithms for Mining Association Rules". In *Proc. of PDIS*, pp. 19–30, Dec. 1996. 18, 20
15. T. Shintani, M. Kitsuregawa "Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy." In *Proc. of SIGMOD*, pp. 25–36, 1998. 16, 18
16. T. Tamura, M. Oguchi, M. Kitsuregawa "Parallel Database Processing on a 100 Node PC Cluster: Cases for Decision Support Query Processing and Data Mining." In *Super Computing 97::High Performance Networking and Computing*, 1997 15, 16
17. Y. Xiao and D. W. Cheung "Effect of Data Skewness in Parallel Data Mining of Association Rules ". In *Proc. of PAKDD*, pp. 48–60, Apr. 1998.

XML Query Languages in Practice: An Evaluation

Zachary G. Ives[1] and Ying Lu[2]

[1] University of Washington, Box 352350, Seattle WA 98195-2350, USA
zives@cs.washington.edu

[2] University of Wisconsin, 1210 W. Dayton St., Madison WI 53706, USA
luy@cs.wisc.edu

Abstract. The popularity of XML as a data representation format has led to significant interest in querying XML documents. Although a "universal" query language is still being designed, two language proposals, XQL and XML-QL, are being implemented and applied. Experience with these early implementations and applications has been instructive in determining the requirements of an XML query language. In this paper, we discuss issues in attempting to query XML, analyze the strengths and weaknesses of current approaches, and propose a number of extensions. We hope that this will be helpful both in forming the upcoming XML Query language standard and in supplementing existing languages.

1 Introduction

With the advent of the Internet and World Wide Web as mediums for electronic commerce and information exchange, the eXtensible Markup Language, XML, has emerged as a topic of great interest in the database community. XML provides a universal format for essentially any type of data, and it is rapidly being adopted as a replacement for proprietary formats in many applications. Shortly after XML's emergence, a number of researchers identified the need for a query language over this data representation format. The result has been a series of proposed languages, the most prominent of which are XML-QL [10] and XQL [17]. In an effort to provide a standard, the World Wide Web Consortium is developing a language called XML Query, for which they hope to have a working draft sometime this year; it is expected to be primarily derived from XML-QL and XQL.

Meanwhile, market demand for storing and querying XML has encouraged companies to develop and market commercial query processors for XML today. Object Design's eXcelon XML repository [18] maps XML documents to objects and supports queries over them based on the XQL language. Both Oracle and IBM support XML export of query results from their relational database systems. In these products and projects, XML is generally treated as a *protocol* for describing relational or object-based information — the focus is on encapsulating traditional data in an XML container, rather than on the unique representational aspects implicit in XML. By contrast, a number of research projects [16,2,12,13]

```
<db>
 <lab ID="baselab" manager="smith1">
  <name>Seattle Bio Lab</name>
  <location>
    <city>Seattle</city>
    <country>USA</country>
  </location>
 </lab>
 <lab ID="lab2">
  <name>Philadelphia Lab</name>
  <location>
    <city>Philadelphia</city>
    <country>USA</country>
  </location>
 </lab>
 <paper ID="Smith991231"
                   source="baselab">
  <title>Automatic Record...</title>
  <biologist>smith1</biologist>
 </paper>
 <biologist ID="smith1">
   <lastname>Smith</lastname>
 </biologist>
</db>
```

Fig. 1. Sample XML document representing biology labs and publications

utilize XML as the basis for a data model, exploring the storage, integration, and processing of tree- and graph-structured XML data. A third perspective is that of users attempting to exploit XML in custom application domains, to obtain functionality not provided by more traditional database systems. One such project is the Cell Systems Initiative [8], an effort to define an ontology and experiment capture system for cellular research. Data in this application is too complex in structure to be effectively stored in a relational system, and requires a semi-structured data model and query language, such as those for XML.

Each of these domains has different needs for querying XML from a data management perspective. One sees XML as a standard exchange protocol; another uses XML as an intrinsic data model; the final applies XML to real-world application domains that would otherwise be unmanageable. In this paper, we discuss the strengths and weaknesses of the query languages applied to these domains, and we propose a number of improvements. Our goal is to present not simply a survey of language features, but an evaluation of how useful these languages are for data management. We do not attempt to address information-retrieval-style queries, which have been the focus of work such as [11]. We hope that our analysis will be useful in defining the standard XML Query language and for extending existing languages in the interim.

The structure of this paper is as follows. We begin in Section 2 with the basics of XML and how it is modeled, then continue in Section 3 with an overview of XML query languages (focusing on XQL and XML-QL). Section 4 describes issues in querying XML trees and graphs. In Section 5, we examine how input data is restructured into an output document. Section 6 discusses the application of XML models and query languages to the Cell Systems Initiative domain, and what difficulties this presents. In Section 7, we discuss related work and conclude with some recommendations.

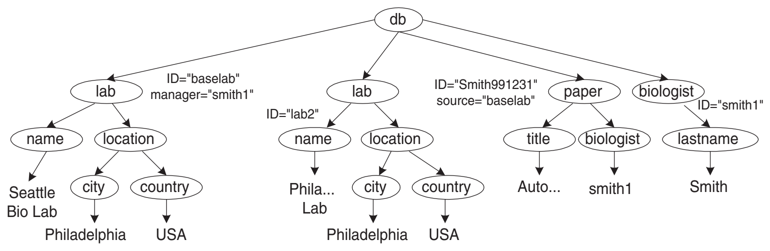


Fig. 2. XQL representation for Fig. 1. Edges indicate subelements.

2 XML Data

An XML document consists of pairs of matching open- and close-tags (elements), each of which may enclose additional elements or data values (in the form of "character data" strings). Every document must be contained within a single set of enclosing tags, known as the *root element*. Additionally, an element tag may include single-valued attributes further describing the element. XML documents may include embedded references to other XML documents in the form of XPointers [9]. See Figure 1 for the sample XML document upon which we shall frequently base our examples throughout this paper (note that it has no XPointers).

An optional companion to the XML document, the Document Type Descriptor or DTD, adds a "schema" to which an XML document must conform to be considered *valid*. The DTD constrains the nesting of elements and assigns typing information to attributes. Attributes of type `ID` are element identifiers; those of type `IDREF` or `IDREFS` are references to other elements, by `ID`, within the XML document. `ID`s are guaranteed to be unique within a document and `IDREF`s may not dangle. For the example of Figure 1, we shall assume that an associated DTD (not shown) defines the `ID` attribute of the various elements to be of type `ID`, the `manager` attribute of the `lab` element to be an `IDREF`, and the `paper` element's `source` to also be a reference.

2.1 XML Data Models

One of the most important differences between XML query languages is in their data models. We begin here with an overview of the approaches to modeling an XML document.

Tree Models XQL [17] and XMAS [14] use the XML Document Object Model (DOM [1]) tree as the basis of their data models (Figure 2). This parse tree represents elements as nodes, contained subelements as edges to nodes, and all attributes as fields accessible from their elements.

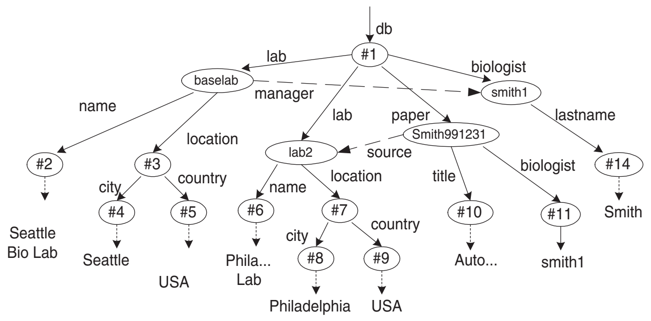


Fig. 3. XML-QL graph for Fig. 1. Dashed edges are `IDREF`s; dotted edges are `PCDATA`.

Graph Models Languages such as XML-QL [10], XML-GL [5], and Lorel [12] treat the XML input document as a graph (Figure 3), where both subelements and `IDREF`s are mapped to edges. Each element is represented as an edge (labeled with the element name) directed to a node (given the element's `ID` if one exists, otherwise a unique identifier). `IDREF` edges are labeled with the `IDREF` name, directed to the referenced element node. In order to allow for intermixing of string data and nested elements within each element, XML-QL creates a `PCDATA` edge to each string. Conventional attributes (not shown in the example) are fields accessible from their element nodes.

The use of `IDREF`s as graph edges allows for modeling of any arbitrary structured or semi-structured data. However, current graph data models do not fully specify ordering within the data graph; to clarify order mappings, we propose the following correspondence. Given an XML data graph, we can take all `IDREF` edges and replace them with identically-named attributes, whose values are the `ID`s of the destination nodes; this will result in a tree equivalent to the DOM tree. A left-to-right depth-first traversal will generate the equivalent XML document. In this mapping, attributes and `IDREF` edges are ordered *before* subelement edges.

2.2 Traversing the Document

The key to querying an XML document lies in selecting the desired data from the input. Most XML query languages use *regular path expressions*, describing paths to be taken from the document "root node" to the data values. A regular path expression enumerates a sequence of node or edge labels to be followed; since XML may have recursively nested elements or irregular structure, it may include regular-expression operators such as the wildcard, the Kleene-star (for repetition), and choice (for alternate sub-paths).

For a tree-structured query language, there is one unique path from the root to a given node. If the data model is graph-structured, however, there may be multiple paths to a given node; here, language semantics generally specify that a path expression will only return each node once.

Even under the graph-structured data model, there are cases where we would like to traverse only subelements in the document, or to query `IDREF`s as attributes instead of edges. XML-QL does not differentiate between these different edge types; the Lorel [12] query language supports this by allowing the user to switch between a graph-structured and a tree-structured mode. We believe that Lorel's modes can be too coarse-grained — the query author may want to maintain the input graph structure, but simply traverse subelements along certain portions of the path. We suggest extending path expressions so they can restrict the type of a given traversal edge to be an `IDREF` or a subelement.

3 Query Language Basics

A number of query languages have characteristics of note. XMAS [14], the language of the MIX mediator system, is essentially a simplification of XML-QL for tree-structured data; its simplified model makes query processing and writing less complex. Lorel, the language for the Lore semi-structured database system, has been extended to support XML, and includes update as well as query capabilities (though these are only supported in Lore's original OEM model). Finally, XML-GL [5] uses diagrams rather than commands to express queries. While these languages have some novel features, we focus on XQL [17,19] and XML-QL [10], the languages that are being implemented for real applications and the greatest influences on the W3C query language specification.

3.1 XQL

XQL is the "parent" of the W3C XPath [6] document navigation standard and the basis of the XSLT [20] transformation standard. However, XQL was recently extended with features not found in its "child"; we shall focus on XQL rather than XPath in this paper. XQL queries are very simple: they extract nodes and subtrees from a single input document and return these in a new XML document.

The XQL tree data model is closely matched to the XML physical format, as described in Section 2.1. Queries return elements and their children in the same order as they appear in the input document. This model makes XQL well suited to finding and returning XML document fragments; yet the data model is considerably less flexible and expressive than a graph model.

An XQL query is divided into a *path expression* and an optional *filter expression*. The path expression defines the nodes to be returned in the query result, and the filter expression selects only nodes meeting specified criteria. A query is of the form *path* `[`*filter*`]`, where *path* is typically a series of element names separated by the slash (`/`) character, and *filter* is a sequence of boolean conditionals, e.g. tests for sub-path existence. A sub-path is a sequence of element names, optionally followed by an attribute (indicated by a prepended `@` sign). XQL paths and sub-paths are expressed relative to a "current location" (defaulting to the entire document for the outermost query). A leading slash restarts the path at the document level; a star (`*`) is a wildcard representing any edge; two

```
WHERE <db>                        CONSTRUCT <result><combo>
        <lab><name>$l</></>                  <laboratory>$l</>
        <biologist>$b</>                     <person>$b</>
      </> IN "fig1.xml"                    </></>
```

Fig. 4. Example XML-QL query for Figure 1

consecutive slashes (//) specify any number of wildcard edge traversals. Once a node has been selected by its path, it can be returned, or various methods can be called on it. Note that an XQL path expression is not as powerful as a full regular path expression, as it does not include true Kleene-star or choice operators.

Example XQL queries over Figure 1 include:

– `/db/lab[@manager="smith1"]` returns only `labs` with `manager` attributes containing the value "`smith1`" — namely the Seattle lab. Note that we return the `lab` subelement and all of its children; the filter within the brackets does not affect the query path.
– `/db/lab` { `location/city | name` } returns each `lab` element, with only its `city` and `name` subelements within. The { } are "grouping" operators and `|` forms a union of element results.

3.2 XML-QL

XML-QL was the first database-style language proposed for querying XML, and has had perhaps the largest impact on the W3C's vision of an "ideal" XML query language. XML-QL uses the graph data model of Section 2.1, and is a full graph-to-graph query and transformation language.

XML-QL uses a `WHERE` *pattern1* `IN` *source1*, ... `CONSTRUCT` *result* syntax, in which each *pattern* template is matched against an input XML data graph from its corresponding *source* (a URI, view, or variable) and the *result* defines the desired structure of the query output graph. XML-QL supports multi-document queries and relational-like operations such as joins and grouping.

An XML-QL pattern is expressed as a set of nested tags with embedded variable names (prefixed by leading dollar-signs) that specify *bindings* of graph nodes to variables. An example XML-QL query appears in Figure 4. We abbreviate each close-tag with a `</>`. The WHERE template is a tree structure of path expressions that get "matched" across the input graph. Each variable (l and `b` above) is bound to the matching node at the end of the path. In this case, we take a `db` edge from the document root. From here, we find a `lab` edge and then a `name` edge to a node we assign to variable l. Now, from the same `db` edge traversed earlier, we find a `biologist` edge to a node we shall call `b`. During query execution, we apply the template and form every possible combination of path expression matches. Each tuple of bindings to variables is evaluated much like a tuple in a relational database.

```
/db/paper[$b=biologist] {
   * | /db/biologist[@ID=$b]
                /lastname
}
```

(a) XQL

```
WHERE <db><paper>
         <biologist>$b</>
       </> CONTENT_AS $p
      </> IN "fig1.xml",
      <db><biologist ID=$b>
         <lastname>$n</></>
      </> IN "fig1.xml"
CONSTRUCT
     <result><paper>$p<lastname>$n</>
     </></>
```

(b) XML-QL

Fig. 5. Join query for data of Figure 1

The CONSTRUCT clause specifies a tree structure to add to the output graph. Wherever an input variable appears in the CONSTRUCT clause, its associated node is inserted into the output. We also "carry forward" all other nodes transitively connected by edges radiating from the original node. In essence, an XML-QL variable bound to an XML graph node represents the *entire subgraph* to which the node transitively connects via "forward-pointing" edges.

4 Querying XML Data

In this section, we examine some of the important considerations in querying XML documents.

4.1 SQL-Like Features

With the simple language elements discussed previously, we can express search-style queries that traverse an XML document and return portions. However, one of the applications of XML is as a "container" for relational data. In this type of application, the query language must support SQL-like relational operations.

Join XQL can join different path expressions within the *same* document. In order to support join predicates, dollar-sign-prefixed *correlation variables* are associated with sub-paths. Figure 5(a) takes `paper biologist` subelements and saves their values in the `b` variable; then finds `db biologist` elements with matching `ID`s. The returned result is the `paper` elements with additional `biologist lastname` subelements inserted within. Note that this is a left outer join, as papers may appear with no biologists. XQL also supports a limited inner join operation that returns the contents of *one* of the two join subtrees.

The XML-QL equivalent to Figure 5(a), in Figure 5(b), is slightly more powerful, as it can combine the `paper` and `author` elements from different sources. If the same variable occurs more than once within a query, it is constrained to have the same value in both places (thus forming an equijoin); or we can add a test such as `$b1 < $b2` to the `WHERE` clause to establish a range constraint on the variables' values. The `CONTENT_AS` specifier in the query expresses a binding of a variable to the contents (the node) of the previous element — in this case, it sets `p` equal to the `paper` node. The constructed output consists of `paper` elements with additional `biologist lastname` subelements, as in the XQL query, except that we have performed an inner join. XML-QL also supports outer join queries, but these are more complex to express and less commonly used.

Null values A problem can arise when mapping relational tables with null attributes to XML. Typically, the subelements corresponding to null relational attributes are simply omitted from the document. Neither XQL nor XML-QL support queries with optional elements that are not required in matching a pattern. If XML is to be effectively used to store relational data, it seems critical that the query language have this capability.

Universal quantification and negation Two important capabilities for certain classes of queries are universal quantification and negation. XQL includes `all` and `not` keywords in the filter expression for this purpose; XML-QL is missing these important features.

Aggregation Aggregate functions such as `average` and `max` are very commonly used for summarization and other purposes in SQL. XQL supports a `count` function, but no other aggregation operations. The original XML-QL specification describes a model for supporting aggregate functions, and this model has been further developed (see Section 5.4).

4.2 References

When XML is used as a graph-structured data format, following references becomes vital. Both XQL and XML-QL include mechanisms for managing `IDREF`s. XML-QL models both `IDREF` and `IDREFS` attributes as edges, so we can follow references with path expressions. In the latest XQL proposals, a global method, `id`, dereferences `IDREF` attributes by value. Unfortunately, XML `IDREFS` attributes consist of a string values with *multiple* ID references separated by delimiter characters; XQL does not include a mechanism for separating these into sub-components, and thus there is no way to de-reference an `IDREFS` attribute.

Neither XML language handles XPointers, but they can fairly easily be extended to do so. A proposed extension to XQL adds a `ref` global method that returns the contents of an XPointer. In XML-QL, one can create a user-defined "function" that does the equivalent, namely takes a URI or XPointer and returns the corresponding XML graph.

4.3 Querying Document Order

Both XQL and XML-QL allow queries to reference a subelement's *index*, i.e. its numeric ordering as a child of its parent element: we can specifically request the *i*-th child of an element, or we can query a node for its index value. However, XML-QL lacks the ability to query for the *i*-th element with label *L* (e.g. the second `<p>` element), which we can do in XQL.

In Section 2.1, we proposed a mapping between an XML graph model and document in which `IDREF`and `IDREFS` edges are ordered prior to sibling subelements. Under this ordering, we can now extend XML-QL in a useful way for graph data — to allow querying for the *i*-th *out-edge* with a particular name. (Note that we do not propose an ordering between edges of different names, since different `IDREF` attributes are unordered with respect to one another.)

5 Special Query Output Behavior

In the previous section, we analyzed the capabilities of the XML query languages on input documents. Both languages support simple "copying" of input subtrees to the output. In this section we discuss how portions of this output can be modified, and also discuss how graph-structured XML can be created in XML-QL.

5.1 Pruning XML Output

An interesting aspect of nearly all XML query languages, including XQL and XML-QL, is that they allow for the selection of a portion of the input document (subtree or subgraph) via a path expression, but support no projection-like operations on it. An XQL path expression will return the matching subtree, however deep it might be, as output. An XML-QL node variable, when it is used in the CONSTRUCT clause, outputs the node plus the entire subgraph to which it is connected. A missing capability would allow the query to restrict the portions of the subtree or subgraph that get copied, i.e. prune the data.

5.2 Modifying Elements

Often, queries need to rename the element labels from the original query. Examples of how to do this in XQL and XQL are in Figure 6, where we rename the `lab` element to `smithlab`. In the XQL query, we use the `->` renaming operator to change the name of the outermost tag of the query result; in XML-QL, we bind to the source element's node content using the `CONTENT_AS` specifier, and later "wrap" the node with a new enclosing tag.

5.3 Nesting Subqueries

XML is fundamentally a tree-structured format, so one of the most common operations is to take elements from one subquery and nest them under elements from a different query (perhaps "matching" parent and child query results with a join condition). Both XQL and XML-QL support this operation quite elegantly: essentially, the subquery is embedded within the portion of the parent query that

```
/db/lab[@manager="smith1"]
                 -> smithlab
```

(a) XQL

```
WHERE <db>
        <lab manager="smith1"></>
                   CONTENT_AS $lab
      </> IN "fig1.xml"
CONSTRUCT
      <result><smithlab>$lab</></>
```

(b) XML-QL

Fig. 6. Renaming an element

```
WHERE <db>
        <paper>
          <biologist>$b</>
        </> CONTENT_AS $p
      </> IN "fig1.xml"
CONSTRUCT
      <db>
        <paper>$p
```

```
  { WHERE <db>
            <biologist ID=$b>
              <lastname>$l</>
            </>
          </> IN "fig1.xml"
    CONSTRUCT <lastname>$l</>
  }
  </>
</>
```

Fig. 7. Nesting in XML-QL

constructs the result. The XQL query of Figure 5(a) outputs `paper`s as parents of a nested `biologist` subquery; its XML-QL equivalent appears in Figure 7. For both languages, the entire subquery is executed and embedded for each set of bindings in the outer query, producing a *1:n* nesting relationship.

5.4 XML Graphs in XML-QL

XQL is a single-document, tree-oriented language, whereas XML-QL is multi-document and graph-oriented. The extra features of XML-QL provide considerably more flexibility, but also add new concerns with respect to the output XML representation. Note that several people have proposed extending XQL to a graph model, so while this discussion is focused on XML-QL, it will also be relevant to such extended versions of XQL.

Skolem Functions A fundamental concept in XML-QL is that of node identity in the output graph. If a query attempts to create an output node with the same node ID more than once, i.e. for more than one tuple of variable bindings, each iteration refers to the same node in the output graph — the output node will only be created once. This enables a query to refer to and extend an existing node. This is where the XML-QL *Skolem function* is useful. Each Skolem function

```
WHERE <db>                              CONSTRUCT
        <paper ID=$i source=$s>           <result>
          <title>$t</>                      <biologist ID=Sk1($b)>
          <biologist>$b</>                    <paper ID=$i source=$s
        </>                                       ref=Sk1($b)>$t</>
      </> IN "labs.xml"                     </> </>
```

Fig. 8. Grouping with Skolem functions

creates a perfect hash value for its arguments, and its values will not collide with those of any other Skolem function.

Skolem functions are used to group elements based on data value and to create multiple references to the same node. For instance, in the query of Figure 8, we take any `paper` elements in Figure 1 and, using the Skolem function `Sk1`, re-group them by `biologist` instead of by paper. For each new value of `b`, we will output a `biologist` node and a nested `paper` node with a reference back to its parent. Each time a duplicate `b` value is bound, we insert a new `paper` node underneath the existing `biologist` node.

Skolem functions also form the basis of aggregation operations: functions such as `average` or `count` can be applied across the sets of values that get consolidated together by the Skolem function. Skolem functions can even perform duplicate removal and force XML-QL to output graph-structured rather than tree-structured data.

XML Graph Irregularities At times, graph-structured data may map into irregular and "ugly" XML. For example, it is possible to use Skolem functions to consolidate nodes such that they have multiple in-edges like the `Hyp2` node in Figure 9. The initial set of variable bindings will create the `Hyp2` node as an XML subelement under some parent element. For future bindings, however, referencing parent nodes must connect to `Hyp2` via `IDREF`s. Thus one of the node's "parents" will be a parent *element*, and all others will be *referrers* — despite the fact that the source query did not distinguish between any of the "parent" nodes.

Another XML mapping artifact arises because XML-QL outputs not only the nodes bound to variables, but also all nodes that are transitively connected to these. This feature is often convenient, but it has indeterminate output when the referenced element has a parent not in the query output. In this case, the most logical approach is to "fold" the referenced node under its first "parent" as an XML subelement.

6 An Application of XML

The goal of the Cell Systems Initiative (CSI) project at the University of Washington is to provide an online, web-like knowledge base representing all aspects

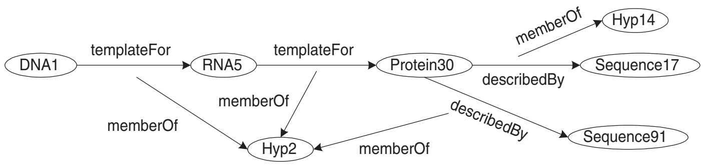


Fig. 9. Model of DNA-RNA-Protein interactions according to hypotheses

of biological data — from experiments to hypotheses to publications. The CSI knowledge base is a complex graph structure with edges, "conditional edges," and various types of nodes. An example graph appears in Figure 9. Note that the `templateFor` and `describedBy` edges indicate relationships, but that these relationships are conditional on the validity of the hypotheses, as expressed by the `memberOf` edges originating from these edges.

A graph like this can be represented relationally, or even in an XML tree model, but querying it would be highly unintuitive and inefficient. We chose to use the XML-QL graph model, after making one transformation: since XML does not allow edges to originate from other edges, we must "split" each conditional edge into a pair of edges with an intermediate node, providing a source for each `memberOf` edge.

Proposed queries for the CSI domain have demonstrated the need for a several extensions to XML-QL. A significant problem occurs because of XML-QL's policy of "carrying over" *all* transitively connected nodes; this may result in "extra" output. A pruning feature, as suggested in Section 5.1, would solve this problem. Additionally, the CSI database is expected to consist of large numbers of interlinked XML documents, each using XPointers to reference other portions of the overall structure. XML-QL must be able to support XPointers to make this work. Overall, however, preliminary designs and results suggest that, with these extensions, XML-QL is are fairly well suited to this application.

7 Related Work and Conclusions

In this paper, we have described the two most widely accepted XML query languages, XQL and XML-QL, and examined how they can be applied to three different domains: relational queries, queries over arbitrary XML data, and graph-structured scientific applications. While we believe this to be the first analysis of XML query languages' applicability, issues in designing an XML query language have been frequently discussed in the literature. In particular, the W3C's 1998 Query Language Workshop included numerous papers describing the the motivations and requirements for querying XML [4,7,15], as well as several important language proposals [17,10,12,5,14]. The application of XML-QL to information retrieval queries was discussed in [11].

Recently, Bonifati and Ceri presented a survey of five major XML query languages [3] that compared the features present in each. The goal of this paper is more than to provide a feature comparison: we hope to promote a greater understanding of XML query semantics, and to detail some of the problems encountered in trying to apply these languages. While a query language containing the "union" of the features present in XQL and XML-QL will go a large way towards solving the needs of querying XML, we also propose a number of extensions that we feel are necessary:

- An XML graph model with defined order between `IDREF`s and subelements
- Regular path expression extensions for subelement, `IDREF`, or arbitrary edges
- Support for "optional" path expression components and null values
- Support for following XPointers
- Pruning of query output
- Clearer semantics for copying subgraphs to query output

In general, XML-QL nearly meets our needs, and it could fairly naturally be extended with the missing capabilities. In particular, if we add universal quantification, negation, and the features listed above, it should be well-suited for our domains of interest. XQL can also be further developed, but the required extensions fit less cleanly into its single-document query model.

While some of these operations may increase the complexity of an XML query processor, all should be possible using well-studied techniques. XML querying is still a young field, but the database community's experience in querying other data models has given it a solid foundation.

Acknowledgements

The primary author was funded by an IBM Research Fellowship. The authors would like to thank Larry Arnstein for providing the sample of the CSI graph model, as well as Don Chamberlin, Michael Carey, Subbu Subramanian, Eugene Shekita, Alon Levy, and Dan Weld for a number of fruitful discussions that motivated this paper.

References

1. V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. L. Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood. Document object model (DOM) level 1 specification. http://www.w3.org/TR/REC-DOM-Level-1, October 1998. 31
2. C. K. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 597–599, 1999. 29
3. A. Bonifati and S. Ceri. Comparative analysis of five XML query languages. *SIGMOD Record*, 29(1):68–79, March 2000. 41

4. A. Bosworth, A. Levy, J. Widom, R. Goldman, J. McHugh, A. Layman, A. Ardelwanu, and D. Schach. Position paper for the W3C query language workshop, December 3, 1998. W3C Query Language Workshop, http://www.w3.org/TandS/QL/QL98/pp, December 1998. 40
5. S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A graphical language for querying and reshaping XML documents. W3C Query Language Workshop, http://www.w3.org/TandS/QL/QL98/pp/xml-gl.html, December 1998. 32, 33, 40
6. J. Clark and S. DeRose. XML path language (XPath) recommendation. http://www.w3.org/TR/1999/REC-xpath-19991116, November 1999. 33
7. P. Cotton and A. Malhotra. Candidate requirements for XML query. W3C Query Language Workshop, http://www.w3.org/TandS/QL/QL98/pp, December 1998. 40
8. Cell Systems Initiative. http://cellworks.washington.edu, 2000. 30
9. S. DeRose, R. D. Jr., and E. Maler. XML pointer language (XPointer) working draft. http://www.w3.org/TR/1999/WD-xptr-19991206, December 1999. 31
10. A. Deutsch, M. F. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language for XML. In *Proceedings of the International Word Wide Web Conference, Toronto, CA*, 1999. 29, 32, 33, 40
11. D. Florescu, D. Kossman, and I. Manolescu. Integrating keyword search into xml query processing. In *Proceedings of the 9th WWW Conference, Amsterdam, NL*, May 2000. 30, 40
12. R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the Lore data model and query language. In *ACM SIGMOD Workshop on the Web (WebDB), Philadelphia, PA*, pages 25–30, 1999. 29, 32, 33, 40
13. Z. G. Ives, A. Y. Levy, and D. S. Weld. Efficient evaluation of regular path expressions over streaming XML data. Submitted for publication, 2000. 29
14. B. Ludäscher, Y. Papakonstantinou, and P. Velikhov. A brief introduction to XMAS. http://www.db.ucsd.edu/Projects/MIX/docs/XMAS-intro.pdf, February 1999. 31, 33, 40
15. D. Maier. Database desiderata for an XML query language. W3C Query Language Workshop, http://www.w3.org/TandS/QL/QL98/pp/maier.html, December 1998. 40
16. J. Naughton, D. DeWitt, D. Maier, J. Chen, L. Galanis, K. Tufte, J. Kang, Q. Luo, N. Prakash, F. Tian, J. Shanmugasundaram, C. Zhang, R. Ramamurthy, B. Jackson, Y. Wang, A. Gupta, and R. Chen. The Niagara internet query system. Submitted for publication, 2000. 29
17. J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). http://www.w3.org/TandS/QL/QL98/pp/xql.html, September 1998. 29, 31, 33, 40
18. eXcelon: The XML application development environment. http://www.odi.com/excelon/main.htm. 29
19. XQL (XML Query Language). http://metalab.unc.edu/xql/xql-proposal.html, August 1999. 33
20. XSL Transformations (XSLT), version 1.0. http://www.w3.org/TR/xslt, 13 August 1999. W3C Working Draft. 33

A Two-Level Method for Clustering DTDs[1]

Weining Qian[1], Long Zhang[1], Yuqi Liang[1], Hailei Qian[1], and Wen Jin[2]

[1] Computer Science Department, Fudan University
200433 Shanghai, P. R. China
{wnqian,lzhang0,yqliang,hlqian}@fudan.edu.cn
[2] Computer Science Department, Simon Fraser University
8888 University Drive, Burnaby, B.C. Canada V5A 1S6
wjin@cs.sfu.ca

Abstract. XML is a standard that is widely applied in data representation and data exchange. However, as an important part of XML, DTD is not taken full advantage of in current applications. In this paper, a new method for clustering DTDs is presented, so that it can be used in XML document clustering. The two-level method clusters the elements in DTDs and DTDs separately. Element clustering forms the first level, and provides the element clusters, which is the generalization of relevant elements. DTD clustering utilizes the generalized information and forms the second level in the whole clustering process. The two-level method has the advantages that: 1) it takes into consideration both the content and the structure within the DTDs; 2) the generalized information about elements is more useful than the separated words in the vector model; 3) the two-level method facilitates the searching of outliers. The experiments show that this method is able to categorize the relevant DTDs effectively.

1 Introduction

XML [3] is widely used to describe information or to exchange data between different web systems. In these applications, the categorization of documents is a basic and useful technique. Clustering XML documents can be used in search engines, digital libraries, XML repositories, and so on. Current clustering methods focus on handling database objects or semi-structured data. However, XML documents are different from either of them. DTD is used to describe the structure of the XML documents as a concise schema. An XML document with corresponding DTD is different from traditional semi-structured data in that its partial schema can be obtained separately without scanning the whole document. Therefore, the efficient method to cluster XML documents should take advantage of DTD. Moreover, it should possess the characteristics of traditional clustering method such as high accuracy, high performance, and the capability to handle large data sets with noises and outliers.

In this paper, we will introduce a new two-level method to cluster DTDs, which can be used to cluster XML documents. First, it clusters the elements in the DTDs

[1] This work was supported by the 973 Projects Foundation of China and the National Doctoral Subject Foundation of China.

based on the content of the elements. The content of an element can be abstracted from the tags of the element and the sub-elements of it. The element clustering is the first level of the whole clustering. After obtaining the clusters of elements, the DTDs are mapped to vectors in multi-dimensional space, in which traditional clustering method is employed to cluster the DTDs, and serves as the second level of the process.

The two-level clustering method considers not only the content of the DTDs but also the structure of them. As mentioned above, the DTDs provide the outlines of the XML documents. The clustering of the DTDs is the clustering of the relevant XML documents. That is to say, using two-level clustering method, we can cluster the XML documents facilitating both the content and the structure of them. Therefore, it can achieve more accuracy and is robust to noises and outliers.

In Section 2, the feature of XML and DTD is analyzed, and some typical clustering methods are introduced. In Section 3, the data model used in our method is presented. We present two-level clustering method in detail in Section 4. In Section 5, experimental results are shown and analyzed. Finally, Section 6 is for concluding remarks.

2 Related Work

XML (eXtensible Markup Language) is a specification of W3C.[3] All XML documents are constructed in *elements* and an element can contain several *attributes*. A *well-formed* XML document is the document whose elements are all nested. The structure of the XML documents can be declared by DTD (Document Type Definition). A DTD declares the possible hierarchical structure of the elements and the possible attributes of the elements. An XML document whose elements and attributes conforming to the DTD is called a *valid* XML document.

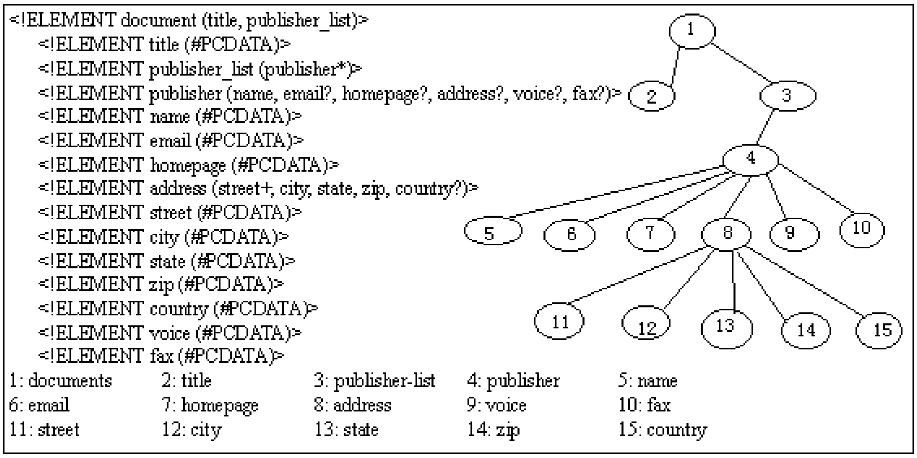
DTD is the structure declaration of an XML document rather than the strict schema definition. It provides the allowable and possible appearance of the elements and attributes, which can be shown as concise schema. Moreover, DTD is usually written or chosen by the author of the XML document, or abided by the standard of a certain domain. The names of the elements and attributes, and the nesting relationship of the elements reflect the content and structure of the documents. In other words, DTDs provide the outline of XML documents, which can be used in document clustering.

The problem of document clustering is to categorize the documents based on the similarity without the prior knowledge on the taxonomy. The research is driven by the hypothesis that closely associated documents tend to be relevant to the same requests, so that grouping similar documents accelerates the searching [4]. There are two major methods to cluster documents: partitioning method and hierarchical method [2]. They all transform documents into vectors or transactions. Each dimension of the vectors or each item in the transactions denotes a key word. Several methods are employed to choose the key words. The naive method is to use all the words appearing in the documents, which leads to very high dimensional objects. Therefore, some techniques are invented to reduce the dimensionality. However, most of these techniques suffer from shortcomings that they are noise-sensitive, and domain-sensitive [2].

ROCK [5] is a typical hierarchical method for clustering objects with categorical attributes. It is insensitive of noises and outliers. CLIQUE [1] is a density-based method for high-dimensional data clustering. This algorithm identifies dense clusters in sub-space of maximum dimensionality automatically. Since density-based, it has the advantage that the time complexity is linear according to the dimensionality of the data.

3 Element and DTD Model

Our method is based on the structure provided by DTDs, which can be generalized to a two-level model. Element is the unit of XML documents. The nested elements reflect the relationship between the elements. In other words, an element and its sub-elements express the content and structure of that element. Element model forms the first level in the framework. However, the element model only denotes the intra-document unit, which could not figure the characteristics of the whole document. As the previous analysis, DTDs provide the outline of the documents. Furthermore, DTD is composed of hierarchical elements. Therefore, DTD model, which forms the second level in the framework, is built on the element model.



```
<!ELEMENT document (title, publisher_list)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT publisher_list (publisher*)>
    <!ELEMENT publisher (name, email?, homepage?, address?, voice?, fax?)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT email (#PCDATA)>
    <!ELEMENT homepage (#PCDATA)>
    <!ELEMENT address (street+, city, state, zip, country?)>
    <!ELEMENT street (#PCDATA)>
    <!ELEMENT city (#PCDATA)>
    <!ELEMENT state (#PCDATA)>
    <!ELEMENT zip (#PCDATA)>
    <!ELEMENT country (#PCDATA)>
    <!ELEMENT voice (#PCDATA)>
    <!ELEMENT fax (#PCDATA)>
```

1: documents	2: title	3: publisher-list	4: publisher	5: name
6: email	7: homepage	8: address	9: voice	10: fax
11: street	12: city	13: state	14: zip	15: country

Fig. 1. A DTD example and its corresponding element tree

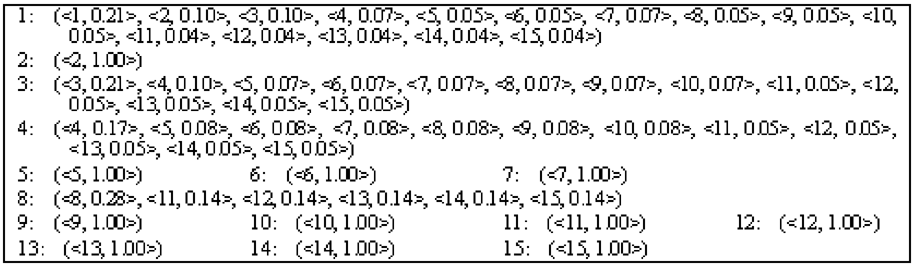
Our model is derived directly from the DTD, which can be regarded as a tree whose nodes are elements or attributes, and edges denote the nested relationship. For example, Fig. 1 shows a DTD and its corresponding element tree.

3.1 Element Model

Every element has its unique identifier, called *element-id*, in this model. An element is denoted as a triple: <*id, tag, depth*>, in which *id* is the element-id, *tag* is the markup, and *depth* is the depth of the element counted from the root of the DTD.

Attributes are treated as special elements that have no sub-elements. In the rest of this paper, we will use the word ‘element’ to describe both elements and attributes.

The characteristics of an element can be transformed to a transaction, in which transaction id is the element-id of it, and the items are the content of that element, including the element itself. Each item is of the form: *<id, weight>*, in which *id* is the element-id of the sub-element that can be reached from the host of the transaction, and *weight* denote the importance of the item in the transaction. Then, the transaction is of the form: $id{:}(<id_1, w_1>, <id_2, w_2>, \ldots, <id_n, w_n>)$. The weight is calculated like this: $w_i = 1/(d_i-d+1)^c/\sum_{j=1}^{n}(1/(d_j-d+1)^c)$, in which d_i denotes the depth of element id_i, d is the depth of element *id*, and c is a constant. Constant c denotes the degree of association between depth and weight. The larger c is, the more important the high level elements are. Furthermore, the item with small depth always acts important role in the transaction because it is closer to the host than other items. The transactions corresponding to the DTD in Fig. 1 is as follows in Fig. 2, in which, c is set to one.



```
1:  (<1, 0.21>, <2, 0.10>, <3, 0.10>, <4, 0.07>, <5, 0.05>, <6, 0.05>, <7, 0.07>, <8, 0.05>, <9, 0.05>, <10,
     0.05>, <11, 0.04>, <12, 0.04>, <13, 0.04>, <14, 0.04>, <15, 0.04>)
2:  (<2, 1.00>)
3:  (<3, 0.21>, <4, 0.10>, <5, 0.07>, <6, 0.07>, <7, 0.07>, <8, 0.07>, <9, 0.07>, <10, 0.07>, <11, 0.05>, <12,
     0.05>, <13, 0.05>, <14, 0.05>, <15, 0.05>)
4:  (<4, 0.17>, <5, 0.08>, <6, 0.08>, <7, 0.08>, <8, 0.08>, <9, 0.08>, <10, 0.08>, <11, 0.05>, <12, 0.05>,
     <13, 0.05>, <14, 0.05>, <15, 0.05>)
5:  (<5, 1.00>)            6:  (<6, 1.00>)            7:  (<7, 1.00>)
8:  (<8, 0.28>, <11, 0.14>, <12, 0.14>, <13, 0.14>, <14, 0.14>, <15, 0.14>)
9:  (<9, 1.00>)            10:  (<10, 1.00>)          11:  (<11, 1.00>)          12:  (<12, 1.00>)
13:  (<13, 1.00>)          14:  (<14, 1.00>)          15:  (<15, 1.00>)
```

Fig. 2. Transactions

The algorithm to build transactions for all elements from DTD tree is as follows:

```
procedure build_transaction (DTD) begin
  add_items (root);
  for each transaction t
    calculate_weight (t);
end
procedure add_items (current) begin
  T[current.id] = {<current.id, 0>};
  for each child c of current begin
    T[current.id] = T[current.id]∪{<c.id, 0>};
    for each element e and <current.id, 0>∈T[e.id]
      T[e.id] = T[e.id]∪{<c.id, 0>};
    add_items (c);
  endfor
end
```

In the algorithm, *T[i]* stores the transaction whose id is *i*. The idea of the algorithm is to find the sub-elements of each element recursively, and then calculate the weight of items in each transaction together.

We define the similarity between two elements based on the transaction model. One assumption is made that the distance between two tags is defined, which denotes the relationship between the words. Furthermore, the value of the distance is larger than 1. The smaller the value is, the closer the relationship is. Semantic distance, which is the distance defined based on the similarity of two tags' content, is chosen to define the distance. Taxonomy or parasynonyms dictionary can be employed to help define the distance. Furthermore, domain experts can define the distance too.

Then, the similarity between two elements is defined as follows: $similarity(e_1, e_2)=\sum_{i,j}(1/distance(tag_i', tag_j'')*f(w_i',w_j''))/(n*m)$, in which the transactions of element e_1 and e_2 are $id_1:(<id_1',w_1'>, <id_2',w_2'>, ..., <id_n',w_n'>)$, and $id_2:(<id_1'',w_1''>, <id_2'',w_2''>, ..., <id_m'',w_m''>)$ respectively, tag_i' and tag_j'' are tags of id_i' and id_j''. The function $f(w_i',w_j'')$ is to calculate the importance of the relationship of the tags acts in the relationship between the elements. And we use $f(w_i',w_j'')=2*w_i'*w_j''/(w_i'+w_j'')$ in our experiments. In this model, the similarity between two elements denotes the weighted-average of the relationship between each pair of items belonging to different transactions. The larger the value is, the more similar the elements are.

3.2 DTD Model

A DTD is treated as a set of elements, although nested. Therefore, a DTD is formalized as: $dtd\text{-}id: (id_1, id_2, ..., id_n)$. After element clustering, there are k element clusters: $c_1, c_2, ..., c_n$. For DTD i, there are n_i^j elements in it belonging to c_j. And $n_i=\sum_{j=1}^{k} n_i^j$ is the number of elements in DTD i. Then, every DTD can be formalized as a k-ary tuple: $(n_i^1/n_i, n_i^2/n_i, ... , n_i^k/n_i)$, so that each DTD is mapped to a vector in k-dimensional space. The scale of each dimension is $[0, 1]$ in the data space. This vector-model is different from keyword-based model. Firstly, the DTD model is based on the element clusters. Since their scale is much smaller than that of keywords, the dimensionality of the DTDs in our model won't be very large. Secondly, each dimension in the k-ary-tuple denotes an element cluster, which is the generalization of a category of elements with similar content and structure. Since our model has considered the relationship among elements, it is more meaningful than other method. Finally, other than the dimensional reduction methods, our low dimensional model of DTD is obtained from the information provided by the author or standard. Our model does not eliminate any part of it. It generalizes the information, so that it can be seen from a universal perspective.

4 Two-Level Clustering Algorithm

The two-level clustering method is corresponding to the two-level model. After forming k element clusters, the DTDs are mapped to vectors in k-dimensional space. Then, DTD clustering is processed. Since element model and DTD model have different forms, their clustering methods should be different. There are some other important issues should be considered:

- There is no prior information about the clusters, especially about element clusters. Even the number of clusters is unknown. And the end condition of the process is hard to define. So, traditional methods are not appropriate for element clustering.
- There must be noises or outliers in elements and DTDs, which are of no similarity with any of the clusters. The mistaken result of element clustering caused by them

may be accumulated in DTD clustering phase. Therefore, the clustering algorithms should be robust to noises and outliers.

– DTDs containing lots of elements belonging to noises or outliers denote that they are in common in some aspects. Therefore, they should be examined further.

The process of our method is as follows: mapping elements→clustering elements →mapping DTDs→clustering DTDs. The introduction of mapping elements has been shown in subsection 3.1. We will provide the detailed description of rest part orderly.

4.1 Element Clustering

The hierarchical method is chosen to cluster elements, for it has the advantage of independence of prior *k*. For using hierarchical method, the goodness measure between two clusters is defined as follows: $g(c_i, c_j)=\sum_{e\in ci,e'\in cj} similarity(e, e')/(n_i*n_j)$. Here n_i and n_j are numbers of elements of c_i and c_j respectively. $g(c_i, c_j)$ is the average value of similarity of all inter-cluster pairs of elements. In hierarchical clustering method large cluster tends to absorb small clusters near it, which seriously effects the clustering accuracy. Since the average value is used in the measure, the large cluster, which in general has more elements similar to elements in other cluster, would not have priority over small clusters to be chosen. So it overcomes the cause of the tendency. Furthermore, the average value reflects the similarity between two clusters from a more universal view than min-, max-, or mean-based strategy.

The process of clustering is the process of merging the pair of sub-clusters having the largest goodness measure. In the initiation, each element is a sub-cluster. The merging iteration does not stop until all elements are merged to one cluster. However, each merge operation is recorded, so that the user can point out in which step the merging should stop. The pseudocode of the clustering process is as follows:

```
1. procedure element_clustering (E) begin
2.   similarity = compute_similarity (E);
3.   EC = initial_clusters (E);
4.   g = initial_goodness(EC, similarity);
5.   process = initial_process (E.size);
6.   while (EC.size > 1) begin
7.     r = EC.choose ();
8.     s = r.similarest;
9.     EC.delete (r);
10.    EC.delete (s);
11.    t = merge (r, s);
12.    process.insert (t, r, s, g [r, s]);
13.    for each x
14.      g[x, t] = (g[x, r]*(x.size*r.size)
             +g[x, s]*(x.size*s.size))/(x.size*t.size);
15.    EC.insert (t);
16.  endwhile
17.end
```

In the procedure, *E* is the set of elements, the matrix *similarity* stores the similarity between two elements using the similarity function defined in subsection 3.1, and the matrix *g* stores the goodness measure between two clusters. Lines 2-4 initialize the structures that will be used in the merging process. The process of merging is recorded in *process*. In line 5, *process* is initialized to contain *n-1* items, where *n* is the number of elements. This is because that the merging phase generates a bintree, whose leaves are elements, and each internal node of the bintree denotes a merge operation. Each item in *process* records the new-generated clusters, the merged sub-clusters, and the goodness between them. Lines 7-12 present the process of choosing sub-clusters to be merged and the merging. In line 7, *r*, the sub-cluster has largest goodness measure between it and its most similar cluster *s* is chosen. Line 13-14 takes goodness function, so that the new-generated cluster is considered. At last, in line 15, the new-generated cluster is inserted into the clusters. Several techniques are borrowed from ROCK [5] to improve the efficiency. For lack of space, the detailed implementation is not shown here.

Note that the algorithm merges all the elements into a whole large cluster. It generates a bintree stored in *process*, which is a sequence of items has the form <*cluster, sub-cluster*$_1$*, sub-cluster*$_2$*, goodness*>. Furthermore, the sequence of items in *process* is same as the sequence of merge operation took place. Therefore, if user specifies an item in *process* that should be the last merge operation in the clustering phase, which is called ***cut item***, the merge operations from the first item to the *cut item* denote the clustering phase. Scanning from the end of the *process* to the item after the *cut item* will obtain a bintree whose leaves are the final clusters. Therefore, if user points out the *cut item*, the final cluster can be determined from *process*. An interactive approach is employed to help the user to determine the *cut item*. Before the approach is introduced, a theorem is presented first:

Theorem 1: If *i* and *j* are two items in process, where *i* is in front of *j* in process, then *i.goodness* is no smaller than *j.goodness*.

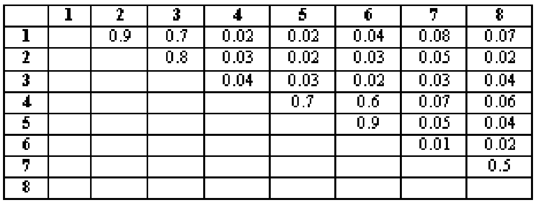
Proof: Consider two neighboured items: $<x,p,q,g(p,q)>$, and $<y,k,l,g(k,l)>$. If $k \neq x$, and $l \neq x$, then $g(p,q) \leq g(k,l)$. Otherwise, assume that $k=x$, which will not effect the conclusion. Then based on line 14 in the procedure, $g(k,l)=(g(p,l)*p.size*l.size+g(q,l)*q.size*l.size)/((p.size+q.size)*l.size)=(g(p,l)*p.size+g(q,l)*q.size) / (p.size+q.size)$. Since $g(p,l) \leq g(p,q)$, and $g(q,l) \leq g(p,q)$, $g(k,l) \leq g(p,q)$. Therefore, it is proved.□

If the goodness measures of two neighbors merge operations is far different, it seems that the nature of these two merge operations is different. Therefore the former merge operation may be a candidate for *cut item*. After the element clustering, a chart, whose x-axis denotes the sequence of items in *process* and y-axis denotes the gradient between the goodness measures of the pairs of neighbored items, is drawn. The points with distinct small value in y-axis are the candidates for *cut item*.

Example: Fig. 3 gives a similarity matrix. Then, the clustering tree is generated. The records in *process* and the corresponding curve are shown in Fig. 4. Point *13* is the cut item for it has distinct small value. So, there are three final clusters: *{1, 2, 3}*, *{4,5,6}*, and *{7, 8}*, as shown in Fig. 3. The nodes in the square are the final clusters.□

It is also possible that there are several points have distinct small values. Then they are all candidates for *cut item*. User can choose any one of them. The detailed analysis on experiments of the choice of *cut item* will be shown in section 5.

Noises and ***outliers*** are element clusters whose members are of small numbers. If the number of elements in a cluster is smaller than a pre-defined threshold *min_element*, the elements in the cluster are treated as noises. Although they may be useful for DTD clustering, as it will be shown later, they should not be treated as separated clusters. Otherwise they will affect the clustering accuracy and efficiency very much. On the other hand, these elements cannot be eliminated because DTD clustering is based on the result of element clustering, in which outliers and noises are important information. Therefore, they are treated as in a special cluster, which is called ***exceptive cluster***. The additional information about them is stored for advanced study on some DTDs. Then, after element clustering, *k* normal element clusters and one exceptive cluster is obtained.



	1	2	3	4	5	6	7	8
1		0.9	0.7	0.02	0.02	0.04	0.08	0.07
2			0.8	0.03	0.02	0.03	0.05	0.02
3				0.04	0.03	0.02	0.03	0.04
4					0.7	0.6	0.07	0.06
5						0.9	0.05	0.04
6							0.01	0.02
7								0.5
8								

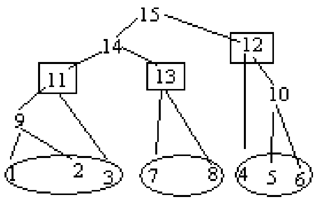
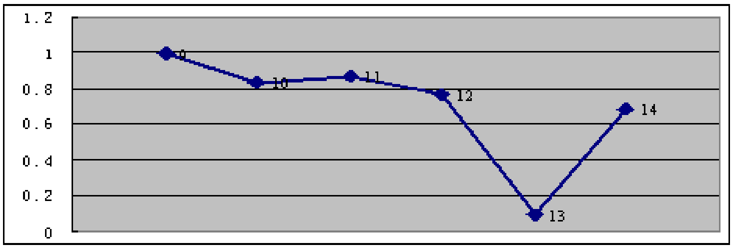


Fig. 3. Similarity matrix and clustering tree



Process:

9:	0.9
10:	0.9
11:	0.75
12:	0.65
13:	0.5
14:	0.048
15:	0.033

Fig. 4. Records in *process* and the chart

4.2 Mapping DTDs

Note that each DTD has only one root element [3]. Therefore, the root element can be regarded as identification of that DTD. Moreover, the transaction of the root element contains all the sub-element belonging to the DTD. In other words, this transaction is equivalent to the element set. Then, after element clustering process, by scanning the root elements, it is easy to mapping the DTDs into vectors. Since the procedure is straightforward, the detailed pseudocode is omitted here. The time complexity of the mapping process is $O(k*N+n)$, in which N is the number of DTDs, and n is the number of elements in all DTDs.

4.3 DTD Clustering

All DTDs are viewed as vectors in k-dimensional space. The dimension that denotes the exceptive cluster is not taken into consideration here. Then, traditional clustering methods discovered for high-dimensional data can be employed on them. CLIQUE [1], which is an automatic subspace method discovered for clustering high-dimensional data, is chosen in our implementation. Since the detailed introduction of CLIQUE can be found in, we will not be verbose here.

The DTDs that don't belong to any cluster found by CLIQUE are treated as noises and outliers. Some of these DTDs have common content and structure. These DTDs are called ***outliers***. Others are ***noises***. Users may be interested in the outliers. Therefore, DTD clustering should have the ability to identify them. Although the outliers could not be found via the *k*-dimension clustering, some of them, which have relatively more elements belonging to exceptive cluster, can be found by checking their elements in exceptive cluster. In general, the percentage of their elements belonging to element noises must be larger than that of other DTDs. Then, the *0*-dimension, which denotes the exceptive cluster, of all DTDs, is scanned. The deviation σ_i of *0*-dimension of DTD *i* is computed: $\sigma_i = (x_i - x)^2$, in which x_i is the value of DTD *i* in *0*-dimension, and *x* is the average value of *0*-dimension of all DTDs. If the deviation σ_i of DTD *i*, which is found to be noises or outliers in CLIQUE, is larger than the squared deviation $\sigma = \sum_{j=1}^{N}(x_j - x)^2/N$, in which *N* is the number of all DTDs, and x_i is larger than *x*, then this DTD may be an outlier. These kinds of DTDs are called ***outlier candidates***. All outlier candidates are mapped to vectors in k'-dimensional space, in which k' is the number of element-clusters that are treated as noises or outliers, and each dimension denotes one of these element-clusters. Then, CLIQUE is used on these outlier candidates, the clusters found is the outliers. Although their scale is relatively smaller than the DTD clusters, they are concentrative for their similar content and structure.

4.4 Complexity Analysis

The time complexity of mapping elements is $O(n*m_{avg})$, in which m_{avg} is the average number of sub-elements of an element. The time complexity of element clustering is $O(n^2*(m_{avg}*m_{max})+n^2+n^2*\log n)$, in which *n* is the number of elements, and m_{max} is the maximum number of sub-element of an element. The space complexity of element clustering is $O(n^2+n*m_{avg})$. The time complexity of mapping DTDs is $O(n+k*N)$. The time complexity and space complexity of DTD clustering can be found in [1].

The two-level clustering method has the advantage that no pre-knowledge required. The interactive way leaves the problem of end-condition to the user. In most cases, it is easy for user to point out the cut item. Furthermore, the method is robust to noises and outliers. Firstly, both the content and structure information is considered, which makes the similarity between two elements more comprehensible. Secondly, the noises and outliers that found in element clustering would not be eliminated. Moreover, the information of the noises and outliers is stored for additional study in DTD clustering, so that the outliers in DTDs could also be discovered. Finally, DTD clustering method finds the large clusters first. Then, the noises and outliers are examined further. The DTDs that have common characteristic are found based on the result of element clustering. Element clustering generalizes the similar elements to clusters, so that the DTDs can be mapped to vectors that are more comprehensible, so that the two-level model can achieve good quality, as is shown in section 5.

5 Experimental Result

The purpose of the experiments is to test the effect and efficiency of our method. We use 200 DTDs, some of which are downloaded from Internet, and others are simulated. The generated DTDs are mapped from some groups of web pages, which is written in HTML. Each group contains similar web pages downloaded from the same web site with similar style. The number of elements in the DTDs scales from 9 to 32. The maximum depth of these DTDs is 7, and the maximum number of children of an element is 8. The DTDs are classified to three groups artificially based on their content. The distance function of pair of words is defined. The experiment is done in a Pentium III 500 PC with 128M memory, and Windows NT 4.0.

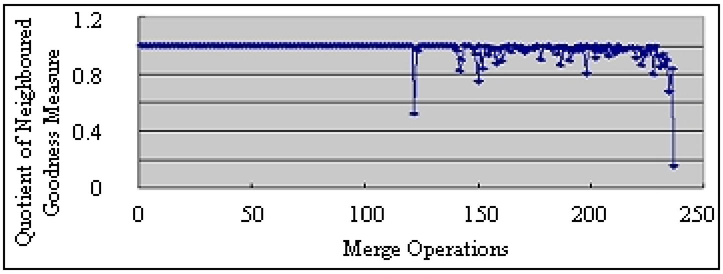


Fig. 5. Candidates for cut items

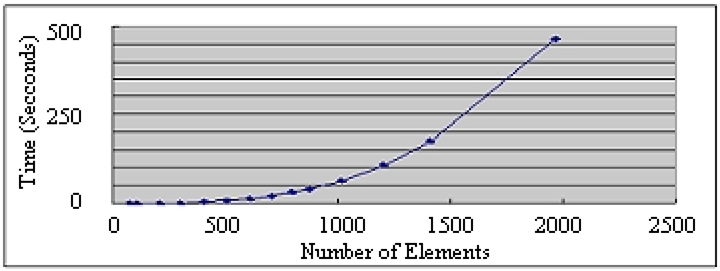


Fig. 6. Performance

Since the result of element clustering depends highly on the choice of *cut item*, we analyze the chart generated first. There always are several candidates as shown in Fig. 5. The points before the first candidates are all of high values near *1*. After examining the records in *process* we found that most of these points denote the merge operations on leaves in element trees. The merged leaves are elements that use synonyms as tags. Although the first candidate is usually of distinct small value, it should not be chosen, or most elements in high level of element tree will be treated as noises and outliers. As the first candidate, the last few candidates also have distinct small values. However, this is because of that the non-relevant clusters are merged together. Since the clusters are of no relationship, the goodness measure is near 0. Therefore, these candidates are not appreciated for *cut item*. In experiments, if user chooses one of the few last candidates, it effects the result of the DTD clustering greatly. The rest candidates are analogous for accuracy of DTD clustering, as is shown later. Each candidate represents a level of generalization of elements. The clusters merged between two candidates are treated as similar when seen from a certain perspective.

Fig. 6 illustrates the performance of the element clustering.

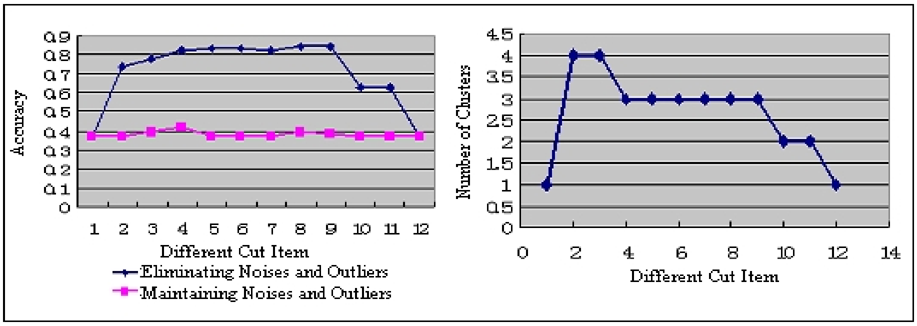


Fig. 7. DTD clustering result

The types of the DTDs are pre-defined as mentioned above. The ***type*** of a cluster is the type which has largest support in the cluster. It is the type that has most DTDs in the cluster according to it. If a DTD's type is agree with the type of its cluster, it is ***right-clustered***. Otherwise, it is ***mis-clustered***. The ***accuracy*** of clustering result is the percentage of right-clustered DTDs. The accuracy of DTD clustering is shown in Fig. 7. There are two curves. One denotes the accuracy that is obtained by eliminating the noises and outliers of element, the other denotes the accuracy that is obtained when taking consideration all the element clusters, including noises and outliers. It is obvious that the former is more accurate than the latter is. The reason is that noises and outliers always appear randomly, and the latter one treat them as independent dimensions, which greatly effect the accuracy. As a matter of fact, CLIQUE tends to merge all the DTDs into one in the latter case, because the vectors are distributed in the high-dimensional space randomly. Different points in the same curve denote the accuracy when choosing different *cut items*. The candidates are sorted by their appearance in the curve generated from *process*. Choosing the first candidate as *cut item* achieves poor accuracy; so does the last few cases. However, if user chooses the candidates in the middle part, DTD clustering achieves high accuracy. Furthermore, the number of clusters found by DTD clustering is shown in Fig. 7. As the accuracy, the number of clusters found by choose the first candidate and the last few candidates lead to wrong result. In addition, the last few candidates generate fewer clusters. Since no-relevant elements are merged together, then it is hard for DTD clustering to identify irrelevant DTDs. The first candidate also generates fewer clusters. This is because that most of the elements that have relationship has not been merged yet. Therefore, CLIQUE considers that the vectors are randomly distributed in the high dimensional space. However, a few candidates lead to more numbers of result clusters. This is because of that some element clusters is not merged, and add some additional dimensions that distinguish the DTDs we classified to the same category. Moreover, CLIQUE tend to merge all the DTDs together when the noises and outliers are taken into consideration in DTD clustering phase, as we mentioned above. And the number of final DTD clusters found is always wrong in this case.

We add another five DTDs that do not similar with any of the three pre-defined groups. The DTDs are not found as a cluster for its low density in high-dimensional space. After examining the element noises and outliers, the five DTDs are all found.

6 Conclusions

In this paper, a new two-level method for DTD clustering, which can be used in XML clustering, is presented. The method clusters the elements and DTDs separately. Furthermore, the process of DTD is based on the result of element clustering.

When processing element clustering, the elements are mapped to transactions. This model takes into consideration of both the content and the structure of the elements. The content is denoted by the words used as tags in the element and its sub-elements. The structure is denoted by the nested relationship between elements. The degree of association between elements is denoted by the weight of items in the transactions. Furthermore, element clustering is prior knowledge independent. That is to say, user need not know the information about the element clusters. An interactive way is employed to help the user determine the end condition of element clustering. The experiment shows that with the strategy of choosing middle part candidates as *cut item*, the two-level method can obtain result with high quality.

DTD clustering is based on the result of element clustering, which provides the generalized information of elements. It is different than traditional keyword based vector models, since the words of tags are chosen by the writers. Furthermore, the information is generalized, so that it denotes the relevant elements.

Noises and outliers are also considered in our method. Firstly, element noises and outliers are found to avoid their effect on DTD clustering. Secondly, the noises and outliers found in DTD clustering is further studied by examining the noises and outliers of elements in them. So the outliers of DTDs can be found.

The experimental results show that the accuracy of the two-level method is high because that it takes full advantage of the information provided by DTDs.

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P.: Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. Proc. of the ACM SIGMOD Int'l Conference on Management of Data, Seattle, (1998).
2. Boley, D., Gini, M., Gross, R., Han, E.H., and Hastings, K.: Partitioning-Based Clustering for Web Document Categorization. To appear in Decision Support Systems Journal.
3. Bray, T., Paoli, J., and Sperberg-McQueen, C.M.: Extensible Markup Language (XML) 1.0, February 1998. W3C Recommendation available at http://www.w3.org/TR/1998/REC-xml-19980210, (1998).
4. Faloutsos, C., and Oard, D.: A Survey of Information Retrieval and Filtering Methods. Department of Computer Science, University of Maryland, Technical Report, CS-TR-3514, (1995).
5. Guha, S., Rastogi, R. and Shim, S.: ROCK: A Robust Clustering Algorithm for Categorical Attributes. Proceedings of ICDE, (1999).

XML Queries via SQL

Cindy Xinmin Chen[1,⋆] and Ashok Malhotra[2]

[1] Computer Science Department
University of California at Los Angeles, Los Angeles, CA 90095, USA
`cchen@cs.ucla.edu`
[2] IBM Thomas J. Watson Research Center
30 Saw Mill River Road, Hawthorne, NY 10532, USA
`petsa@us.ibm.com`

Abstract. As XML [12] becomes more popular, we expect XML documents to be stored in databases with different representations. So an important issue is to translate XML queries into the language of the database. XPath [13], the XML Path Language, is a standard for expressing navigation and selection in XML documents. Hence it can be used to specify certain kinds of XML queries. If XML documents are stored in a relational database, XPath queries have to be translated into SQL [11]. This paper proposes an algorithm to translate queries expressed in XPath to SQL statements. The algorithm uses information that specifies the mapping of XML documents to relational tables.

1 Introduction

XML [12] is the next generation mark-up language, it has more functionality than HTML [7] and is easier to learn and use than SGML. Usually, a set of XML documents has a common DTD (Document Type Definition) [12] file to specify the structure of the XML documents. XML is a good candidate for web repositories, so a major issue is to be able to query the contents of XML documents efficiently.

There are several proposed XML query languages. The most powerful of them is XML-QL [4], which evolved from the Strudel [9] query language for ordered, semi-structured data. XML-QL uses a directed labeled graph data model. In this model, each XML tag becomes an edge labeled with the tag name and directed to an individual node. Non-leaf nodes correspond to attribute-value pairs. Leaf nodes correspond to element values. Each node has a unique ID and there is no order relation between nodes representing sibling elements. XML-QL includes a `construct` clause to specify the structure of the result and allows nested `where` and `construct` clauses.

Lorel [6] uses a data model similar to that of XML-QL. Its design is based on OQL [1], thus it is syntactically more complex than XML-QL.

XMAS [8] is the query language used in MIX [2]. It is declarative and rule-based. However, XMAS is more or less a subset of XML-QL and lacks many of features one expects in a standard query language.

⋆ This work was done while the author was at the IBM Thomas J. Watson Research Center during the summer of 1999.

For SQL users, these query languages have the disadvantage of requiring one to learn another language. XML users would prefer an XML-style query language but none of the languages described above is a standard. [10] discusses a possible way to query XML documents stored on relational database by keeping the translation to SQL transparent to XML users. They propose translating XML-QL queries into SQL statements. Yet [10] cannot handle all the semantics of semi-structured queries over XML data.

XPath [13] is a standard proposed by the World Wide Web Consortium to address navigation and selection in XML documents and can be used to express a certain subset of XML queries. An XPath location path consists of a sequence of one or more location steps separated by "/". The steps in a location path are interpreted from left to right. The leading "/" selects the root element of the XML document. Each succeeding step, in turn, selects a set of children of the parent element. Selection criteria can be included in location paths by encoding them within square brackets "[]".

In this paper we describe an algorithm to query XML documents efficiently by translating XPath location paths to SQL statements. This is similar to [10] although we use a standard XML language to express the queries. In Section 2, we describe how an XML document is stored in a database. In Section 3, we demonstrate how to query a set of XML documents with some examples. In Section 4, we discuss the design and implementation of the algorithm to translate an XPath queries into an SQL statements. Section 5 concludes the paper. The algorithm is shown in Appendix A.

2 Storing XML Documents in a Database

The Document Object Model (DOM) [5] is an application programming interface for parsed documents. It defines the logical structure of documents and the way a document and its parts are accessed and manipulated.

In the DOM, parsed documents are modeled as a tree of objects — specifically, a hierarchy of nodes. Some nodes may have child nodes while others are leaf nodes that contain values but no children. Figure 1 is a representation of a DOM tree of the following sample XML document.

```
<State>
 <City>
  <Locality>
   <Street>Shady Grove</Street>
   <Street>Aeolian</Street>
  </Locality>
  <Locality>
   <Street>Over the River, Charlie</Street>
   <Street>Dorian</Street>
  </Locality>
 </City>
</State>
```

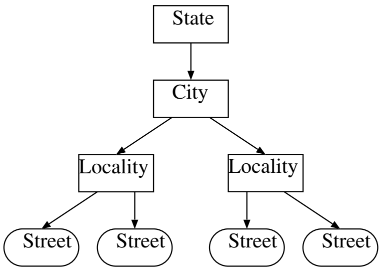


Fig. 1. DOM tree representation of the sample XML document

Each rectangle node in the DOM tree in Figure 1 corresponds to an element in XML that has children and each rounded rectangle node corresponds to an attribute of the element or a sub-element that has a value. Arrows between nodes indicate parent-child relationships.

To store a collection of XML documents that obey a particular DTD as a set of relational tables, we need to specify how the data in the documents is mapped into the relational tables. The IBM DB2 XML Extender [3] uses an XML file called a Document Access Definition (DAD) to specify this mapping. Typically, each type of element is stored in a table of its own. The attributes of the element appear as columns of that table.

The name of the column is typically the name of the attribute but can be different. In addition to this, the DAD file also stores the following information about each attribute:

– type — data type of the attribute of an XML element.
– path — the path from the root of the document to that attribute, i.e., to which XML element the attribute belongs and all of its ancestor XML elements. The DAD file also specifies the key for each table here, denoted by *[@Key]* after the name of the attribute.
– multi_occurrence — whether the attribute may appear more than once in the XML document.

3 Example of Querying XML Documents

In this section, we discuss how to query XML documents shredded into relational tables using SQL through the following examples.

Consider a set of XML documents containing information about customer orders. Each XML document contains a root `Order` element and each `Order` element has several `Customer` and `Part` sub-elements. Each `Part` element has `Shipment` sub-element and `Price` attribute, etc.

```
<Order Key="1">
 <Customer>
  <Name>American Motors</Name> <Email>parts@am.com</Email>
 </Customer>
 <Part Key="68">
  <Color>red</Color> <Quantity>36</Quantity> <Price>34850.16</Price>
  <Tax>0.06</Tax>
  <Shipment>
   <ShipDate>1998-08-20</ShipDate> <ShipMode>AIR</ShipMode>
  </Shipment>
 </Part>
 <!-- information about other parts -->
</Order>
```

The DOM tree representation for such a document is shown in Figure 2. A possible DAD file is shown below.

```
<DAD>
 <dtdid>E:\dtd\order.dtd</dtdid>
 <validation>YES</validation>
 <Xcolumn>
  <table name="order_tab">
   <column name="order_key">
           type="integer" path="/Order[@Key]" multi_occurrence="NO"/>
  </table>
  <table name="customer_tab">
   <column name="Name">
           type="varchar(50)" path="/Order/Customer/Name[@Key]"
           multi_occurrence="NO"/>
   <column name="Email">
           type="varchar(50)" path="/Order/Customer/Email"
           multi_occurrence="NO"/>
  </table>
  <table name="part_tab">
   <column name="part_key"
          type="integer" path="/Order/Part[@Key]" multi_occurrence="NO"/>
   <column name="color"
           type="varchar(50)" path="/Order/Part/Color"
           multi_occurrence="YES"/>
   <column name="quantity"
           type="integer" path="/Order/Part/Quantity"
           multi_occurrence="YES"/>
   <column name="price"
           type="double"
           path="/Order/Part/Price"
           multi_occurrence="YES"/>
   <column name="tax"
           type="double" path="/Order/Part/Tax" multi_occurrence="YES"/>
  </table>
  <table name="shipment_tab">
   <column name="shipdate"
           type="date" path="/Order/Part/Shipment/ShipDate"
           multi_occurrence="YES"/>
   <column name="shipmode"
           type="varchar(50)" path="/Order/Part/Shipment/ShipMode"
           multi_occurrence="YES"/>
  </table>
 <Xcolumn>
</DAD>
```

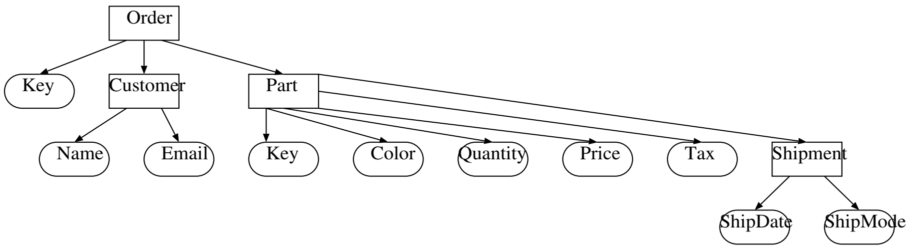


Fig. 2. DOM representation of order.xml

The above DAD file specifies that the order documents are stored in four tables; one table for each XML element. It also specifies that attributes of the element are stored as columns of each table.

To create a database according to this DAD file, some additional information is required. For those tables, which do not have a "*[@Key]*" in their column path, the keys of tables corresponding to their parent elements must be borrowed. The key columns of the parent tables are added into the child table and specified as the key of the child table. Keys of the parent tables are always included in the child table as foreign keys referring to the parent tables.

Thus, the database schema derived from the above DAD file is:

```
order_tab(order_key)
customer_tab(name, email, order_key)
part_tab(part_key, color, quantity, price, tax, order_key)
shipment_tab(part_key, shipdate, shipmode)
```

An XML query for this database may be: "find the shipping date for parts whose prices are greater than 20000". This can be expressed in XPath syntax as:

```
/order/part[price>20000]/shipment/@shipdate
```

The initial "/" indicates the document root. Following this we navigate to the set of all `order` children of the root. From there we select the `part` children whose `price` attribute has a value greater than 20000. Further navigation gets us the set of `shipment` sub-elements of the selected parts and then their `shipdate` attributes which are returned.

According to our algorithm in Appendix A, and the DAD file shown above, this query can be expressed in SQL as:

```
SELECT shipment_tab.shipdate
FROM   order_tab, part_tab, shipment_tab
WHERE  part_tab.price > 20000 AND
       part_tab.order_key = order_tab.order_key AND
       shipment_tab.part_key = part_tab.part_key
```

Now consider the following query: "find the `shipment` element of the parts whose price are higher than 20000".

The XPath syntax for this query is:

```
/order/part[price>20000]/shipment
```

The last step in the location path is now an XML element rather than an attribute. Thus every column in the table that stores the XML element should be returned.

The corresponding SQL statement is:

```
SELECT shipment_tab.part_key, shipment_tab.shipdate,
       shipment_tab.shipmode
FROM   order_tab, part_tab, shipment_tab
WHERE  part_tab.price > 20000 AND
       part_tab.order_key = order_tab.order_key AND
       shipment_tab.part_key = part_tab.part_key
```

Lastly, let us look at the following query: "find the `part` element and all of its sub-elements for the parts whose price are higher than 20000".

The XPath syntax is:

```
/order/part[price>20000]
```

The last step in the location path is an XML element with sub-elements. So all the attributes of this element and all the attributes of its sub-elements, i.e., the nodes appear in the DOM tree below it are returned.

The corresponding SQL statement is:

```
SELECT part_tab.part_key, part_tab.color, part_tab.quantity,
       part_tab.price, part_tab.tax, part_tab.order_key,
       shipment_tab.part_key, shipment_tab.shipdate,
       shipment_tab.shipmode
FROM   order_tab, part_tab, shipment_tab
WHERE  part_tab.price > 20000 AND
       part_tab.order_key = order_tab.order_key AND
       shipment_tab.part_key = part_tab.part_key
```

4 Translating XPath Queries into SQL Statements

Since the DAD file contains all the information related to database schema design, the first step of the translation is to read the DAD file and store the information it contains in a data structure in main memory. We used a hash table for this. Each entry in the hash table represents an XML element or attribute. Each entry also contains the path information of the element or attribute, the names of the table and column where the element or attribute is stored, and the key and the foreign key of the table. In addition to the hash table, we used an array to store the path information of all the columns defined in the DAD file.

The input XPath query is then parsed into navigation tokens separated by the "/" symbol. The name of the XML element or attribute in each token is then used as the key to search the hash table. Information about the element or attribute is retrieved, and corresponding additions to the `SELECT`, `FROM` and `WHERE` clauses of an SQL statement are generated.

Column names are added into the `SELECT` clause; the table names are added into the `FROM` clause and the foreign key constraints are added into the `WHERE` clause. As we mentioned, strings representing selection criteria can appear in square brackets "[]". These selection criteria are added into the `WHERE` clause as well while necessary additions to the `SELECT` and `FROM` clause associated with the column mentioned in the selection criteria are made at the same time.

The key point of translating an XPath query into an SQL statement is to keep the database query process transparent to XML users. Users are able to query XML documents using the cognitive model of the XML documents to write the query. This method efficiently handles all the semi-structured queries over XML data that can be expressed by XPath standard.

5 Conclusion

This paper discusses an algorithm to query XML documents by translating an XPath queries into SQL statements. The XML documents are stored in tables in a database with the schema of the database derived from the DAD file that specifies the mapping of the XML documents onto database tables. Foreign key constraints of the tables are used to maintain the parent-child relationship between the elements of the XML document.

References

1. A. M. Alashqur, S. Y. W. Su and H. Lam. OQL: A Query Language for Manipulating Object-oriented Databases. In *Proc. Fifteenth VLDB*, pp.433-442, 1989. 53
2. C. Baru, A. Gupta, B. Ludaescher, R. Marciano, Y. Papakonstantinou and P. Velikhov. XML-Based Information Mediation with MIX. *Exhibitions Program of ACM SIGMOD*, 1999. 53
3. The DB2 XML Extender. *http://www-4.ibm.com/software/data/db2/extenders/xmlext/*, 1999. 55
4. A. Deutsch, M. F. Fernandez, D. Florescu, A. Levy and D. Suciu. XML-QL:A Query Language for XML. *http://www.w3.org/TR/NOTE-xml-ql*, 1999. 53
5. Document Object Model (DOM) Level 1 Specification Version 1.0. *http://www.w3.org/TR/REC-DOM-Level-1*, 1998. 54

6. R. Goldman, J. McHugh and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99)*, pp. 25-30, 1999. 53
7. Hypertext Markup Language, "HTML 3.2 Reference Specification" *http://www.w3.org/TR/REC-html32* , 1997. 53
8. B. Ludascher, Y. Papakonstantinou and P. Velikhov. A Brief Introduction to XMAS. *http://www.db.ucsd.edu/Projects/MIX/docs/XMAS-intro.pdf*, 1999. 53
9. M. Fernandez, D. Florescu, A. Levy and D. Suciu. A Query Language for a Web-Site Management System, *SIGMOD Record*, vol.26, no.3, pp.4-11, 1997. 53
10. J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt and J. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. *Proc. 25th VLDB*, 1999. 54
11. The SQL Standard. *http://www.jcc.com/SQLPages/jccs_sql.htm*, 1999. 53
12. Extensible Markup Language (XML) Version 1.0. *http://www.w3.org/TR/REC-xml*, 1998. 53
13. XML Path Language (XPath) Version 1.0. *http://www.w3.org/TR/xpath*, 1999 53, 54

A Algorithm

```
begin
  parse dad_file
  record all paths in array
  create hash_table for database schema

  parse xpath_query at ‘‘/’’ boundaries

  for each xpath_query step except the last one do
    begin
      parse the xpath_query step by ‘‘[’’
      if string following ‘‘[’’
        put string into WHERE clause
      search hash_table
      put table name into FROM clause
      put foreigh key constraint into WHERE clause
    end

  for last xpath_query step
    if string following ‘‘[’’
      put string into WHERE clause
    if ‘‘@’’
      search hash_table
      put column name into SELECT clause
      put table name into FROM clause
      put foreigh key constraint into WHERE clause
    else
      compare xpath_query with the array of paths
      for each element in the difference
      begin
        search hash_table
        put column name into SELECT clause
        put table name into FROM clause
        put foreigh key constraint into WHERE clause
      end

end
```

Inferring DTD to Facilitate User-Oriented XML Documents Query[*]

Shihui Zheng

Computer Science Department
Fudan University, Shanghai, China, 200437
shzheng0@fudan.edu.cn

Abstract. In the scenario of XML repository, where are large amounts of relevant documents with DTDs and without DTDs. To query the explicit and implicit documents in the repository, we provide user with a pattern specification language to specify the structural pattern of his/her interest, and organize all of the data matching the pattern are grouped as view documents. For facilitates the query formulation and evaluation, a view DTD is inferred from the repository. This paper introduces the algorithm for inferring the view DTD from the XML repository.

1 Introduction

Extensible Markup Language (XML) [1] is the new standard of data exchange and representation on Internet. XML is simple, easily parsed, and self-describing. An *XML document* is optionally associated with a *Document Type Declaration (DTD)*, which provides the structural description the *XML document*. DTD can be viewed as the schema of the *XML document*, but compared with schema in database, DTD is less restrictive and permit more variation in the data. Furthermore, DTDs are not mandatory. *XML documents* may be without DTDs.

We consider the scenario of XML repository, where are large amounts of relevant *XML documents* coming form different Web sites. These documents may have or have not DTDs. The structures of these documents are explicit (with DTDs) or implicit (without DTDs). We propose a user-oriented approach to flexibly query the explicit and implicit documents in this setting. In user–oriented query (the architecture is shown in fig.1.), user specifies the information of his/her interest by a pattern specification language. According to the pattern specification, the explicit and implicit data matching this pattern are grouped and merged into (virtual or materialized) pattern view documents, at the same time, a (pattern) view DTD is inferred from them. View DTD adds a structuring interface on the tops of the underlying heterogeneous *XML documents*. It facilitates the *XML documents* query in the following

[*] The work is supported by the 973 Foundation of China and National Doctoral Subject Foundation of China.

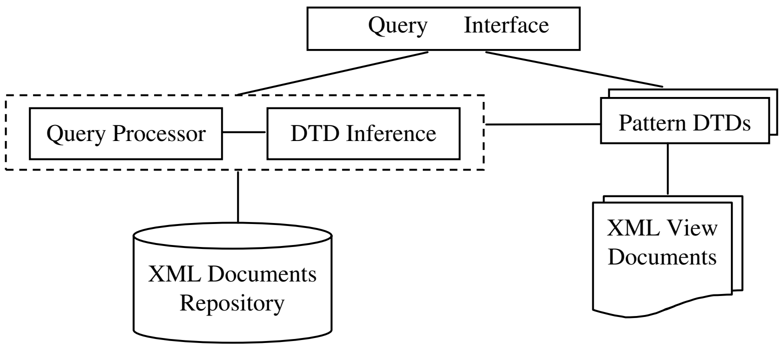


Fig. 1. The architecture of the user-oriented query

ways: 1) it is referred by user query interface which displays the structure of view documents and assists user formulate reasonable query; 2) it can be passed to query processor to evaluate the query against the view efficiently. In this paper, we focus on the technique for inferring pattern view DTDs.

2 Document Model and Pattern Specification Language

Our XML model can be viewed as a version of Object Exchange Model (OEM) model [5]. We use tree model for *XML documents* as well as DTDs. In our model, An *XML document* or a DTD is an ordered tree, where vertices are elements or types, and edges denote the relationships between the vertices. Every vertex in *XML document* tree is reachable from the root element or type. Without loss of generality, in our model, we do not consider mixed content and empty elements, and we do not treat the attributes orther than ID attribute. *XML documents* and DTDs are formally defined as follows.

Definition 2.1: *(XML document)* A *XML document* is tuple $<E, N_E, R_E>$, Where N_E is the set of element names, R_E is the distinguished root element of the *XML document, E* is a sequence of elements. Each element $e \in E$ is a triplet *<eid, name, content>,* where *eid* is the identifier of element, we assume that every element has a unique ID attribute. *name* is a function *name:* $E \longrightarrow N_E$. The *content* is either a character string, or an list of {*name:eid*} pairs .

Definition 2.2: *(Document Type Declaration)* A *Document Type Declaration (DTD)* is a tuple $<T, N_T, R_T>$, where *T* is a sequence of *DTD* type, N_T is the set of type names. Every *DTD* has a unique root type R_T. Each type $\mathcal{T} \in T$ is a triplet *<tid, name, content>*, where *tid* is the type identifier, *name* is the name of type $\mathcal{T}$, it is a function *name:* $T \longrightarrow N_T$. The *content* is either *PCDATA* or a regular expression

r::= (r.r) | (r|r) | (r*) | r+ | r? | name.

A regular expression is associated with a regular language. In the following of the paper, we use L(r) to express the regular language of r.

Definition 2.3 (*Valid XML Document*) A *XML document d* with a *DTD D* is *valid*, if the following hold:

1) the root element R_E of d conforms to the root type R_T of D, i.e. $name(R_E)=name(R_T)$, and
2) given an element e in d, it conforms to a unique type $\mathcal{T}$ in D, s.t. $name(e)=name(\mathcal{T})$ and if the *content* of e is a string then $content(\mathcal{T})=PCDATA$, else $content(e) \in L(content(\mathcal{T}))$.

In this paper, we assume all of the documents with DTDs are *valid*. For a document without DTD, its DTD is implicit. Each element in the implicit document implies its type, and its type can be inferred from it (illustrated in section 3).

The examples of XML element fragments and its *valid* DTD type are following .

publication: e_1*= (title:* e_2*; author:* e_3*; journal:* e_4*; abstract:* e_5*; year:* e_6*)*
publication: e_7*= (title:* e_8*; author:* e_9*; author:* e_{10}*; proceedings:* e_{11}*; abstract:* e_{12}*; year:* e_{13}*)*

…

PUBLICATION: $\mathcal{T}_1$*=(title:*$\mathcal{T}_2$ *; (author:*$\mathcal{T}_3$*)+; (journal:*$\mathcal{T}_4$ | *proceedings:*$\mathcal{T}_5$*); abstract:*$\mathcal{T}_6$*; year:*$\mathcal{T}_7$*)*

In the user-oriented query, we provide pattern specification language for user to specify the structure of his/her interest. Our pattern specification language can be viewed as a subset of XML-QL [2], absorbing some properties of information extracting. A structural pattern specification is basically a query, where SELECT clause defines one or more pattern variables. Each pattern variable binds to the elements (in implicit documents) and the types (in the DTDs of the documents) matching the structural pattern. WHERE clause defines the conditions of the pattern variables. All of the elements (in implicit and explicit documents) bound by pattern variables will be grouped as a (virtual or materialized) view document named by the pattern name following the PATTERN clause.

Example 2.1 The following example defines a pattern named *deptpublication*. SELECT clause contains a pattern variable *p*, it binds to all of the publications that can be reached from the root department via zero or more edges.

```
PATTERN=deptpublication
SELECT p
WHERE <publication><*>
   p: <publication></></></>
```

In some cases, user may have known some structural information (e.g. from explicit DTDs) about the underlying repository and want to find the elements and types with similar structures. In a structural pattern specification, user can specify a query retrieving all of the types and elements that have structures similar with an explicit DTD type using function *approximate* in WHERE clause.

Example 2.3 Consider the following query for finding all of the elements and types approximate matching an explicit type *professor*, where the IN clause indicates the type professor coming from DTD "www.csdepartmentinfo.dtd".

```
PATTERN=deptpapers
SELECT P
WHERE <department><*>
   P: <$a><\><\><\>
      <department><professor>
   X: <papers><\><\><\> IN www.csdepartmentinfo.dtd;
AND approximate(P,x)
```

3 Pattern view DTD Inference

When user defines a pattern, at first, query processor search and find all of the elements and types matching the pattern specification from the implicit and explicit documents in *XML* repository. Then these data are grouped as view documents and a pattern view DTD is inferred automatically accompanying with it. The pattern DTD inferring algorithm is shown in fig. 2.

Algorithm 3.1 Pattern DTD Inference

```
Scan the implicit and explicit documents and pick out
all of the element $t_1, t_2, \ldots, t_m$ and types $T_1, T_2, \ldots, T_n$ matching
the structural pattern.
For each $t_i$ ($1 \leqslant i \leqslant m$) do
 translate($t_i$, $T_i$)
Divide $T_1, T_2, \ldots, T_n$ into k classes $D_1, D_2, \ldots, D_k$ by type names;
For each $D_i$ ($1 \leqslant i \leqslant k$) do
  $D_p^i := \phi$;
  For each $T_i \in D_i$ do
    $D_p^i$:=merge($T_i, D_p^i$);
new($D_p$);
name($D_p$):=patternname;
content($D_p$):=($D_p^1, D_p^2, \ldots, D_p^k$);
```

Fig. 2. The pattern DTD Inference Algorithm

The first stage of algorithm is to translate the elements matching the pattern as the types corresponding to them. If an element is a character string, we convert it as PCDATD, otherwise, for complex element, we just substitute a regular expression for the multiply occurrences of a same name (of child element). For example, the element e_1 and e_2 in the example 2.1 will be translated as the following types T_1' and T_7'.

PUBLICATION: T_1' = *(title; author; journal; abstract; year)*

PUBLICATION: T_7' = *(title; author+; proceedings; abstract; year)*

Notice that there are types with same name, which may be extracted from different DTDs or translated from elements. So the next step of inferring algorithm, is to merge the source types with same name into one target type which contains all of the structures of those source types, and group all those target types as a DTD over the view documents. Intuitively, it is easy to find a target *structure subsumes* the source types. However, observes that the view DTDs may describe more documents that can not appear in view documents. For example, the following DTD type $\mathcal{T}$'may be the result type of merging $\mathcal{T}_1$' and $\mathcal{T}_7$' above.

$\mathcal{T}$'= (title; author+; journal?; proceedings?; abstract; year)

It contains more structural information than $\mathcal{T}_1$' and $\mathcal{T}_7$', because in $\mathcal{T}$', publication may appear in journal and proceedings simultaneously, but neither $\mathcal{T}_1$' nor $\mathcal{T}_7$' permit this. For browsing structure and computing query, a view DTD should be as accurate as possible, i.e. it describes the "fewest" documents that cannot appear in a view [6]. In inferring algorithm, we provide one criterion *least structure subsumes* to evaluated against the merging of types with same name. The *structure subsumes* and the *least structure subsumes* are defined as follows.

Definition 3.1: (*structure subsumes*) Type $\mathcal{T}$ *structure subsumes* type $\mathcal{T}$', denoted as $\mathcal{T}' \sqsubseteq \mathcal{T}$, if 1) name($\mathcal{T}$)=name($\mathcal{T}$') and 2) if $\mathcal{T}$' is a PCDATA then $\mathcal{T}$ is a PCDATA , else L(content($\mathcal{T}$'))⊆L(content($\mathcal{T}$)) and given a child type t_j' in $\mathcal{T}$', there is a child type t_i in $\mathcal{T}$, s.t. $t_j' \sqsubseteq t_i$.

Definition 3.2 (*least structure subsumes*) Given type $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_n$, we say type $\mathcal{T}$ *least structure subsumes* $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_n$, if 1) $\mathcal{T}$ *structure subsumes* $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_n$, and 2) there is no type $\mathcal{T}' \neq \mathcal{T}$, $\mathcal{T}$' *structure subsumes* $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_n$, and $\mathcal{T}' \sqsubseteq \mathcal{T}$.

Given a group of types, our inference algorithm would generate a least pattern view DTD in which each type *least structure subsumes* all of the source types with same name. So our view DTD is the least view DTD over the (virtual or materialized) view documents.

In inference algorithm, function merge is used to merge two source types $\mathcal{T}$ and $\mathcal{T}$' as a type that *least structure subsumes* $\mathcal{T}$ and $\mathcal{T}$'. It merges the content of $\mathcal{T}$ and $\mathcal{T}$' at first, then merge their child types with same name. The contents of types $\mathcal{T}$ and $\mathcal{T}$' are regular expressions r and r'. We use operator "$\vee$" to specify the merging of regular expression r and r'. Another operator "$\oplus$" is used to express the merging of two regular expressions with same name. Operators "$\vee$" and "$\oplus$" are cumulative, they are shown as follows.

Table 1. The operator "$\oplus$"

$r \oplus r = r$	$r \oplus r? = r?$	$r \oplus r+ = r+$	$r \oplus r* = r*$
$r? \oplus r+ = r*$	$r? \oplus r* = r*$	$r+ \oplus r* = r*$	

Table 2. The operator "$\vee$"

r	r'	$r \vee r'$	*if*
r	r'	$r \oplus r'$	name(r)=name(r')
r	r'	r | r'	$L(r) \neq L(r')$
(r_1, r_2)	(r_1', r_2')	$(r_1 \oplus r_1', r_2 \vee r_2')$	name(r_1)=name(r_1')
$(r_1 \mid r_2)$	(r_1', r_2')	$((r_1 \oplus r_1') \mid r_2, r_2 \vee r_2')$	name(r_1)=name(r_1')
$(r_1 \mid r_2)$	$(r_1' \mid r_2')$	$(r_1 \oplus r_1') \mid (r_2 \vee r_2')$	name(r_1)=name(r_1')
$(r_1 \mid r_2)$	(r_1', r_2')	$(r_1 \oplus r_1', r_2 \oplus r_2')$	name(r_1)=name(r_1') and name(r_2)=name(r_2')

Example 3.1 Consider the *PUBLICATION* types $\mathcal{T}_1$ in section 2 and the following type $\mathcal{T}$

$\mathcal{T}$ = *(title; author+; (techreport, abstract) | proceedings; year).*

We use the beginning letters to denote the name of child type, then $\mathcal{T}_1$ and $\mathcal{T}$ are merged as following

$$
\begin{aligned}
&(t, a^*, j|p, ab, y) \vee (t, a+, (te, ab)|p, y)\\
&= (t \oplus t, a^* \oplus a+, (j|p, ab) \vee ((te, ab)|p), y \oplus y)\\
&= (t, a^*, ((j|p, ab) \vee (te,ab)) \mid ((j|p, ab) \vee p), y)\\
&= (t, a^*, (j|p|te) \vee te, ab \oplus ab)| (j| (p \oplus p), ab), y)\\
&= (t, a^*, (j|p|te, ab)| (j|p, ab), y) = (t, a^*, j|p|te, ab, y)
\end{aligned}
$$

4 Related Work and Conclusion

Our work is related with schema inference of semistructured data [3][4][8]. Our work is distinguished from these works in that pattern view DTD inference is user-oriented, the pattern structure to be inferred is specified by user through pattern specification language. Another related work [6] introduces an algorithm for inferring view DTD in MIX mediator. In [6], view DTD is inferring from a view definition and the source DTDs directly and no implicit structures are inferred. Our pattern view DTD is inferred form both sources DTDs and source documents, and the implicit structures matching the pattern are also inferred.

Now a prototype implement of pattern view DTD inference is under developed. And other issues about view DTDs (e.g. the incremental maintenance) are also under explored.

References

1. T.Bray, J.Paoli, and C.Sperberg-McQueen. Extensible Markup Language (XML) 1.0. World Wide Web Consortium Recommendation. Available at http://www.w3.org/TR/REC-xml, Feb. 1998.
2. A.Deutsch, M.Fernandez, D.Florescu, A.Levy, and D.siciu. XML-QL: A Query Language for XML. Available at http://www.w3.org/TR/NOTE-xml-ql.
3. R.Goldman and J.Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In Proc. VLDB, pages 436-445, 1997.
4. S.Nestorov, S.Abiteboul, R.Motwani. Inferring Schema from Semistructured Data. In Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, May 1997.
5. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. In Proc. ICDE, pages 251-260, 1995.
6. Y.Papakonstantinou, V.Vassalos. Enhancing Semistructured Data Mediators with Document Type Definitions. In Proc. ICDE, 1999.
7. K.Wang, H.Liu. Schema Discovery for Semistructured Data. In Proc. KDD, 1997.

An Object Oriented Multidimensional Data Model for OLAP

T. B. Nguyen, A M. Tjoa, R.R. Wagner

Institute of Software Technology (E188), Vienna University of Technology
Favoritenstrasse 9-11/188, A-1040 Vienna, Austria
{binh,tjoa}@ifs.tuwien.ac.at
Institute of Applied Knowledge Processing, University of Linz
Altenberger Strasse 69, A-4040 Linz, Austria
wagner@ifs.uni-linz.ac.at

Abstract. Online Analytical Processing (OLAP) data is frequently organized in the form of multidimensional data cubes each of which is used to examine a set of data values, called measures, associated with multiple dimensions and their multiple levels. In this paper, we first propose a conceptual multidimensional data model, which is able to represent and capture natural hierarchical relationships among members within a dimension as well as the relationships between dimension members and measure data values. Hereafter, dimensions and data cubes with their operators are formally introduced. Afterward, we use UML (Unified Modeling Language) to model the conceptual multidimensional model in the context of object oriented databases.

1. Introduction

Data warehouses and OLAP are essential elements of decision support [5], they enable business decision makers to creatively approach, analyze and understand business problems [16]. While data warehouses are built to store very large amounts of integrated data used to assist the decision-making process [9], the concept of OLAP, which is first formulated in 1993 by [6] to enable business decision makers to work with data warehouses, supports dynamic synthesis, analysis, and consolidation of large volumes of multidimensional data [7]. OLAP systems organize data using the multidimensional paradigm in the form of data cubes, each of which is a combination of multiple dimensions with multiple levels per dimension. Summarized data is pre-aggregated and stored with the main purpose to explore the relationship between independent, static variables, *dimensions*, and dependent, dynamic variables, *measures* [3]. Moreover, dimensions always have structures and are linguistic categories that describe different ways of looking at the information [4]. These dimensions contain one or more natural hierarchies, together with other attributes that do not have a hierarchy's relationship to any of the attributes in the dimensions [10]. Having and handling the predefined hierarchy or hierarchies within dimensions provide the foundation of two typical operations like *rolling up* and *drilling down*. Because unbalanced and multiple hierarchical structures (Fig. 1,2) are the common structures of dimensions, the two current OLAP technologies, namely ROLAP and MOLAP, have limitations in the handling of dimensions with these structures [15].

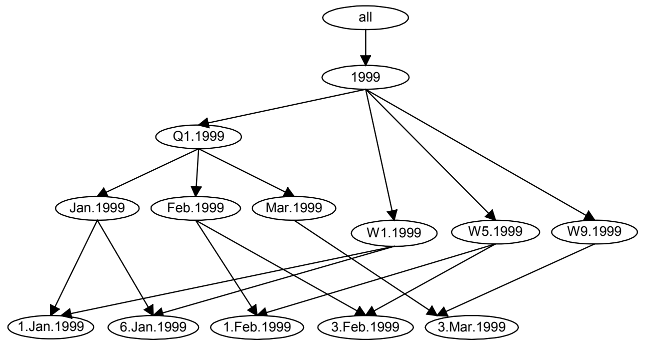


Fig. 1. An instance of the dimension *Time* with unbalanced and multiple hierarchical structure

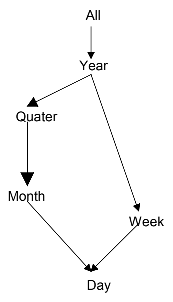


Fig. 2. A schema of the dimension *Time* with multihierarchical structure.

ROLAP (Relational OLAP) products are set on top of existing relational database management systems (RDBMS), which are well standardized and meet the needs of storing large amounts of data. Dimensions and facts are mapped into relational tables, called fact and dimension tables, organized as *Star Schema* and/or *Snowflake Schema* [10]. Therefore in many cases, ROLAP products are not suitable for handling dimensions with multihierarchical and unbalanced structures. Furthermore, existing relational query languages (e.g. SQL) are not sufficiently powerful or flexible enough to support true OLAP capabilities [19]. [3] clearly demonstrated the mismatch between multidimensional operations and SQL.

Although MOLAP (Multidimensional OLAP) easily supports dimensions with multiple and unbalanced hierarchical structures and MOLAP queries are very powerful and flexible in terms of OLAP processing [14], there are still several challenges for these products. First, the underlying data structures are limited in their ability to support multiple subject areas and to provide access to detailed data. Navigation and analysis of data is limited because the data is designed according to previously determined requirements [7]. In addition, with products that require complete pre-calculation, the dimensional explosion could result in physical database that is unmanageable [14].

The first goal of this paper is the introduction of a conceptual multidimensional data model that facilitates a precise rigorous conceptualization for OLAP. First, the model is able to represent and capture natural hierarchical relationships among members within a dimension. Therefore, dimensions with complex structures, such as: unbalanced and multihierarchical structures, can be handled. Moreover, the data model is able to represent the relationships between dimension members and measure data values by mean of cube cells. Hereafter, the data cubes, which are basic components in multidimensional data analysis, are formally introduced. Furthermore, cube operators (e.g. *jumping*, *rollingUp* and *drillingDown*) are defined in a very elegant manner.

The second goal is the modeling of the conceptual multidimensional data model in term of classes by using UML. Based on the formal representation of the class specifications in UML, the design and implementation of the data model for object oriented databases are straightforward.

The remainder of this paper is organized as follows. In section 2, we discuss about related works. Then in section 3, we introduce a conceptual data model that will be mapped into object-oriented database by means of UML in section 4. The paper concludes with section 5, which presents our current and future works.

2. Related works

Since Codd's [6] formulated the term Online Analytical Processing (OLAP) in 1993, many commercial products, like Arborsoft (now Hyperion) Essbase, Cognos Powerplay or MicroStrategy's DSS Agent have been introduced on the market [2]. But unfortunately, sound concepts were not available at the time of the commercial products being developed. The scientific community struggles hard to deliver a common basis for multidimensional data models ([1], [4], [8], [11], [12], [13], [21]). The data models presented so far differ in expressive power, complexity and formalism. In the followings, some research works in the field of data warehousing systems and OLAP tools are summarized.

In [12] a multidimensional data model is introduced based on relational elements. Dimensions are modeled as "dimension relations", practically annotating attributes with dimension names. The cubes are modeled as functions from the Cartesian product of the dimensions to the measure and are mapped to "grouping relations" through an applicability definition.

In [8] n-dimensional tables are defined and a relational mapping is provided through the notation of completion. Multidimensional database are considered to be composed from set of tables forming denormalized star schemata. Attribute hierarchies are modeled through the introduction of functional dependencies in the attributes of dimension tables.

[4] modeled a multidimensional database through the notations of dimensions and f-tables. Dimensions are constructed from hierarchies of dimension levels, whereas f-tables are repositories for the factual data. Data are characterized from a set of roll-up functions, mapping the instance of a dimension level to instances of other dimension level.

In statistical databases, [17] presented a comparison of work done in statistical and multidimensional databases. The comparison was made with respect to application areas, conceptual modeling, data structure representation, operations, physical organization aspects and privacy issues.

In [3], a framework for Object-Oriented OLAP is introduced. Two major physical implementations exist today: ROLAP and MOLAP and their advantages and disadvantages due to physical implementation were introduced. The paper also presented another physical implementation called O3LAP model.

[20] took the concepts and basic ideas of the classical multidimensional model based on the Object-Oriented paradigm. The basic elements of their Object Oriented

Multidimensional Model are dimension classes and fact classes. They also presented cube classes as the basic structure to allow a subsequent analysis of the data stored in the system.

In this paper, we address a suitable mutidimensional data model for OLAP. The main contributions are: (a) the introduction of a formal multidimensional data model; (b) the very elegant manners of definitions of three cube operators, namely *jumping*, *rollingUp* and *drillingDow*n; (c) the modeling of the conceptual multidimensional data model in term of classes by using UML.

3. A Conceptual Data Model

In our approach, a multidimensional data model is constructed based on a set of dimensions $\mathcal{D} = \{\mathrm{D}_1,..,\mathrm{D}_x\}, x \in \mathbf{N}$, a set of measures $\mathcal{M} = \{\mathrm{M}_1,..,\mathrm{M}_y\}, y \in \mathbf{N}$ and a set of data cubes $\mathcal{C} = \{\mathrm{C}_1,..,\mathrm{C}_z\}, z \in \mathbf{N}$. The following sections formally introduce the descriptions of dimensions with their structures, measures and data cubes.

3.1. The Concepts of Dimensions

First, we introduce hierarchical relationships among dimension members by means of one hierarchical domain per dimension. A hierarchical domain is a set of dimension members, organized in hierarchy of levels, corresponding to different levels of granularity. It allows us to consider a dimension schema as a partially ordered set of levels. In this concept, a hierarchy is a path along the dimension schema, beginning at the root level and ending at a leaf level. Moreover, the recursive definitions of two dimension operators, namely *ancestor* and *descendant*, provide abilities to navigate along a dimension structure. In a consequence, dimensions with any complexity in their structures can be captured with this data model.

Definition 3.1.1. [*Dimension Hierarchical Domain*] A hierarchical domain of a dimension D is a non-empty set and denoted by $dom(\mathrm{D}) = \{all\} \cup \{dm_1,..,dm_n\}$, where:

- Each dimension member dm_i is a data item within a dimension. E.g. *1999*, *Q1.1999*, *Jan.1999*, and *1.Jan.1999*, etc are dimension members within the dimension *Time* (Fig. 1).
- Such that the graph $G_{\prec_M} = (V,E)$, defined as the representation over the binary relation over the $dom(\mathrm{D})$, is a tree and defined as follows:

  $V = dom(\mathrm{D})$,

  $E \subset dom(\mathrm{D}) \times dom(\mathrm{D})$. $\forall (dm_i, dm_j) \in E : dm_i \prec_M dm_j$ is an edge in $G_{\prec_M}$. The edge is given when there is an ordered relationship in the sense of hierarchy.
- And the two operators {+,-}: $\forall dm_i \in dom(\mathrm{D})$:

  $-(dm_i) = \{dm_j \in dom(\mathrm{D}) : dm_j \prec_M dm_i\}$

$+(dm_i) = \{dm_k \in dom(\mathrm{D}) : dm_i \prec_M dm_k\}$

- The all or root member: $(\exists! all \in dom(\mathrm{D}))(\neg\exists dm \in dom(\mathrm{D}) : dm \prec_M all)$.
- Leaf members: $(\forall dm_i \in dom(\mathrm{D}))(\neg\exists dm_j \in dom(\mathrm{D}), i \neq j : dm_i \prec_M dm_j)$.

Example: Figure 1 shows a representation in tree term of the dimension *Time*. Hereafter, we have:

dom(*Time*)={*all,1999,Q1.1999,..,3.Mar.1999*},

$all \prec_M 1999, 1999 \prec_M Q1.1999, ..., Mar.1999 \prec_M 3.Mar.1999$,

-(*1999*)=*all*; +(*1999*)={*Q1.1999,W1.1999,W5.1999,W9.1999*}

Definition 3.1.2. [*Dimension Levels*] Let $Levels(\mathrm{D}) = All \cup \{l_1,..,l_h\}, h \in \mathbf{N}$ be a finite set of levels of a dimension D, where:

- The collection of subsets $\{dom(l_1),..,dom(l_h)\}$ is a partition of *dom*(D),
- The *All* or root level: $\exists! All \in Levels(\mathrm{D}) : dom(All) = \{all\}$,
- Leaf levels: $\{l_i \in Levels(\mathrm{D}) \mid \forall dm_j \in dom(l_i) : dm_j \text{ a } leafmember\}$.

Example: The dimension *Time* has three levels *Levels*(*Time*)={*All,Year,Quarter, Month,Week,Day*}. And:

dom(*All*)= {*all*}, *dom*(*Year*)={*1999*}, *dom*(*Quarter*)= {*Q1.1999*}
dom(*Month*)= {*Jan.1999,Feb.1999,Mar.1999*},
dom(*Week*)={*W1.1999,W5.1999,W9.1999*},
dom(*Day*)= {*1.Jan.1999,6.Jan.1999,1.Feb.1999,3.Feb.1999,3.Mar.1999*}

Definition 3.1.3. [*Dimension Schema*] A schema of a dimension D, denoted by $DSchema(\mathrm{D}) = \langle Levels(\mathrm{D}), \prec_L \rangle$, is a partially ordered set of levels:

- *Levels*(D) is a finite set of dimension levels,
- And $\prec_L$ is an ordered relation over the levels and satisfies the following condition:

  $l_i \prec_L l_j$ if $(\exists dm_t \in dom(l_i))$ *and* $(\exists dm_u \in dom(l_j)) : dm_t \prec_M dm_u$.

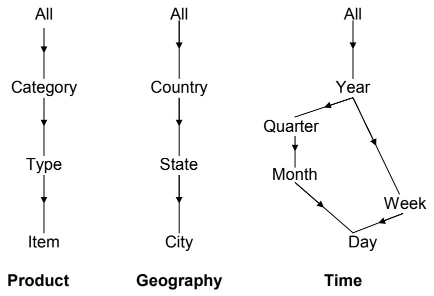


Fig. 3. Schemas of three dimensions *Product, Geography* and *Time*

Example: Figure 3 is used to describe schemas of three dimensions *Product, Geography*, and *Time*.

DSchema(*Product*)={$All \prec_L Category, Category \prec_L Type, Type \prec_L Item$}

DSchema(*Geography*)={*All* $\prec_L$ *Country, Country* $\prec_L$ *State, State* $\prec_L$ *City*}

DSchema(*Time*)={*All* $\prec_L$ *Year* , *Year* $\prec_L$ *Quarter, Quarter* $\prec_L$ *Month, Month* $\prec_L$ *Day* , *All* $\prec_L$ *Year, Year* $\prec_L$ *Week, Week* $\prec_L$ *Day*}

Definition 3.1.4. [*Dimension Path*] A path within a dimension schema is a linear, totally ordered list of levels and can be defined as follows:

$\forall l_i, l_j \in Levels(\mathrm{D})$:

$$path(l_i,l_j) = \begin{cases} \{l_i \prec_L l_j\} & \text{If } l_i \prec_L l_j \\ \{l_i \prec_L l_t,..,l_u \prec_L l_j\} & \text{Else if } \exists l_t,..,l_u \in Levels(\mathrm{D}) : l_i \prec_L l_t,..,l_u \prec_L l_j \\ \phi & \text{Else} \end{cases}$$

Definition 3.1.5. [*Dimension Hierarchy*] A hierarchy is defined by a $path(All, l_{leaf})$ within the schema of a dimension D. The path begins at *All* (*root*) level and ends at a leaf level.

Let $H(\mathrm{D}) = \{h_1,..,h_m\}, m \in \mathbf{N}$ be a set of hierarchies of a dimension D. If *m*=1 then the dimension has single hierarchical structure, else the dimension has multihierarchical structure.

Definition 3.1.6. [*Dimension Operators*] Two dimension operators (DO), namely *ancestor* and *descendant*, are defined recursively as follows:

$\forall l_i, l_a, l_d \in Levels(\mathrm{D})$ and $\forall dm \in dom(l_i) \subset dom(\mathrm{D})$:

$$ancestor(dm, l_a, \mathrm{D}) = \begin{cases} undefined & \text{If } (path(l_a, l_i) = \phi) \\ dm^- \in dom(l_a) : dm^- \in -(dm) & \text{Else If } (l_a \prec_L l_i) \\ ancestor(dm^-, l_a, \mathrm{D}), \text{where :} & \text{Else} \\ dm^- = ancestor(dm, l_p, \mathrm{D}) : l_p \prec_L l_i & \end{cases}$$

$$descendant(dm, l_d, \mathrm{D}) = \begin{cases} undefined & \text{If } (path(l_i, l_d) = \phi) \\ \{dm^+ \in dom(l_d) : dm^+ \in +(dm)\} & \text{Else If } (l_i \prec_L l_d) \\ descendant(dm^+, l_d, \mathrm{D}), \text{where :} & \text{Else} \\ dm^+ \in descentdant(dm, l_n, \mathrm{D}) : l_i \prec_L l_n & \end{cases}$$

Example: *ancestor*(*Q1.1999,Year,Time*)=*1999*,
descendant(*Q1.1999,Month,Time*)= {*Jan.1999,Feb.1999,Mar.1999*},

3.2. The Concepts of Measures

In this section we introduce measures, which are the objects of analysis in the context of multidimensional data model.

Definition 3.2.1. [*Measure Schema*] A schema of a measure M is a tuple $MSchema(\mathrm{M}) = \langle Fname, \mathrm{O} \rangle$, where:

- *Fname* is a name of a corresponding fact,
- $\mathrm{O} \in \Omega \cup \{\mathrm{NONE}, \mathrm{COMPOSITE}\}$ is an operation type applied to a specific fact [2]:
  - Ω={SUM, COUNT, MAX, MIN} is a set of aggregation functions,
  - COMPOSITE is an operation (e.g. average), where measures cannot be utilized in order to automatically derive higher aggregations,
  - NONE measures are not aggregated. In this case, the measure is the fact.

Definition 3.2.2. [*Measure Domain*] Let $\mathcal{N}$ be a numerical domain where a measure value is defined (e.g. **N**, **Z**, **R**). The domain of a measure M is a subset of $\mathcal{N}$. We denote by $dom(\mathrm{M}) \subset \mathcal{N}$.

3.3. The Concepts of Data Cubes

A multidimensional cube is constructed based on a set dimensions and a set of measures, and consists a collection of cells. Each cell is an intersection among a set of dimension members and measure data values. Furthermore, cells are grouped into granular groupbys, each of which expresses a mapping from the domains of *x*-tuple of dimension levels (independent variables) to *y*-numerical domains of *y*-tuple of numeric measures (dependent variables).

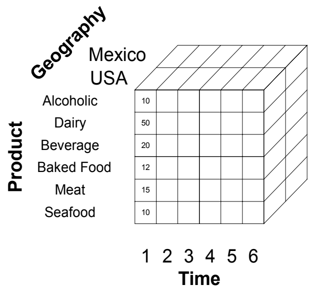


Fig. 4. *Sale* cube includes dimensions: *Geography, Product* and *Time* and a fact: *Sale amount*.

Given *x* dimensions $\mathrm{D}_1,..,\mathrm{D}_x, x \in \mathbf{N}$, and *y* measures $\mathrm{M}_1,..,\mathrm{M}_y, y \in \mathbf{N}$.

Definition 3.3.1. [*Cube Schema*] A cube schema is tuple $CSchema(\mathrm{C}) = \langle Cname, DSchemas, MSchemas \rangle$:

- *Cname* is the name of a cube,
- *DSchemas* are the schemas of *x* dimensions, denoted by $DSchemas =< DSchema(\mathrm{D}_1),..,DSchema(\mathrm{D}_x) >$,
- *MSchemas* are the schemas of *y* measures, denoted by $MSchemas =< MSchema(\mathrm{M}_1),..,MSchema(\mathrm{M}_y) >$.

Definition 3.3.2. [*Cube Domain*] Given a function:

$$f : dom(\mathrm{D}_1)\times..\times dom(\mathrm{D}_x)\times dom(\mathrm{M}_1)\times..\times dom(\mathrm{M}_y) \to \{true, false\},$$

A cube domain, denoted by $dom(\mathrm{C}) = \{c_1,..,c_k\}, k \in \mathbf{N}$ is determined as follows:

$$dom(\mathrm{C}) = \{c = (dms, fms) | dms \in dom(\mathrm{D}_1)\times..\times dom(\mathrm{D}_x),$$
$$fms \in dom(\mathrm{M}_1)\times..\times dom(\mathrm{M}_y) : f(dms, fms) = true\}$$

3.4. Operating Group by, Rolling Up, Drilling Down in our model

Let a cube C be constituted from x dimensions $\mathrm{D}_1,..,\mathrm{D}_x, x\in \mathbf{N}$, and y measures $\mathrm{M}_1,..,\mathrm{M}_y, y\in \mathbf{N}$. We define groupby and three operators, namely *jumping*, *rollingUp* and *drillingDown* as follows:

Definition 3.4.1. [*Groupby*] A groupby is triple G=$\langle Gname, GSchema(\mathrm{G}), dom(\mathrm{G})\rangle$ where:

- *Gname* is the name of this groupby,
- $GSchema(\mathrm{G})=\langle GLevels(\mathrm{G}), GMSchemas(\mathrm{G})\rangle$:

  $GLevels(\mathrm{G}) =< l_{\mathrm{D}_1},.., l_{\mathrm{D}_x} >\in Levels(\mathrm{D}_1)\times..\times Levels(\mathrm{D}_x)$ is a x-tuple of levels of the x dimensions $\mathrm{D}_1,..,\mathrm{D}_x, x\in \mathbf{N}$.

  $GMSchemas(\mathrm{G}) =< MSchema(\mathrm{M}_1),.., MSchema(\mathrm{M}_y) >$ is a y-tuple of measure schemas of the y measures $\mathrm{M}_1,..,\mathrm{M}_y, y\in \mathbf{N}$.
- $dom(\mathrm{G}) = \{c =< dms, fms >\in dom(\mathrm{C}) \mid dms \in dom(l_{\mathrm{D}_1})\times..\times dom(l_{\mathrm{D}_x}),$

  $fms \in dom(\mathrm{M}_1)\times..\times dom(\mathrm{M}_y)\}$

Let h_i be a number of levels of each dimension D_i ($1\le i\le x$). The total set of groupbys over a cube C is defined as $Groupbys(\mathrm{C}) = \{\mathrm{G}_1,..,\mathrm{G}_p\}, p = \prod_{i=1}^{x} h_i$ [18].

Definition 3.4.2. [*Cube Operators*] The three basic navigational cube operators (*CO*), namely *jumping*, *rollingUp* and *drillingDown*, which are applied to navigate along a data cube C, corresponding to a dimension D_i, are defined as follows:

Given a current groupby G_c, associated with a level l_c of a dimension D_i, and three other levels $l_j, l_r, l_d \in Levels(\mathrm{D}_i)$.

- *Jumping:*

  $$jumping(\mathrm{G}_c, l_j, \mathrm{D}_i) = \mathrm{G}_j =< GLevels(\mathrm{G}_j), GMSchemas(\mathrm{G}_j) >$$

  Where:

  $$GMSchemas(\mathrm{G}_j) = GMSchemas(\mathrm{G}_c),$$
  $$GLevels(\mathrm{G}_j)(i) = l_j,\ GLevels(\mathrm{G}_j)(k) = GLevels(\mathrm{G}_c)(k), \forall k \neq i.$$
- *Rolling Up:* $\forall dm \in dom(l_c)$, $\mathrm{G}_r = jumping(\mathrm{G}_c, l_r, \mathrm{D}_i)$:

$$rollingUp(\mathrm{G}_c, dm, l_r, \mathrm{D}_i) = \mathrm{G}_r^{sub} =< GSchema(\mathrm{G}_r^{sub}), dom(\mathrm{G}_r^{sub}) >$$

Where:

$GSchema(\mathrm{G}_r^{sub}) = GSchema(\mathrm{G}_r)$,

$dom(\mathrm{G}_r^{sub}) = \{c_r \in dom(\mathrm{G}_r) \mid \exists c \in dom(\mathrm{G}_c) : c.dms(i) = dm,$

$c_r.dms(i) = ancestor(dm, l_r, \mathrm{D}_i)$, $c_r.dms(j) = c.dms(j), \forall j \neq i\}$

- *Drilling Down:* $\forall dm \in dom(l_c)$, $\mathrm{G}_d = jumping(\mathrm{G}_c, l_d, \mathrm{D}_i)$:

$$drillingDown(\mathrm{G}_c, dm, l_d, \mathrm{D}_i) = \mathrm{G}_d^{sub} =< GSchema(\mathrm{G}_d^{sub}), dom(\mathrm{G}_d^{sub}) >$$

Where:

$GSchema(\mathrm{G}_d^{sub}) = GSchema(\mathrm{G}_d)$,

$dom(\mathrm{G}_d^{sub}) = \{c_d \in dom(\mathrm{G}_d) \mid \exists c \in dom(\mathrm{G}_c) : c.dms(i) = dm,$

$c_d.dms(i) \in descendant(dm, l_d, \mathrm{D}_i)$, $c_d.dms(j) = c.dms(j), \forall j \neq i\}$

4. Modeling Multidimensional Data Model

In this section, UML is used to model dimensions, measures and data cubes in context of an object oriented data model. All conceptual components, which are introduced in section 3, are mapped as classes. Figure 5 illustrates the modeling for our data model in term of class diagrams by using UML.

4.1. The Modeling of Dimensions

The dimension concepts, such as: dimension members, levels, dimension schemas, hierarchy and dimension, are modeled in term of class diagrams by using UML. First, a hierarchical domain of dimension members within a dimension is handled by means of the *DMember* class. The two dimension member operators {+} and {-} are mapped into the two methods *getFathers* and *getChildren*, built-in every instance of this class. Hereafter, The *Level*, *DSchema*, *Hierarchy* classes are defined to describe the concepts of level, dimension schema and dimension hierarchy. Afterwards, each instance of the *Dimension* class describes a dimension. The dimension operators, namely *ancestor* and *descendant* are mapped as two methods with the same names.

4.1.1. *DMember.* Each instance of this class describes a dimension member within a hierarchical domain of a dimension.

- *Attributes:*
  description (String) - That is a data item within a dimension.
- *Relationships:*
  fathers (Set<*DMember*>) - A set of referred *DMember* objects.
  children (Set<*DMember*>) - A set of referred *DMember* objects
- *Main methods:*
  getFathers() (return Set<*DMember*>) - This method describes the operator {-}.
  getChildren() (returns Set<*DMember*>) - This method describes the operator {+}.

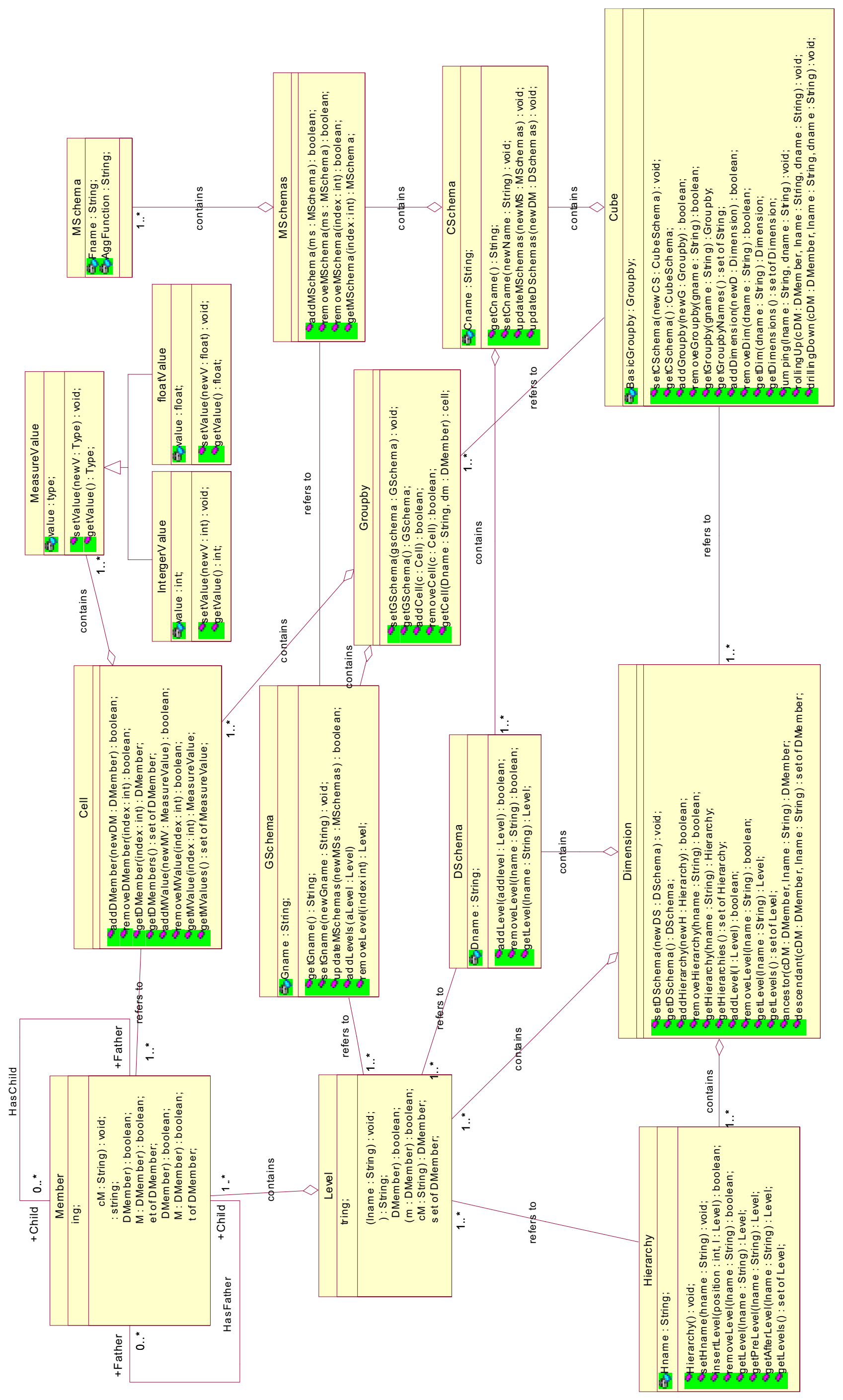

MSchema
Fname : String;
AggFunction : String;
MSchemas
addMSchema(ms : MSchema) : boolean;
removeMSchema(ms : MSchema) : boolean;
removeMSchema(index : int) : boolean;
getMSchema(index : int) : MSchema;
CSchema
Cname : String;
getCname() : String;
setCname(newName : String) : void;
updateMSchemas(newMS : MSchemas) : void;
updateDSchemas(newDM : DSchemas) : void;
Cube
BasicGroupby : Groupby;
setCSchema(newCS : CubeSchema) : void;
getCSchema() : CubeSchema;
addGroupby(newG : Groupby) : boolean;
removeGroupby(gname : String) : boolean;
getGroupby(gname : String) : Groupby;
getGroupbyNames() : set of String;
addDimension(newD : Dimension) : boolean;
removeDim(dname : String) : boolean;
getDim(dname : String) : Dimension;
getDimensions() : set of Dimension;
Jumping(lname : String, dname : String) : void;
rollingUp(cDM : DMember, lname : String, dname : String) : void;
drillingDown(cDM : DMember, lname : String, dname : String) : void;
MeasureValue
value : type;
setValue(newV : Type) : void;
getValue() : Type;
floatValue
value : float;
setValue(newV : float) : void;
getValue() : float;
IntergerValue
value : int;
setValue(newV : int) : void;
getValue() : int;
Groupby
setGSchema(gschema : GSchema) : void;
getGSchema() : GSchema;
addCell(c : Cell) : boolean;
removeCell(c : Cell) : boolean;
getCell(Dname : String, dm : DMember) : cell;
Cell
addDMember(newDM : DMember) : boolean;
removeDMember(index : int) : boolean;
getDMember(index : int) : DMember;
getDMembers() : set of DMember;
addMValue(newMV : MeasureValue) : boolean;
removeMValue(index : int) : boolean;
getMValue(index : int) : MeasureValue;
getMValues() : set of MeasureValue;
GSchema
Gname : String;
getGname() : String;
setGname(newGname : String) : void;
updateMSchemas(newMSs : MSchemas) : boolean;
addLevels(aLevel : Level)
removeLevel(index int) : Level;
DSchema
Dname : String;
addLevel(addlevel : Level) : boolean;
removeLevel(lname : String) : boolean;
getLevel(lname : String) : Level;
Dimension
setDSchema(newDS : DSchema) : void;
getDSchema() : DSchema;
addHierarchy(newH : Hierarchy) : boolean;
removeHierarchy(hname : String) : boolean;
getHierarchy(hname : String) : Hierarchy;
getHierarchies() : set of Hierarchy;
addLevel(l : Level) : boolean;
removeLevel(lname : String) : boolean;
getLevel(lname : String) : Level;
getLevels() : set of Level;
ancestor(cDM : DMember, lname : String) : DMember;
descendant(cDM : DMember, lname : String) : set of DMember;
Member
ing;
cM : String) : void;
: string;
DMember) : boolean;
M : DMember) : boolean;
et of DMember;
DMember) : boolean;
M : DMember) : boolean;
t of DMember;
Level
tring;
(lname : String) : void;
) : String;
DMember) : boolean;
(m : DMember) : boolean;
cM : String) : DMember;
set of DMember;
Hierarchy
Hname : String;
Hierarchy() : void;
setHname(hname : String) : void;
insertLevel(position : int, l : Level) : boolean;
removeLevel(lname : String) : boolean;
getLevel(lname : String) : Level;
getPreLevel(lname : String) : Level;
getAfterLevel(lname : String) : Level;
getLevels() : set of Level;
contains
refers to
1..*
0..*
+Child
+Father
HasChild
HasFather

Fig.5. The modeling of the object oriented multidimensional data model

4.1.2. Level. Each instance of the *Level* class describes a level of a dimension.
Attributes:
dmembers (Set<*DMember*>) - A set of contained *DMember* objects.

4.1.3. DSchema. Each instance of the *DSchema* describes a dimension schema.

- *Attributes:*
  dname (*String*) - That is a dimension name.
- *Relationships:*
  levels (Set<*Level*>) - A set of Level objects.

4.1.4. Hierarchy. Each instance of the Hierarchy class describes a hierarchy of levels within a dimension.

- *Attributes*:
  dname (String) - That is a hierarchy name
- *Relationships*:
  levels (Set<*Level*>) - A set of *Level* objects

4.1.5. Dimension. Each instance of this class describes a dimension.

- *Attributes:*
  dschema (*DSchema*) - A *DSchema* object.
  hierarchies (Set<*Hierarchy*>) - A set of *Hierarchy* objects.
  levels (Set<*Level*>) - A set of *Level* objects.
- *Main methods:*
  ancestor(*DMember* cDM;*lname*:String) (returns *DMember*): *ancestor* operator.
  descendant(*DMember* cDM,*lname*:String) (Set<*DMember*>): *descendant* operator.

4.2. The Modeling of Measures

Measure schemas and measure data values are mapped into classes. First, an *MSchema* describes a measure schema. Afterwards, The *MValue* is an abstract type that serves as a common super type for measure values. It is obvious that the two classes, namely *intMValue* and *floatValue*, are subclasses of *MValue*.

4.2.1. MSchema. Each instance of this class describes a measure schema.

- *Attributes:*
  fname (String) - That is a fact name.
  aggFunction (String)- An aggregation function, such as *Max, Min, Sum, Count, None* and *Composite*.

4.2.2. MValue. *MValue* is an abstract type that serves as common super type for measure values.
Attributes:
value (*Type*) – That describes a measure value.

4.2.3. intMValue. The *intMValue* is a subclass of *MValue*. Each instance of this class describes an integer measure value.

- *Specializes:*
  MValue: The *intMValue* class inherits from *MValue* class

- *Attributes:*
  value (int) – The value is overridden as integer.

4.2.4. floatMValue. The *floatMValue* is a subclass of *MValue*. Each instance of this class describes a float measure value.

- *Specializes:*
  MValue: The *floatMValue* class inherits from *MValue* class
- *Attributes:*
  value (float) – The value is overridden as float.

4.3. The Modeling of Multidimensional Components

First, each instance of the *CSchema*, which contains an *x*-tuple of *DSchemas* and a *y*-tuple of *MSchemas*, describes a cube schema. Then, a *Cell* refers to *x DMembers* of *x* dimensions and *y MValues* of *y* measures. In addition, a *GSchema* contains *x Level*s of *x* dimensions and the *y*-tuple *MSchemas*. Afterwards, a *Groupby* contains a *GSchema* and a subset of *Cell*s. As a consequence, a *Cube* contains a *CSchema* and a *BasicGroupby* (*Groupby*), is associated with a set of *Dimension*s, and a set of *Groupby*s. The three methods, i.e. *jumping*, *rollingUp* and *drillingDown*, are the mappings of the three operators with the same names.

4.3.1. CSchema. Each instance of the *CSchema* describes a cube schema.

- *Attributes:*
  dschemas (Set<*DSchema*>) - A set of *x DSchema* objects,
  mschemas (Set<*MSchema*>) - A set of *y MSchema* objects.

4.3.2. Cell. Each instance of this class describes a cube cell.

- *Relationships:*
  dmembers (Set<*DMember*>) - A set of *x DMember* objects.
  mvalues (Set<*MValue*>) - A set of *y MValue* objects.

4.3.3. GSchema. Each instance of the *GSchema* class describes a groupby schema.

- *Relationships:*
  levels (Set<*Level*>) - A set of *x Level* objects.
  mschemas (Set<*MSchema*>) - A set of *y MSchema* objects.

4.3.4. Groupby. Each instance of this class describes a groupby.

- *Attributes:*
  gschema (*GSchema*): A *GSchema* object that describes a groupby schema.
  cells (Set<*Cell*>): A set of *Cell* objects.

4.3.5. Cube. Each instance of the *Cube* class describes a data cube.

- *Attributes:*
  cschema (*CSchema*) – A *CSchema* describes a cube schema of a cube.
  basicgroupby (*Groupby*) - A *Groupby* object.
- *Relationships:*
  dimensions (Set<*Dimension*>) - *Dimensions* express for *x* dimensions of a cube.
  groupbies (Set<*Groupby*>) – A set of *Groupby* objects,

- *Main methods:*
  jumping(*lname*:String;*dname*:String) (void) - *jumping* operator.
  rollingUp(*cDM*:*DMember*;*lname*:String;*dname*:String) (void) - *rollingUp* operator.
  drillingDown(*cDM*:*DMember*;*lname*:String;*dname*:String) (void) - *drillingDown* operator.

5. Conclusion and future works

In this paper, we have introduced the conceptual multidimensional data model, which facilitates even sophisticated constructs based on multidimensional data units or members such as dimension members, measure data values and then cells. The model is able to represent and capture natural hierarchical relationships among dimension members. Dimensions with complexity of their structures, such as: unbalanced and multihierarchical structures, can be modeled in an elegant and consistent way. Moreover, the data model represents the relationships between dimension members and measure data values by mean of cube cells. In consequence, data cubes, which are basic components in multidimensional data analysis, and their operators are formally introduced in a very elegant manner. We have also proposed a modeling of the conceptual multidimensional data model in term of classes by means of UML, which is an object oriented standard analysis and design notation.

In context of future works, we are investigating two approaches for implementation: pure object-oriented orientation and object-relational approach. With the first model, dimensions and cube are mapped into an object-oriented database in term of classes. In the other alternative, dimensions, measure schema, and cube schema are grouped into a term of metadata, which will be mapped into object-oriented database in term of classes. Some useful methods built in those classes are used to give the required Ids within those dimensions. The given Ids will be joined to the fact table, which is implemented in relational database.

Acknowledgment

This work is partly supported by the EU-4th Framework Project GOAL (Geographic Information Online Analysis)-EU Project #977071 and by the ASEAN European Union Academic Network (ASEA-Uninet), Project EZA 894/98.

References

1. Agrawal, R., Gupta, A., Sarawagi, A.: Modeling Multidimensional Databases. IBM Research Report, IBM Almaden Research Center, September 1995.
2. Albrecht, J., Guenzel, H., Lehner, W.: Set-Derivability of Multidimensiona Aggregates. First International Conference on Data Warehousing and Knowledge Discovery. DaWaK'99, Florence, Italy, August 30 - September 1.
3. Buzydlowski, J. W., Song, II-Y., Hassell, L.: A Framework for Object-Oriented On-Line Analytic Processing. DOLAP 1998
4. Cabibbo, L., Torlone, R.: A Logical Approach to Multidimensional Databases. EDBT 1998

5. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record Volume 26, Number 1, September 1997.
6. Codd, E. F., Codd, S.B., Salley, C. T.: Providing OLAP (On-Line Analytical Processing) to user-analysts: An IT mandate. Technical report, 1993.
7. Connolly, T., Begg, C.: Database system: a practical approach to design, implementation, and management. Addison-Wesley Longman, Inc., 1999.
8. Gyssens, M., Lakshmanan, L.V.S.: A foundation for multi-dimensional databases, Proc. VLDB'97.
9. Hurtado, C., Mendelzon, A., Vaisman, A.: Maintaining Data Cubes under Dimension Updates. Proc IEEE/ICDE '99.
10. Kimball, R.: The Data Warehouse Lifecycle Toolkit. John Wiley & Sons, Inc., 1998.
11. Lehner, W.: Modeling Large Scale OLAP Scenarios. 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, 23-27, March 1998.
12. Li, C., Wang, X.S.: A Data Model for Supporting On-Line Analytical Processing. CIKM 1996.
13. Mangisengi, O., Tjoa, A M., Wagner, R.R.: Multidimensional Modelling Approaches for OLAP. Proceedings of the Ninth International Database Conference "Heterogeneous and Internet Databases" 1999, ISBN 962-937-046-8. Ed. J. Fong, Hong Kong, 1999
14. McGuff, F., Kador, J.: Developing Analytical Database Applications. Prentice Hall PTR, 1999.
15. Nguyen, T.B., Tjoa, A M., Wagner, R.R.: *Conceptual Object Oriented Multidimensional Data Model for OLAP*. Technical Report, IFS, Vienna 1999.
16. Samtani, S., Mohania, M.K., Kumar, V., Kambayashi, Y.: Recent Advances and Research Problems in Data Warehousing. ER Workshops 1998.
17. Shoshani, A.: OLAP and Statistical Databases: Similarities and Differences. Tutorials of PODS 1997.
18. Shukla A., Deshpande, P., Naughton, J. F., Ramasamy, K.: Storage Estimation for Multidimensional Aggregates in the Presence of Hierarchies. VLDB 1996: 522-531
19. Thomsen, E.: OLAP solutions: Building Multidimensional Information Systems. John Wiley& Sons, Inc., 1997.
20. Trujillo, J., Palomar, M.: An Object Oriented Approach to Multidimensional Database. Conceptual Modeling (OOMD), DOLAP 1998.
21. Vassiliadis, P.: Modeling Multidimensional Databases, Cubes and Cube operations. In Proc. 10th Scientific and Statistical Database Management Conference (SSDBM '98), Capri, Italy, June 1998.
22. Wang, M., Iyer, B.: Efficient roll-up and drill-down analysis in relational database. In 1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997.

Applying Object-Oriented Conceptual Modeling Techniques to the Design of Multidimensional Databases and OLAP Applications

Juan Carlos Trujillo, Manuel Palomar, and Jaime Gómez

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, Spain
{jtrujillo,mpalomar,jgomez}@dlsi.ua.es

Abstract. Few works have been presented in the area of Object-Oriented (OO) Conceptual Modeling to specify the design of multidimensional databases (MDB) and OLAP applications. In this context, this paper describes an OO conceptual modeling approach (based on a subset of UML) to address the peculiarities associated to this kind of systems. The structure of the system is specified by means of a class diagram that considers the semantics of multidimensional (MD) data models with a minimal use of constraints and extensions on UML. Furthermore, user's requirements are specified as object collections (cube classes) in the class diagram. To do so, new specific graphical elements are defined to represent these classes. The result is an OO conceptual modeling approach that considers the semantics of MD data models as well as user's requirements in the same model in a natural way.

Keywords: conceptual modeling, OLAP, multidimensional databases, UML

1 Introduction

There has recently been an increased interest in how to properly design OLAP [4] scenarios. These applications impose different requirements than On-line Transactional Processing (OLTP) systems, and therefore, different data models and implementation methods are required for each type of system. Unlike OLTP systems where the logical data schema is hidden underneath an application layer, the logical MD schema of an OLAP system is directly used by the end user to formulate queries. Thus, the MD schema is crucial as it determines the type of queries the user can formulate.

Many design methods as well as conceptual models [6,7] are mainly driven by the structure of the operational data sources. Furthermore, they are not able to express all the underlying semantics of MD data models [16], such as the many-to-many relationships between facts and some dimensions or the strictness and completeness of classification hierarchies [12].

In this context, there have lately been some approaches [8,14] that try to apply OO techniques in OLAP areas. These approaches address the design of MD models taking advantage of the benefits provided by the OO paradigm, such as inheritance, code reuse, polymorphism and so on. Unfortunately, these approaches are centered in the solution space (implementation level) and do not provide a modeling language to specify the peculiarities of OLAP applications in the problem space (conceptual model level).

In this paper, therefore, we propose an OO multidimensional conceptual model based on a subset of UML [3,5] that supports the conceptual design of MDB and OLAP applications. Our approach introduces some constraints on UML that are needed for an adequate representation of the semantics of MD data models. Furthermore, it also extends the set of graphical elements necessary for considering the specific properties of the typical OLAP user's requirements. The conclusion is that a minimal adaptation of an OO modeling language (UML) allows us to consider the semantics of MD data models as well as the representation of the user's requirements in the MD conceptual model.

This paper is organized as follows. Section 2 details the major features that must be taken into account in the conceptual design of MD data models. An example that will be handled throughout the paper to present our approach is also explained. Section 3 describes how the proposed model takes advantage of most well-known OO conceptual modeling techniques to represent all the semantics of MD data models presented in section 2. In Section 4, we present the extended elements that have been added to the model to consider the user's requirements. Section 5 compares our work with other approaches, and finally, section 6 draws some conclusions and sketches some works that are currently being carried out.

2 The Classical Multidimensional Model

In this model, a **fact** is an item of interest for an enterprise, and is described through a set of attributes called measures or fact attributes (atomic or **derived**), which are contained in cells or points in the data cube. Moreover, this set of measures are based on a set of **dimensions** that are the granularity adopted for representing facts, (i.e., the context in which facts are to be analyzed). Thus, dimensions are also characterized by attributes that are called dimension attributes.

With reference to measures, the concept of **additivity** or summaribility on measures along dimensions is crucial for MD data modeling [6,2,9] and [18]. A measure is additive along a dimension if the sum operator can be used to aggregate attribute values along all hierarchies defined in that dimension. The aggregation of some fact attributes (roll-up in OLAP terminology), however, might not be semantically meaningful for all measures along all dimensions. In our example, *number of clients* (estimated by counting the number of purchase-receipts for a given *product*, *day* and *store*) is not additive along the product dimension [17]. Since the same ticket may include other products, adding the *number of clients* along two or more products would lead into inconsistent results. However, other aggregation operators could be used (e.g. AVG).

Other relevant features of the model are the **classification hierarchies** defined on dimension attributes along dimensions. A dimension attribute may also be aggregated to more than one dimension attribute, and therefore, **multiple classification hierarchies** and **alternative path hierarchies** are also relevant. For this reason, a common way of representing and considering dimensions with their attributes are by means of Directed Aciclyc Graphs. In Figure 1, we can observe all the above-mentioned properties through an example that will be handled throughout the paper, extracted from [17], consisting of analyzing *sales of products* (facts) by means of the purchased tickets in a great store chain.

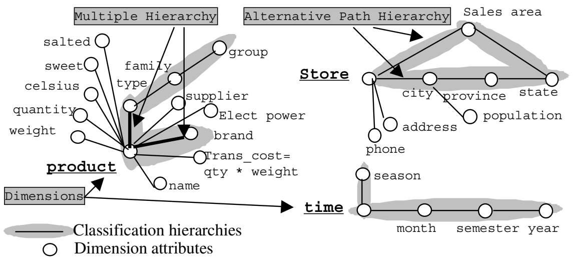


Fig. 1. Dimensions and classification hierarchies

With reference to classification hierarchies, the concepts of **strictness** and **completeness** are quite important, not only for conceptual purposes, but also for further design steps of MD modeling [12,13] and [18]. 'strictness' means that an object of a lower level of a hierarchy belongs to *only* one of a higher level, i.e. a *province* is only related to one *state*. 'Completeness' means that all members belong to one higher-class object and that object consists of those members only, i.e. a *state* may be formed by the *provinces* recorded and by those alone without including any more *provinces*.

One of the main problems in MD data models occurs when the modeled OLAP scenarios become very large since the dimensionality increases significantly, and therefore, this leads to extremely sparse dimensions and data cubes. In this way, there are attributes that are normally valid for all elements within a dimension while others are only valid for a subset of elements. In Figure 1, as observed in the *product* dimension, the attribute *electricity power* should only be relevant for *household products*. Thus, a proper MD data model should be able to consider attributes only when necessary, depending on the **categorization of dimensions**.

Finally, the set of operations (usually called OLAP operations, [9]) generally applied to the MD view of data are as follows: roll-up (increasing the level of aggregation) and drill-down (decreasing the level of aggregation) along one or more classification hierarchies, slice-dice (selection and projection) and pivoting (re-orienting the MD view of data which allows us to exchange dimensions for facts; i.e. **symmetric** treatment of facts and dimensions).

Having identified the major aspects for a proper MD conceptual modeling (facts, dimensions and OLAP operations with their corresponding features), we shall illustrate how these concepts can be addressed by the use of OO conceptual modeling techniques. In our approach, the conceptual model consists of, on one hand, all the semantics of MD data models (presented in section3), and, on the other hand, of the user's requirements (presented in section 4).

3 The OO Conceptual Modeling Approach

Two main aspects must be clearly stated when introducing a conceptual model:

– The conceptual modeling terms that are provided by the model.
– The notation (in our case, based on UML) that is provided to properly represent those conceptual modeling terms.

To do so, in section 3 we present the concepts described in section 2 in a proper order, and describe the items used from UML for a proper OO multidimensional conceptual modeling. We start by considering what we call **fact** classes (FC) (e.g. *Sales of product*) and we define the set of measures as attributes within these classes. We can also explicitly consider **derived measures** as well as their derivation rules within the graphical conceptual schema using derived attributes (see fact class in Figure 2). Secondly, we define **dimensions** as basic classes that are called dimension classes (DC).

In UML, there are two principal kinds of static relationships between types of objects: i.e. associations and subtypes. Associations represent relationships between instances of classes (a *store* is included in a *sales area*; a *sales area* has a number of *stores*). Each association has two roles; each role is a direction on the association and can be explicitly named with a label. A role also has multiplicity (cardinality), which is an indication of how many objects may participate in the given relationship, i.e. it indicates the lower and upper bounds. The cardinalities are as follows: an object A is always associated with one B (*1*), an A is always associated with one or more B (*1..**), an A is associated with zero or one B (*0..1*) and A is associated with zero, one, or more B (***). Moreover, the navigability of an association is an arrow that indicates the other associated objects; for example, an arrow from *sales area* to *stores* would indicate the *stores* that belong to a *sales_area*. If the navigability in an association is only in one direction it is called uni-directional; a bi-directional association contains navigabilities in both directions. In UML, an association with no arrows means that either the navigability is unknown or is bi-directional. Finally, aggregation is a special case of association in which the association relationship between the classes is considered as a "whole-part". Furthermore, shared aggregation is a special case of normal aggregation in which the parts may be parts in any whole.

In our approach, therefore, fact classes are specified as a shared aggregation of dimension classes. The role between facts and dimensions always has a lower bound of *1* to indicate that a fact is always related to all dimensions. We should point out, however, that we can specify the exact cardinality in the shared aggregation as required. Moreover, this shared aggregation is considered bi-directional to provide the required flexibility to treat facts and dimensions symmetrically in further steps of analysis (section 2). In our example, the fact is considered as the purchased tickets in a great store chain, and therefore, each sale can be related to more than one product (see Figure 2).

3.1 Additivity

The aggregation of some fact attributes (roll-up in OLAP terminology) might not be semantically meaningful along all dimensions (e.g. *number_of_people*). For that reason we introduce a definition on fact attributes to consider this.

Definition Let a_i be a fact attribute. We say that a_i is either

– additive if the SUM operator can be used to aggregate the values of a_i along all dimensions,
– semi-additive if it is not additive along one or more dimensions or
– non-additive if it is additive along no dimension.

Nevertheless, this kind of semi-additive or non-additive attributes can still be aggregated by using operators such as count, maximum, minimum or average (section 2). For that reason, a MD conceptual approach should specify which kind of aggregation operators, if any, can be applied to a measure in case that the measure is not additive.

In UML, apart from the constraints inherent in a class diagram, we can explicitly define non strict syntax for describing constraints, putting them inside braces (). Thus, in our approach, all fact attributes are considered additive along all dimensions by default. Semi-additivity is represented explicitly by defining the constraint that specifies the allowed aggregation operators, if any, for each semi- or non-additive attribute with the dimension along which values cannot be added. We use informal English to define the constraint and place it somewhere around the fact class as can be seen in Figure 2.

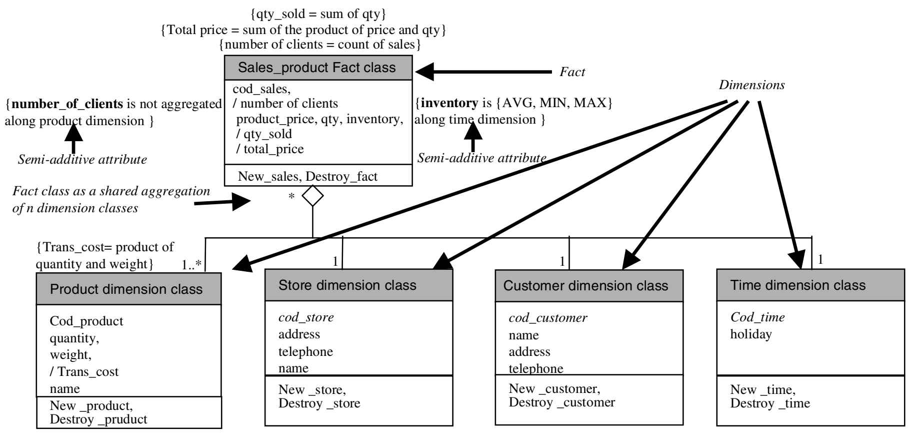


Fig. 2. A fact class as a shared aggregation of 'n' dimension classes

3.2 Expressing Classification Hierarchies

Dimensions are generally structured in classification hierarchies to address the further analysis of data. In our approach, we consider each level of a classification hierarchy as a basic class.

Thus, we define the relationships between two levels of a classification hierarchy as an association of classes. The only constraint required is that the classes used for defining classification hierarchies along a dimension must define a Directed Acyclic Graph (DAG), rooted in the dimension class, as follows: DAG=(C,V) with C being the finite set of classes c_1, c_2, .., c_k and $V=(c_i,c_j) \mid i \neq j \wedge 1 \leq i, j \leq k \wedge c_j$ is an association of $c_i \wedge c_1$ is the dimension class. Thus, each class (c_i) can be considered as a level (l_i) of the hierarchy. Due to the special semantics of classification hierarchies, no cycles must be contained in the graph as this could lead to semantically not reasonable infinite roll-up paths (i.e. *city* rolls-up to *province* and *province* rolls-up to *city*). In this context, the rolls-up relationship is considered as follows: c_1 rolls-up to c_2 means that c_2 is an association of c_1. The name of the roles of the association relation, if any, describes the criteria of classification, i.e. a *province* "is composed of" *cities* and, on the other hand, a *city* "belongs to" a specific *province*. It may be noticed that in the above-defined DAG, alternative path and multiple classification hierarchies are easily considered.

In UML, when a role has its multiplicity's upper bound greater than *1* (e.g. ***), we can define the constraint {*dag*} to express that the target objects form a Directed Acyclic Graph. In our approach, we make use of that constraint and place it on each dimension class to express that the above-defined DAG must be true along each dimension class (as it can be seen in Figure 3).

The concept of **strictness** (section 2) is considered by means of the multiplicity *1* defined in the role from c_i to c_j (see above DAG definition) and the non-strictness by *1..**. The **completeness** of a classification hierarchy is considered by the definition of the constraint {*completeness*} in the role from c_j to c_i. In our approach, all classification hierarchies are qualified as non-completeness by default. In figure 3, we construct classification hierarchies along the store and customer dimension. It should be noticed that the constraint {*dag*} has been defined on dimension classes. Moreover, we add the constraint {*completeness*} to the role from *State* to *Province* to indicate that a *state* consists of only those *provinces* (and no more). It can also be seen that if an association is defined {*completeness*} and the target class (c_i) is not the dimension class, the multiplicity of the role of the association between c_i and c_{i-1} should be *** for coherent reasons, as in the example where there may be provinces that are not related to any city.

Furthermore, an alternative path hierarchy has been considered along the store dimension; and it is easy to see that our approach also considers multiple hierarchies. The path from *store* to *sales_area* is non-strict as a *store* may be related to more than one *sales_area* as well as a *sales_area* may make reference to more than one *state*. Moreover, if the multiplicity from *sales_area* to *state* was *** instead of *1..**, this would also indicate that we would allow *sales_areas* to be related to no *states* (known as non-partitioning relationships in [12]). Finally, it may also be observed that two dimensions share the classification hierarchy; i.e. *store* and *customer*. Thanks to the flexibility provided by the association of classes (and their multiplicities) in UML, all peculiarities of classification hierarchies can be easily considered through the multiplicites of the roles of the dimensions.

3.3 Categorization of Dimensions

As previously stated in section 2, there are some cases, however, in which a MD conceptual model should consider the 'is a' relationship to model categorization of dimensions; mainly when the properties (attributes) specific to some subtypes increases considerably. In UML, the categorization of entities is considered by the concept of generalization-specialization by means of subtype relationships. It should be born in mind that everything that is true for a super-type (associations, attributes, operations) is also true for subtypes, and therefore, any kind of classification hierarchy valid for a supertype, it is also valid for a subtype. Thus, we use this concept to model additional features of the subtypes of an entity.

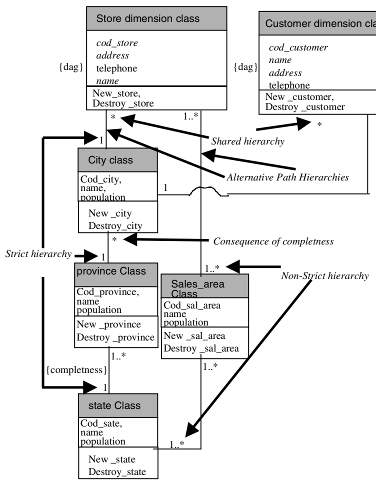


Fig. 3. *Classification hierarchies along store and customer dimensions*

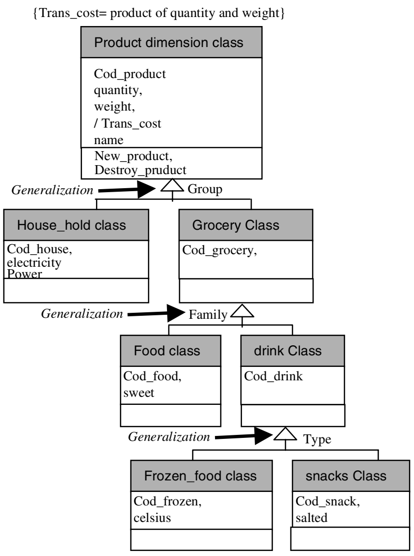


Fig. 4. *Subtypes of products*

In figure 4, it can be seen how the *product* dimension class has been modeled depending on the different subtypes (*groups*, *families* and *types*) considered in the system.

3.4 Aggregated Facts

As our approach can be considered as a Star/Snowflake schema [9], our basic approach can rapidly be extended to consider fact attributes at a particular level of aggregation by defining a new fact as a new shared aggregation. Based on Figure 2, we could rapidly consider a new fact (e.g. *sales_state*) to consider the *quantity_sold* of products aggregated by *states*. In this particular case, the shared aggregation relationship with respect to the *store* dimension should directly be aggregated to the *state* class instead of to the *store* dimension class.

4 On Querying the Conceptual Model

One of the best achievements of our approach is that we can express the user's requirements in the MD conceptual model in greater detail than other proposals presented so far. In OLAP systems, requirements start with a basic requirement, from which users start a navigational interaction process by applying OLAP operations successively to cubes. We call a **session** the interactive process needed to answer an OLAP requirement and it contains the basic cube that needs to be specified as well as the successive cubes obtained by the application of OLAP operations. In our model, we call cube classes (CC) those classes that are needed to analyze data, and are considered as object collections.

In UML, object collections are considered as a special case of association that provide the operator between classes that allows us to manipulate these situations in which the key property of a class is to be related to many objects of other/s, although this associated class will have its own emerging properties. We extend the definition of its graphical notation to adequate it to the needs of MD and OLAP contexts due to the different nature of the data to be analyzed.

The definition of CC is always based on a *fact class*, and therefore, will contain data from dimension classes. This means that we need n-dimension classes and at least one *fact class* to build this basic CC. With reference to the operations permitted on cube classes, however, our approach is twofold. On the one hand, we have the operations that can create (*new*) or destroy (*destroy*) an object of the class, and on the other, those that permit different analyses of the data contained in CC (OLAP operations).

We, therefore, define a graphical structure to consider cube classes, as shown in Figure 5. It can be observed that we can differentiate between the following sections:

– **Head:** this contains the name of the cube class, i.e. the query to be answered
– **Fact area:** contains the fact attributes to be analyzed
– **Slice area:** which contains the dimension conditions that objects must fulfill to be automatically included in the cube class
– **Dice area:** it contains the dimensions required to analyze data. In this area we can also specify the grouping conditions, if any, that provides details about the concrete element within a dimension as follows: <dimension_name.{(class_name.attribute_name)|specialization_criteria)}> means that the objects will be grouped by that level of the hierarchy (class_name), and particularly by a specific attribute within that class_name. On the other hand, if a dimension contains a hierarchy structure defined along it, the specialization criteria may be included into group objects (see section 3.3). The non-inclusion of a class name means that the dimension is not grouped at all.

Finally, let us suppose the following requirement to have a complete view of cube classes, (Figure 6): We wish to analyze the *quantity sold* **where** the *group_of_products* is "Grocery" and the *store_state* is "Valencia" **grouped** according to the *product family* and *brand* **and** the *store province* and *city*. In the Slice area it can be seen how we specify that we do not wish any condition on certain dimensions.

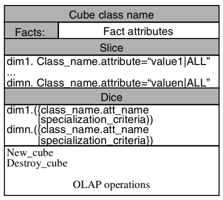

Cube class name
Facts:
Fact attributes
Slice
dim1. Class_name.attribute="value1|ALL"
...
dimn. Class_name.attribute="valuen|ALL"
Dice
dim1.({class_name.att_name
|specialization_criteria))
dimn.({class_name.att_name
|specialization_criteria})
New_cube
Destroy_cube
OLAP operations

Fig. 5. *Definition of cube classes*

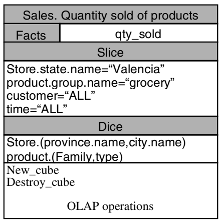

Sales. Quantity sold of products
Facts
qty_sold
Slice
Store.state.name="Valencia"
product.group.name="grocery"
customer="ALL"
time="ALL"
Dice
Store.(province.name,city.name)
product.(Family,type)
New_cube
Destroy_cube
OLAP operations

Fig. 6. *An example of cube class*

5 Comparison with Other Related Work

Several approaches have been published with respect to MD data models (a review of all of them can be found in [15]). There have been few works, however, focused on the aim of this paper, i.e. MD conceptual modeling. The methodology presented in [7] and its underlying Dimensional Fact (DF) model [6] are based on the construction of the multidimensional (MD) schema from the entity-relationship (ER) schemes of the operational systems and they provide mechanisms to represent query patterns graphically. Our approach has a greater power of expression as we can represent the exact cardinality between all elements, such as the many-to-many relationships between facts and some dimensions, and the strictness and completeness along classification hierarchies; issues that, throughout the paper, have been proved to be essential in MD conceptual modeling. Finally, our approach is more user-centered and the graphical representation we propose for query patterns provide further details about the elements to be analyzed, i.e. Fact, Slice and Dice areas).

The proposal presented in [18] extends the ER model to consider multidimensional properties. This approach has some similarities with ours in representing the exact cardinality between facts and dimensions as well as the strictness and completeness along classification hierarchies. Nevertheless, as it is based on the ER model, neither functionality nor behavior can be considered. Therefore, user's requirements cannot be considered. Our approach defines cube classes in the conceptual model to capture the user's requirements. This feature allows us to enrich the power of expression of the model and make the design process much more interactive with the final user.

Another relevant work in this context is the Multidimensional ER (M/ER) model presented in [16], in which graphical elements are defined to consider facts and roll-up relationships. Nevertheless, relevant semantics such as the exact cardinality between facts and dimensions and the strictness and completeness along classification hierarchies are not considered.

The work presented in [8] defines easy and powerful algorithms to translate the Star/Snowflake schema into an OO schema based on the further knowledge provided by the user's queries to facilitate the maintenance of views. Although, to the best of our knowledge, this is the best approach in considering OO on MD modeling, the paper refers to pointers as basic connections between entities (classes), and therefore, it clearly centered in the implementation issues.

In [14] there is a proposal of complex OLAP reports supported by an OO model described in UML. This approach is the best one in considering complex OLAP reports. Nevertheless, the description of relationships between entities, and therefore, basic semantics of the MD data models (e.g. the additivity of fact attributes, cardinality between facts and dimensions,...) are not considered.

The proposal of Lehner in [10] considers Multidimensional objects in the proposed Nested Multidimensional Model and defines a group of OLAP operations to be applied to these objects to permit a subsequent data analysis. However, it does not consider the whole database schema, and basic MD data semantics (as the additivity of measures along dimensions) cannot be considered.

6 Conclusions

In this paper, we have presented an OO conceptual modeling approach based in a subset of UML. This approach takes advantage of the well-known OO conceptual modeling techniques to allow us to support the conceptual design of MDB and OLAP applications. The basic components of our approach are classes that are related through association and shared aggregation relationships. In this context, thanks to the flexibility of the associations in UML and their cardinalities, all the semantics required for a proper MD conceptual modeling are considered, such as the exact cardinality between facts and dimensions, and the strictness and completeness along classification hierarchies. Furthermore, UML provides object collections to manipulate situations in which the key property of a class is to be related to many objects of other/s. Our approach defines an adequate graphical structure for these object collections to consider the user's requirements in great detail.

We are currently working on the modeling of the navigation between these object collections by means of the application of OLAP operations. To start with, we need a correct definition of the OLAP operations to be applied that depends on the relationships between the data considered in requirements ([17]). Finally, let us add that a group of Master students is currently developing a tool in C++ Builder to accomplish the MD conceptual modeling, as described in the paper.

References

1. *Proceedings of the ACM 2nd International Workshop on Data warehousing and OLAP*, Kansas City, Missouri, USA, November 1999. 94

2. M. Blasckha, C. Sapia, G. Höfling, and B. Dinter. Finding Your Way Through Multidimensional Models. In *In Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, pages 198–203, Vienna, Austria, August 1998. 84
3. Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998. 84
4. E.F. Codd, S.B. Codd, and C.T. Salley. Providing OLAP (On-Line Analytical Processing) to User Analyst: An IT Mandate. Available from Arbor Software's web site. http://www.arborsoft.com/OLAP.html, 1996. 83
5. M. Fowler and K. Scott. *UML DISTILLED. Applying the standard object modeling language*. Addison-Wesley, USA, 1998. 84
6. M. Golfarelli, D. Maio, and S. Rizzi. Conceptual Design of Data Warehouses from E/R Schemes. In *Proceedings of the 31st Hawaii Conference on System Sciences*, pages 334–343, Kona, Hawaii, 1998. 83, 84, 92
7. M. Golfarelli and S. Rizzi. A methodological Framework for Data Warehouse Design. In *Proceedings of the ACM 1st International Workshop on Data warehousing and OLAP*, pages 3–9, Washington D.C., USA, November 1998. 83, 92
8. V. Golpalkrishnan, Q. Li, and K. Karlapalem. Star/Snow-Flake Schema Driven Object-Relational Data Warehouse Design and Query Processing Strategies. In Mohania and Tjoa [11], pages 11–22. 84, 92
9. R. Kimball. *The data warehousing toolkit*. John Wiley, 1996. 84, 86, 90
10. W. Lenher. Modelling Large Scale OLAP Scenarios. In *Advances in Database Technology - EDBT'98*, volume 1377, pages 153–167. Lecture Notes in Computer Science, 1998. 93
11. Mukesh Mohania and A. Min Tjoa, editors. *In Proceedings of the First International Conference On Data Warehousing and Knowledge Discovery*, volume 1676 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999. 94
12. TB. Pedersern and CS. Jensen. Multidimensional Data Modeling of Complex Data. In *Proceedings of the 15th IEEE International Conference on Data Engineering*, Sydney, Australia, 1999. 83, 85, 89
13. E. Pourabbas and M. Rafanelli. Characterization of Hierarchies and Some Operators in OLAP Environment. In *Proceedings of the ACM 2nd International Workshop on Data warehousing and OLAP* [1], pages 54–59. 85
14. T. Ruf, J. Goerlich, and T. Reinfels. Dealing with Complex Reports in OLAP Applications. In Mohania and Tjoa [11], pages 41–54. 84, 93
15. C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. An Overview of Multidimensional Data Models for OLAP. Technical Report. http://www.forwiss.tu-muenchen.de/system42, 1998. 92
16. C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. Extending the E/R Model for the Multidimensional Paradigm. In Yahiko Kambayashi, Dik Lun Lee, Ee-Peng Lim, Mukesh K. Mohania, and Yoshifumi Masunaga, editors, *In Proceedings of the First International Workshop on Data Warehouse and Data Mining*, volume 1552 of *Lecture Notes in Computer Science*, pages 105–116. Springer-Verlag, 1998. 83, 92
17. J. Trujillo, M. Palomar, and J. Gómez. Detecting patterns and OLAP operations in the GOLD model. In *Proceedings of the ACM 2nd International Workshop on Data warehousing and OLAP* [1], pages 48–53. 85, 93
18. N. Tryfona, F. Busborg, and J.G. Christiansen. starER: A Conceptual Model for Data Warehouse Design. In *Proceedings of the ACM 2nd International Workshop on Data warehousing and OLAP* [1], pages 3–8. 84, 85, 92

An Optimal Locking Scheme in Object-Oriented Database Systems

Woochun Jun[1] and Le Gruenwald[2]

[1] Dept. of Computer Education
Seoul National University of Education, Seoul, Korea
wocjun@ns.seoul-e.ac.kr
[2] School of Computer Science, University of Oklahoma
Norman, OK 73069, USA
gruenwal@cs.ou.edu

Abstract. In this paper, a locking-based concurrency control scheme is presented for object-oriented databases (OODBs). It is designed for controlling accesses to class hierarchies, which is an important concept in OODBs. Based on access frequency of each class, the proposed scheme incurs less locking overhead than the existing works, explicit locking and implicit locking, for any OODB environments. This paper also proves that the proposed scheme performs better than the existing schemes.

1 Introduction

OODBs have been popular for many non-traditional database applications such as e-commerce with multimedia repositories, computer-aided design, document management, etc. In a typical OODB, a class object consists of a group of instance objects and class definition objects. The class definition consists of a set of attributes and methods that access attributes of an instance or a set of instances. In OODBs, users can access objects by invoking transactions consisting of a set of method invocations on objects [2].

A concurrency control scheme is used to coordinate multiple accesses to the multi-user database so that it maintains the consistency of the database. A concurrency control scheme allows multi-access to a database but incurs an overhead whenever it is invoked. This overhead may degrade the performance of OODBs where many transactions are long-lived. Thus, reducing the concurrency control overhead is critical to improve the overall performance.

Inheritance is a very important concept in OODBs. That is, a subclass inherits definitions defined on its superclasses. Also, there is an *is-a* relationship between a subclass and its superclasses so that an instance of a superclass is a generalization of its subclasses [5]. This inheritance relationship between classes forms a class hierarchy. There are two types of accesses to a class hierarchy, MCA (Multiple Class Access) and SCA (Single Class Access), respectively [6]. MCA is an operation accessing possibly more than one class in the class hierarchy. Examples of MCAs include class definition modification operations and instance accesses to all or some instances of a given class and its subclasses. On the other hand, SCA is an operation accessing one class in the hierarchy. Examples of SCAs are class definition read operations and instance access to a single class. Due to inheritance, for a lock-based concurrency control scheme, when an MCA operation is requested on some class C, it may be necessary to get locks for C as well as all subclasses of C.

In the literature, there are two approaches dealing with class hierarchy, explicit locking and implicit locking, which will be discussed in Section 2. These approaches may work well only for particular applications in OODBs. That is, explicit locking incurs less locking overhead for transactions invoking mostly SCA operations. On the other hand, implicit locking incurs less locking overhead for transactions invoking mostly MCA operations. Recently a lock-based concurrency control scheme for class hierarchy in OODBs is presented [6]. The scheme is based on a so called *special class* (SC) and can be used for any applications with less locking overhead than both explicit locking and implicit locking. In [6], with an assumption that the number of access is stable for each class, it is shown that the proposed scheme (called SC-based scheme) performs better than both explicit locking and implicit locking. Based on that work, in this paper, a new concurrency control scheme is proposed and proven to incur less locking overhead than explicit locking, implicit locking and the SC-based scheme.

This paper is organized as follows. In Section 2 we review previous works dealing with class hierarchy. In Section 3 a new concurrency control scheme is proposed. In Section 4, it is shown that the proposed scheme performs better than existing works. The paper concludes with future research issues in Section 5.

2 Related Works

2.1 Explicit Locking and Implicit Locking

In the literature, there are two major locking-based approaches dealing with a class hierarchy: explicit locking [2,9] and implicit locking [5,7,8]. In explicit locking, for an MCA operation on a class, C, a lock is set not only on the class C, but also on each subclass of C in the class hierarchy. For an SCA operation, a lock is set for only the class to be accessed (called target class). Thus, for an MCA, transactions accessing a class near the leaf in a class hierarchy will require fewer locks than transactions accessing a class near the root in the class hierarchy. Also, another advantage is that it can treat single inheritance where a class can inherit the class definition from one superclass, and multiple inheritance where a class can inherit the class definition from more than one superclass, in the same way. But, explicit locking incurs more locking overhead for transactions accessing a class near the root in a class hierarchy.

On the other hand, the implicit locking is based on intention locks [3]. The purpose of an intention lock on a class indicates that some lock is set on a subclass of the class. Thus, when a lock is set on a class C, it is required to set extra locking on a path from C to its root as well as on C. In implicit locking, when an MCA operation is accessed on a class, C, locks are not required for every subclass of the class C. It is sufficient to set a lock only on the class C (in single inheritance) or locks on C and its subclasses having more than one superclass (in multiple inheritance) [5]. Thus, for an

MCA access, it incurs less locking overhead than explicit locking. But, implicit locking requires more locking overhead when a target class is near the leaf in a class hierarchy due to intention lock overhead.

2.2 The SC-based Scheme

In [6], the SC-based scheme is proposed to incur less locking overhead than the existing schemes, explicit locking and implicit locking. The scheme is based on SC where an SC is a class on which MCA operations are performed frequently. How to determine if a class is an SC or not will be discussed later.

The basic idea is summarized as follows. In SC-based scheme, intention locks are set on only SCs. Thus, locking overhead is less than implicit locking which requires intention locks on every superclass of the target class. Also, in order to have less locking overhead than explicit locking, the following principle is adopted: for an SCA access, a lock is set on only the target class like explicit locking. For an MCA access, unlike explicit locking, locks are set on every class from the target class to the first SC through the subclass chain of the target class. If there is no such SC, then locks are set on leaf classes. If the target class is an SC itself, then lock is set only on the target class.

The scheme is presented as follows. Assume that a lock is requested on class C. For simplicity, strict two-phase locking [1,4] is adopted.

Step 1) Locking on SCs
• For each SC (if any) through the superclass chain of C, check conflicts and set an intention lock.

Step 2) Locking on a target class
•If the lock request is an SCA, check conflicts with locks set by other transactions and set a lock on only the target class C and set a lock on an instance if a method is invoked on the instance
• If the lock request is an MCA, then, from class C to the first *SC* (or leaf class if there is no SC) through the subclass chain of C, check conflicts and set lock on each class. If the class C is an SC, then set a lock only on C.
• If class C has more than one subclass, perform the same step 2) for each subclass chain of C.

For the SC-based scheme, the following SC assignment scheme is adopted in [6]. Assuming that the number of accesses to each class is stable and the access frequency (of MCA and SCA) to each class is known in advance, the SC assignment scheme is constructed as follows.

```
//Start from each leaf class until all classes are checked //
step 1) If a class is a leaf, then the class is assigned as non-SC.
If a class C has not been assigned yet and all subclasses of C have been already assigned, then do the following:
    for class C and all of its subclasses,
        calculate the number of locks ($N_1$) when the class is assigned as SC
        calculate the number of locks ($N_2$) when the class is assigned as non-SC

step 2) Assign it as SC only if $N_1 < N_2$
```

For example, consider a simple single-inheritance class hierarchy as in Fig 1.a and assume access frequency information on each class as in Fig. 1.b. Note that, for MCA operations, the numbers represent only accesses initiated at a given class. Thus, the number of MCA accesses initiated at its superclasses is not counted. The SC assignment to each class is as follows. First C_1 is assigned as non-SC since C_1 is a leaf class. At the class C_2, if C_2 is assigned as non-SC, the numbers of locks needed for class C_1 and C_2 are 200 (for C_1) and 700 (for C_2), respectively, resulting 900 locks. On the other hand, if C_2 is assigned as SC, then locks needed for classes C_1 and C_2 are 800 locks, where 400 locks are for C_1 (200 locks for MCA and 200 locks for SCA) and 400 locks are for C_2 (100 locks for MCA and 300 locks for SCA). Thus, C_2 becomes SC. Similarly, two other classes C_3 and C_4 become non-SCs. Fig. 1.c shows the result of the SC assignment scheme.

$$\begin{tikzcd}
C_4 \arrow[d] \\
C_3 \arrow[d] \\
C_2 \arrow[d] \\
C_1
\end{tikzcd}$$

Fig. 1.a. A class hierarchy

C_4 : SCA:200, MCA: 100

C_3 : SCA: 150, MCA: 100

C_2 : SCA:100, MCA: 300

C_1: SCA:100; MCA:100

Fig. 1.b. Access frequency for each class

$$\begin{tikzcd}
C_4 \arrow[d] \\
C_3 \arrow[d] \\
C_2\text{:SC} \arrow[d] \\
C_1
\end{tikzcd}$$

Fig.1.c. Results of SC assignment

Based on the above assignment scheme, consider the following lock requests by two transactions T_1 and T_2 on a class hierarchy in Fig. 2.a

1) T_1: class definition modification operation on class C6
2) T_2: class definition read on class C4

Let L_i be a lock L set by transaction T_i. Assume that classes C1, C4, C7 and C10 are SCs. As in Fig 2.b, 2.c, and 2.d, 6, 7 and 10 locks are required for T_1 and T_2 by the SC-based scheme, explicit locking, and implicit locking, respectively.

$$\begin{tikzcd}
C1 \arrow[d] & C1(SC){:}L_1;L_2 \arrow[d] & C1 \arrow[d] & C1{:}L_1;L_2 \arrow[d] \\
C2 \arrow[d] & C2 \arrow[d] & C2 \arrow[d] & C2{:}L_1;L_2 \arrow[d] \\
C3 \arrow[d] & C3 \arrow[d] & C3 \arrow[d] & C3{:}L_1;L_2 \arrow[d] \\
C4 \arrow[d] & C4(SC){:}L_1;L_2 \arrow[d] & C4{:}L_2 \arrow[d] & C4{:}L_1;L_2 \arrow[d] \\
C5 \arrow[d] & C5 \arrow[d] & C5 \arrow[d] & C5{:}L_1 \arrow[d] \\
C6 \arrow[d] & C6{:}L_1 \arrow[d] & C6{:}L_1 \arrow[d] & C6{:}L_1 \arrow[d] \\
C7 \arrow[d] & C7(SC){:}L_1 \arrow[d] & C7{:}L_1 \arrow[d] & C7 \arrow[d] \\
C8 \arrow[d] & C8 \arrow[d] & C8{:}L_1 \arrow[d] & C8 \arrow[d] \\
C9 \arrow[d] & C9 \arrow[d] & C9{:}L_1 \arrow[d] & C9 \arrow[d] \\
C10 \arrow[d] & C10\ (SC) \arrow[d] & C10{:}L_1 \arrow[d] & C10 \arrow[d] \\
C11 & C11 & C11{:}\ L_1 & C11
\end{tikzcd}$$

Fig 2.a class hierarchy Fig. 2.b. Locks by SC-based scheme Fig. 2.c. Locks by Explicit locking Fig. 2.d. Locks by Implicit locking

3 Proposed Class Hierarchy Locking Scheme

3.1 Background

The proposed scheme is based on the SC-based scheme. The basic idea is that some redundant locks can be reduced without affecting the correctness of the scheme. Assume that a class C is accessed so that it needs to be locked. For the SC-based scheme, an intention lock is set on every SC through the superclass chain of C. On the other hand, the proposed scheme does not have to set intention locks on every SC through the superclass chain. That is, only the first SC near root and the last SC near the class C need to be locked as long as SCs excluding the first SC and the last SC have only one subclass.

For example, consider the class hierarchy in Fig. 2.a. Also, assume the following accesses by transaction T_3. Fig 3.a and Fig. 3.b show locks by the SC-based scheme and the proposed scheme, respectively.

T_1: class definition update operation on class C11.

$$\begin{tikzcd}
C1(SC){:}L_3 \arrow[d] \\
C2 \arrow[d] \\
C3{:} \arrow[d] \\
C4(SC){:}L_3 \arrow[d] \\
C5 \arrow[d] \\
C6{:} \arrow[d] \\
C7(SC){:}L_3 \arrow[d] \\
C8 \arrow[d] \\
C9 \arrow[d] \\
C10\ (SC){:}L_3 \arrow[d] \\
C11{:}\ L_3
\end{tikzcd}$$

Fig. 3.a. Locks by SC-based scheme

$$\begin{tikzcd}
C1(SC){:}L_3 \arrow[d] \\
C2 \arrow[d] \\
C3{:} \arrow[d] \\
C4(SC) \arrow[d] \\
C5 \arrow[d] \\
C6 \arrow[d] \\
C7(SC) \arrow[d] \\
C8 \arrow[d] \\
C9 \arrow[d] \\
C10(SC){:}L_3 \arrow[d] \\
C11{:}\ L_3
\end{tikzcd}$$

Fig. 3.b. Locks by the proposed scheme

3.2 A New Class Hierarchy Locking Scheme

Based on the ideas explained as in Section 3.1, the proposed scheme is as follows. Assume that a lock is requested on class C. Also, it is assumed that the strict two-phase locking scheme is adopted.

Step 1) Locking on SCs

• (case I) at least one of SCs excluding the first SC and last SC through the superclass chain of C has more than one subclass.

For each *SC* (if any) through the superclass chain of C, check conflicts and set an intention lock.

• (case II) Otherwise

For the first SC and the last SC through the superclass chain of C, check conflicts and set an intention lock.

Step 2) Locking on a target class

•If the lock request is an SCA, check conflicts with locks set by other transactions and set a lock on only the target class C and set an a lock on the instance to be accessed if a method is invoked on the instance.

• If the lock request is an MCA, then, from class C to the first *SC* (or leaf class if there is no SC) through the subclass chain of C, check conflicts and set a lock on each class. If the class C is an SC, then set a lock only on C.

The reason to set a lock on each class (besides the first SC) from class C to the first SC (not including the SC) is as follows: if a lock is set only on the first SC, then some conflict may not be detected. For example, if a requester accesses a subclass of a lock holder's class locked by MCA, then such a conflict may not be detected. For example, in Fig. 3.b., assume that class C8 is MCA locked by a transaction T_1 and C9 is to be SCA locked by another transaction T_2. In this case, if C9 is not MCA locked by T_1, T2 can get a lock successfully in the presence of conflict.

•If class C has more than one subclass, perform the same step 2) for each subclass chain of C.

For example, consider the class hierarchy as in Fig. 1.a. Also, assume that locks are requested by T_1 and T_2 as follows.

1) T_1: class definition update operation on class C6

2) T_2: class definition update operation on class C7

As in Fig 4.a, 4.b, 4.c and 4.d, 6, 7 11 and 13 locks are required for T_1 and T_2 by the proposed scheme, SC-based scheme, explicit locking, and implicit locking, respectively.

$$\begin{tikzcd}
\text{C1(SC)}:L_1;L_2 \arrow[d] & \text{C1(SC)}:L_1;L_2 \arrow[d] & \text{C1} \arrow[d] & \text{C1}:L_1;L_2 \arrow[d] \\
\text{C2} \arrow[d] & \text{C2} \arrow[d] & \text{C2} \arrow[d] & \text{C2}:L_1;L_2 \arrow[d] \\
\text{C3} \arrow[d] & \text{C3} \arrow[d] & \text{C3} \arrow[d] & \text{C3}:L_1;L_2 \arrow[d] \\
\text{C4(SC)}:L_1 \arrow[d] & \text{C4(SC)}:L_1;L_2 \arrow[d] & \text{C4} \arrow[d] & \text{C4}:L_1;L_2 \arrow[d] \\
\text{C5} \arrow[d] & \text{C5} \arrow[d] & \text{C5} \arrow[d] & \text{C5}:L_1;L_2 \arrow[d] \\
\text{C6}:L_1 \arrow[d] & \text{C6}:L_1 \arrow[d] & \text{C6}:L_1 \arrow[d] & \text{C6}:L_1;L_2 \arrow[d] \\
\text{C7(SC)}:L_1;L_2 \arrow[d] & \text{C7(SC)}:L_1;L_2 \arrow[d] & \text{C7}:L_1;L_2 \arrow[d] & \text{C7}:L_2 \arrow[d] \\
\text{C8} \arrow[d] & \text{C8} \arrow[d] & \text{C8}:L_1;L_2 \arrow[d] & \text{C8} \arrow[d] \\
\text{C9} \arrow[d] & \text{C9} \arrow[d] & \text{C9}:L_1;L_2 \arrow[d] & \text{C9} \arrow[d] \\
\text{C10(SC):} \arrow[d] & \text{C10 (SC)} \arrow[d] & \text{C10}:L_1;L_2 \arrow[d] & \text{C10} \arrow[d] \\
\text{C11} & \text{C11} & \text{C11}: L_1;L_2 & \text{C11}
\end{tikzcd}$$

Fig 4.a. Locks by Proposed scheme

Fig. 4.b. Locks by SC-based scheme

Fig. 4.c. Locks by Explicit locking

Fig. 4.d. Locks by Implicit locking

From the proposed scheme, it is clear that it incurs fewer or equal number of locks than the SC-based scheme for any kinds of accesses to OODBs. Especially, if an OODB does not have any SC, then the proposed scheme incurs equal number of locks as the SC-based scheme. The more SCs an OODB have, the bigger difference in performance between the proposed scheme and the SC-based scheme.

4 The Proof of Correctness of the Proposed Scheme

In this Section, we will show that the proposed scheme performs better than the existing works, explicit locking, implicit locking and the SC-based scheme. It is shown that the SC-based scheme performs better than both explicit locking and implicit locking in [6]. Thus, it is sufficient to show that the proposed scheme performs better than only the SC-based scheme. Based on the discussion from Section 3.2, the proposed scheme incurs fewer or equal number of locks than the SC-based scheme for any kinds of accesses to OODBs. Thus, in this Section, it is sufficient to prove that the proposed scheme is correct, that is, it satisfies serializability [1]. We prove this by showing that, for any lock requester, its conflict with a lock holder (if any) is always detected. With this proof, since our class hierarchy locking scheme is based on two-phase locking, it is guaranteed that the proposed scheme satisfies serializability.

Depending on the lock requester's type, lock holders can be divided as follows. If a lock requester is an SCA, then its lock holders (whose lock modes need to be checked for conflict with lock requester) consist of transactions holding locks on the target class and all SCs in the superclass chain of the target class. If a lock requester is an MCA, then its lock holders include those defined above plus transactions holding locks on each class from the target class to the first SC in the subclass chain of the target class.

There are four cases depending on the types of lock requesters and holders.

Case 1) The lock holder is an SCA
The lock requester is an SCA

If a lock holder (H) and a lock requester (R) access different classes, there is no conflict. If a lock holder and a lock requester access the same class, the possible conflicts can be detected on the target class. This is due to the reason that there is no conflict on all SCs through the superclass chain of the target class since intention locks on SCs are compatible with R.

Case 2) The lock holder is an SCA
The lock requester is an MCA

Let C_R and C_H be two target classes on which R requests a lock and H holds a lock, respectively. If C_H is a superclass of C_R, there is no conflict since R does not access C_H. If C_H is C_R itself or its subclass, then there are two subcases. If there exists an SC, which is a superclass of both C_R and C_H, then conflicts are detected on the SC. (case 2.1). That is, in Fig. 5.a, the possible conflicts are detected on SC1 since both R and H must have locks on SC1. Otherwise, the conflicts are detected as follows. As in Fig 5.b, in case 2.2, if there is an SC between C_R and C_H, the conflict is detected on SC1 since C_R and C_H must have locks on SC1 based on the proposed scheme. On the other hand, if there is no such SC between C_R and C_H as in Fig. 5.c (case 2.3), the conflict is detected on C_H since R must have a lock on C_H.

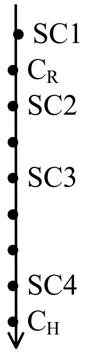


Fig 5. a. case 2.1

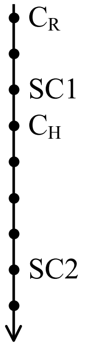


Fig. 5.b. case 2.2

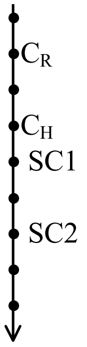


Fig. 5.c. case 2.3

Case 3) The lock holder is an MCA
The lock requester is an SCA

If the C_H is a subclass of the C_R, there is no conflict. If C_H is C_R itself or superclass of C_R, then there are two cases in which conflicts will be detected. If there exists an SC, which is a superclass of both C_R and C_H as in Fig. 6.a, then conflicts are detected on the SC. (case 3.1). This is due to that H and R must a lock on the SC according to our scheme. Otherwise, there are two subcases. At first, if there exists a SC between C_H and C_R, the possible conflicts are detected on the first SC through the subclass chain of C_H. For example, in Fig. 6.b. the conflict can be detected on SC1 (case 3.2). If there is no SC between C_H and C_R as in Fig. 6.c, the conflict is detected on C_R since C_H must set a lock on the class C_R(case3.3).

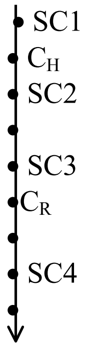


Fig 6. a. case 3.1

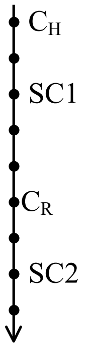


Fig. 6.b. case 3.2

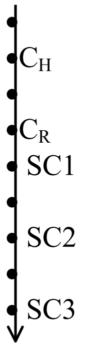


Fig. 6.c. case 3.3

Case 4) The lock holder is an MCA
The lock requester is an MCA

If C_H is C_R itself or a superclass of C_R, conflicts are detected as in case 3. Otherwise, conflicts are detected as in case 2.

From cases 1), 2), 3) and 4), we can conclude that, for any lock requester, it is guaranteed that its conflicts with a lock holder (if any) are always detected. Also, since the proposed scheme is based on two-phase locking, serializability is guaranteed [1]. In turn, this means that the proposed scheme performs better than the existing schemes, explicit scheme, implicit scheme and the SC-based scheme.

5 Conclusions and Further Work

In this paper, a lock-based concurrency control scheme is presented for object-oriented databases (OODBs). It is designed for controlling accesses to class hierarchies, which is an important concept in OODBs. Based on access frequency of each class, the scheme incurs less locking overhead than existing works, explicit scheme and implicit scheme and the SC-based scheme, for any OODB environments. This paper also proves that the proposed scheme performs better than the existing schemes.

Currently we are developing a concurrency control scheme for controlling access to composite object hierarchies, which is also a major aspect in OODBs. Our goal is to combine our class hierarchy scheme with the composite object scheme. Also, we will conduct the performance evaluation study in order to compare our work with the existing schemes using either simulation or theoretical analysis.

References

1. Bernstein,P., Hadzilacos, V. and Goodman, N.: Concurrency Control and Recoveryin Database Systems, Addison-Wesley (1987)
2. Cart, M. and Ferrie, J.: Integrating Concurrency Control into an Object-Oriented Database System, 2nd Int. Conf. on Extending Data Base Technology, Venice, Italy, Mar. (1990) 363 - 377
3. Date, C.: An Introduction to Database Systems, Vol. II, Addison-Wesley (1985)
4. Eswaran, K., Gray, J., Lorie, R. and Traiger, I.: The notion of consistency and predicate locks in a database system, Communication of ACM, Vol. 19, No. 11, Nov. (1976), 624 - 633
5. Garza. J, and Kim, W.: Transaction Management in an Object-Oriented Database System, ACM SIGMOD Int. Conf. on Management of Data, Chicago, Illinois, Jun. (1988) 37 - 45
6. Jun, W. and Gruenwald. L.: An Effective Class Hierarchy Concurrency Control Technique in Object-Oriented Database Systems, Journal of Information And Software Technology, Vol. 40. No. 1, Apr. (1998) 45-53

7. Lee, L. and Liou, R.: A Multi-Granularity Locking Model for Concurrency Control in Object-Oriented Database Systems, IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 1, Feb. (1996) 144 - 156
8. Malta, C. and Martinez, J.: Controlling Concurrent Accesses in an Object-Oriented Environment, 2nd Int. Symp. on Database Systems for Advanced Applications, Tokyo, Japan, Apr. (1992) 192 - 200
9. Malta, C. and Martinez, J.: Automating Fine Concurrency Control in Object-Oriented Databases, 9th IEEE Conf. on Data Engineering, Vienna, Austria, Apr. (1993) 253- 260

An Efficient Distributed Algorithm for Computing Association Rules

Yijun Li[1], Xuemin Lin[1], and Chi-Ping Tsang[2]

[1] School of Computer Science and Engineering
University of New South Wales, Sydney, NSW 2052, Australia
{yijunli,lxue}@cse.unsw.edu.au
[2] Department of Computer Science
University of Western Australia, Nedlands, WA 6907, Australia
tsang@cs.uwa.edu.au

Abstract. Data mining aims to efficiently discover previously unknown knowledge from large databases. It is highly demanding in numerous real-life applications, such as marketing strategy, financial forecast, etc. One of the fundamental problems in the area is the efficient computation of association rules. In this paper, we shall investigate this problem in a distributed database. Particularly, we will present an efficient distributed algorithm for mining distributed association rules. Our experiment results suggest that the proposed algorithm outperforms the existing distributed algorithms. Further, we also study a distributed version of the problem of association rules; and extend our algorithm to solve this new problem.

Keywords: Relational Databases, Distributed Data Mining, and Algorithms.

1 Introduction

The discovery of "association rules" in databases may provide useful background knowledge to decision support systems, selective marketing, financial forecast, medical diagnosis, and many other applications. The problem of efficiently computing association rules has received a great deal of attention in the past several years; and a number [2,6,9,11] of efficient algorithms have been proposed to approach this problem. As the size of a database to be mined can be very large, parallel computation techniques have also been explored [1,10].

Consider that in a distributed organization, the database may be allocated through a computer network. This leads to a real demand for developing distributed computation techniques in data mining. In this paper, we shall restrict ourself to an investigation of mining association rules in a distributed database.

In [9], an efficient distributed algorithm is proposed. It should be clear that the parallel algorithms developed in [1,10] can be immediately used as distributed algorithms. Comparing the distributed algorithm DMA [4] with an implementation of the parallel algorithm CD [1] in a distributed environment, DMA is more efficient than CD because a reduction of both network communication and local processing costs has been made in DMA. The experiment results in [4] confirmed such an improvement.

Although DMA has a smaller message overhead than that of an implementation of the parallel algorithm PDM [10] in a distributed environment, our initial experiment results showed that in a fast network such as LAN, PDM is faster than DMA in many cases. A careful analysis of DMA and PDM reveals that PDM is based on a faster local processing algorithm (sequential algorithm) DHP [9], while DMA employs a better network communication protocol. This leads to a consideration of combining together the advantages from both DMA and PDM into one algorithm. Further, according to our recent research results, there exists a faster sequential algorithm, PLO-DHP, than DHP. The PLO-DHP enjoys a speed-up of up to 50% over DHP according to the experiment results [6]. In this paper, we will present a new distributed algorithm based on PLO-DHP, and it employs a standard polling technique in distributed computing as suggested in DMA. Our experiment results suggest that our algorithm is much more efficient than both PDM and DMA in a fast network, and has a low message overhead. This is the first contribution of the paper.

The second contribution of the paper is to extend our algorithm to solve a new problem - mining *distributed association rules*; that is, computing association rules that are shared by at least N component databases where N is defined by users. For instance, in a multi-national supermarket company association rules shared by its N local supermarkets will give us the information of common purchase interests of local residents, so that good decisions of goods transportation and organization cross the company can be made. We call the problem "N common association rules problem"; and it will be formally defined in section 2.

In fact, the association rules problem is a special case of the distributed association rules problem (N common association rules problem). Specifically, when $N = 1$, these two problems are equivalent. This yields a simplification of the presentation of our results in the paper. The above two contributions will be combined into a single algorithm - MNA (stands for **M**ining of N common **A**ssociation rules). Considering the limitation of paper length, in this paper we present only the algorithm of mining "N commonly large itemsets", which is the most expensive and important part in mining N common association rules, while the interested readers may refer to [7] for the other details.

The rest of paper is organized as follows. In section 2, we will formally define the problem, and present necessary preliminaries. In section 3, we present the algorithm - MNA. Section 4 gives the experiment results and discussions. This is followed by conclusions.

2 N Common Association Rules

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals, called *items*. Suppose that there are n sites $\{s_i : 1 \leq i \leq n\}$ in the computer network. At each site s_i, D_i is a set of transactions (tuples), where each transaction T is a set of items (that is, $T \subseteq I$), and is identified by its local ID L_TID. The global database D is the union of the component databases D_i where each tuple T is identified by its local ID together by the site ID i (that is, identified by (L_TID, i)). Following the assumptions made in [2,9,11], we assume that the items in each transaction (tuple) are sorted on their keys in an increasing order. A transaction T *contains* a set X of items in I, if $X \subseteq T$.

An *association rule* is an implication of the form $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has support s in the database D if $s\%$ of transactions in D contain $X \cup Y$. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence c if $c\%$ of transactions in D that contain X also contain Y.

With respect to a given minimum support value $s\%$, a given confidence $c\%$, and a given integer $N \leq n$, the problem of mining N *common association rules* is to find all association rules $X \Rightarrow Y$ in D, such that

> $X \Rightarrow Y$ has support s and confidence c in D as well as in at least N component databases in D.

A set X of items is a *k-itemset* if it consists of k items, and is *large* if there are at least $s\%$ of transactions in D that contain X. A k-itemset is *N commonly large* if it is large in D as well as large in at least N component databases of D.

Following the framework [2] for mining association rules, we are going to solve the problem by two phases: 1) compute N commonly large itemsets; and then 2) determine the N common association rules from the large itemsets.

The computation of the second phase is trivial and inexpensive, and thus will not be presented in this paper. The interested readers may refer to [7] for the details. In this paper, we mainly discuss the first phase.

Below, we present the notation used in the rest of the paper. Suppose that a distributed database $D = \cup_i^n D_i$ is given where each D_i is component database that reside at site i; and $s\%$, $c\%$, and N are all given. We use $L_{k,N}$ to denote the set of k-itemsets that are N commonly large in D, and use $C_{k,N}$ to denote a superset of $L_{k,N}$. Note that when $N = 1$, $L_{k,1}$ and $C_{k,1}$ are abbreviated to L_k and C_k. We use L_k^i to denote the set of k-itemsets that are large in the component database D_i.

Like the other developments [1,4,10], our algorithm for computing N commonly large itemsets will follow the framework [2] - Algorithm Apriori, and be conducted in an iterative fashion:

Algorithm Apriori

```
Scan $D$ to obtain $L_1$
for ($k = 2; L_{k-1} \neq \emptyset; k++$) do
  { $C_k \leftarrow gen(L_{k-1})$; //generate candidates by $L_{k-1}$.
    if $C_k \neq \emptyset$ then
      { $count(C_k)$; //count candidates against $D$.
        $L_k \leftarrow \{c \in C_k | c.count \geq s\% \times |D|\}$; }
    else $L_k \leftarrow \emptyset$ }
Return $\cup L_k$.
```

Although the algorithm DHP [9] follows the same iterations as suggested in Apriori, in its first iteration a hash table H_2 is built while L_1 is generated. Then in the second iteration of DHP, we check C_2 against H_2 to reduce the size of C_2. Further, in the end of the second iteration of DHP, the original database is trimmed to a smaller one for later iterations. Experiment results [9,6] showed that DHP is more efficient than Apriori.

In our recent research paper [6], we presented an improvement of DHP that has been proven more efficient than DHP. This algorithm is named as PLO-DHP. Our distributed algorithm MNA is based on PLO-DHP.

3 Algorithm MNA

In this section, we present an efficiently distributed algorithm, NMA, for mining N common association rules. The algorithm MNA employs the algorithm PLO-DHP as a base mining algorithm, and applies a standard distributed technique - polling site, as suggested in [4].

Suppose that a distributed transaction database $D = \{D_i : 1 \leq i \leq n\}$ is given such that the n component databases D_is reside at n different sites. Further, let the minimum support value $s\%$, confidence $c\%$ and N be given. The following theorem provides a fundamental base to the correctness of the algorithm MNA; and the interested readers may refer to [7] for a detailed proof.

Theorem 1. *For any given N and k, $L_{k,N} \subseteq \cup_{i=1}^{n} L_k^i$.*

The algorithm MNA iteratively executes the following local processing and message communication phases.

Phase 1: Local Processing. In order to avoid multiple scans of local databases at an iteration k, each site i will generate the global $C_{k,N}$ and do counting on $C_{k,N}$ against its local database to produce $LC_k^i = L_k^i \cap C_{k,N}$.

Phase 2: Message Communication. Once LC_k^i is generated at each site i, message communication through the network is required to generate $L_{k,N}$; and then $L_{k,N}$ will be broadcasted to all sites for the next iteration. In the second iteration, network communication is also required for generating H_2.

Let H_2^i denote the hash table used in DHP for mining D_i. The algorithm MNA is presented below.

Algorithm MNA

```
M1: Each site $i$ generates $L_1^i$ and $H_2^i$ by scanning
      $D_i$;
M2: $L_{1,N} \leftarrow COMP(\cup_{i=1}^{n} L_1^i)$;
M3: $H_2 \leftarrow COMP(\cup_{i=1}^{n} H_2^i)$;
M4: while ($|L_{k-1}| \geq k$) do
M5:   { if ($k = 2$) then at each node $i$ do
M6:       { $C_{2,N} \leftarrow MNA\_gen(L_{1,N})$;
M7:        $LC_2^i \leftarrow MNA\_count(C_{2,N})$;
M8:        $D_i' \leftarrow MNA\_trim(D_i)$;
M9:        $L_{2,N} \leftarrow COMP(\cup_i LC_2^i)$; }
M10:    else at each node $i$ do
M11:      { $C_{k,N} \leftarrow MNA\_gen(L_{k-1,N})$;
N12:        $LC_k^i \leftarrow MNA\_count(C_{k,N})$;
M13:        $L_{k,N} \leftarrow COMP(\cup_i LC_k^i)$; }
M14:    $k \leftarrow k + 1$; }
M15: Return $\cup_k L_{k,N}$.
```

In MNA, line M1 is to generate all locally large 1-itemsets and to build a hash table for each local database D_i, and it is the same as the fisrt iteration in DHP [10]. The lines M6, M7, M8, M11 and M12 are in the phase of local processing and will be explained in Section 3.1, while the lines M2, M3, M9 and M13 are in the phase of message communication and the details will be given in Section 3.2.

3.1 Applying Triple Pruning and On-line Indexing

As noted [6,9], the local processing costs in mining large itemsets are determined by sizes of databases and generated candidates. Smaller the sizes, lower the costs. Motivated by this, in the algorithm we employ a triple pruning technique to reduce the size of candidates, as well as applying the on-line indexing technique [6] to logically and vertically truncate the database.

A Triple Pruning Technique In [2], a sort-merge join on L_{k-1} and L_{k-1} is proposed that generates C_k from L_{k-1}, and together with a pruning procedure that removes an element c from C_k if a sub $(k-1)$-itemset of c is not in L_{k-1}.

In [4], it suggested that to take the advantage of distributed *data pattern skew*, we should include only the elements c in C_k such that all sub $(k-1)$-itemsets of c are locally large at one common component database. This is the second pruning technique applied in our algorithm.

Consider that in many real applications, the size of C_2 is much larger than the sizes of other candidate sets. In [9], a hash pruning technique is provided for reducing the size of C_2. To do this, firstly a hash table H_2 is built; and then each element $c \in C_2$ is checked against an entry H_2 to determine if it should be pruned from C_2.

We noted that these three pruning techniques can be modified to be used in mining N commonly large itemsets. Moreover, in our algorithm MNA we propose to combine these three techniques together to form a triple pruning technique. Let c be a k-itemset and $S(c)$ denote the set of common sites at which all sub $(k-1)$-itemsets of c are locally large. Now we present our algorithm to generate $C_{k,N}$ from $L_{k-1,N}$. Suppose that the items in each element of $L_{k,N}$ are stored according to the increasing order of item keys.

$MNA\_gen(L_{k-1,N})$:

```
$C_{k,N} \leftarrow L_{k-1,N} \bowtie L_{k-1,N}$; //two tuples from $L_{k-1,N}$ joins on the first $k-2$ items.
for each element $c \in C_{k,N}$ do
  if $(\exists c' \subset c \wedge |c'| = k-1 \wedge c' \notin L_{k-1,N})$ then $C_k \leftarrow C_k - \{c\}$;
  else { compute $S(c)$; if $(|S(c)| < N)$ then $C_k \leftarrow C_k - \{c\}$; }
Return $C_k$.
```

Note that when $k = 2$ each $c \in C_2$ also needs to be checked against H_2 to determine whether or not it should be included in C_2; this results in a triple pruning technique. More details about hash pruning technique will be presented in Section 3.2 when we introduce message communication phase of the algorithm MNA. Further, the procedure MNA_gen $(L_{k-1,N})$ is implemented in such a way that once a c is generated from $L_{k-1} \bowtie L_{k-1}$, the pruning criteria shall be checked immediately to determine if c should be included in C_k will be given in [7].

The full details of the proof that for $k \geq 1$, $C_{k,N} \supseteq L_{k,N}$ while applying the procedure MNA_gen This guarantees the generation correctness of candidates.

An On-line Bitmap Indexing Technique At each iteration, once the global candidates $C_{k,N}$ have been obtained, each site i will do local counting against its database D_i. Besides building up a hash tree index T_k on $C_{k,N}$ to speed up the counting process, in [6] we proposed an on-line bitmap indexing technique to logically reduce the length of each tuple. The idea is quite simple. For each tuple $T \in D_i$, instead of using T for counting $C_{k,N}$, we need only to use $T' \subseteq T$ where T' consists only items that are contained by elements in $C_{k,N}$. To implement this efficiently, an on line bitmap index $B_k[l]$ (for $0 \leq l \leq |I|$) is created in the main memory, such that $B_k[l] = 1$ if and only if the item i_l is contained in an element in $C_{k,N}$. The paper [6] gives the detailed implementation of the technique; and the experiments report that there will be up to 50% reduction on counting costs by applying this technique comparing with the standard counting process adopted in Apriori and DHP.

Therefore, the algorithm MNA will adopt the on-line bitmap indexing technique combining with the standard counting process in [2,9].

3.2 Applying Polling to Reduce Message Overhead

After counting $C_{k,N}$ at each site i, every element c in $C_{k,N}$ has a count c_i in D_i. To assemble the global count for each element $c \in C_{j,N}$ from its local counts, a network communication must be invoked. To minimize the message overhead, a polling technique was proposed [4]. In our algorithm, we will apply this technique to both procedures $COMP(\cup_i LC_j^i)$ and $COMP(\cup_i H_2^i)$.

Note that in the procedure of $COMP(\cup_i H_2^i)$, a hash bucket in an H_2^i is created corresponding to the value of $H(x, y)$ where H is a hash function and

(x, y) is a 2-itemset. Each hash bucket stores the number of 2-itemsets falling into it. At each site i, let HL_2^i denote the subset of hash buckets in H_2^i whose counts are larger than $s\% \times |D_i|$. Besides, a same hash function h is employed at each site to hash HL_2^i, and h has n hash buckets. The procedure $COMP(\cup_i H_2^i)$ consists of the following steps:

Step 1: At each site i, hash the buckets of HL_2^i using h. If the hash value is l then send the corresponding hash bucket together with its count to site l. Let $HL_2(l)$ denote the hash buckets that site l received from all other sites.

Step 2: At each site l, once it receives all messages from the other sites it will assemble the global count for each element c in $HL_2(l)$ if c is from at least N sites. To assemble the global count for c, site l needs only to contact the sites j where c is not in HL_2^j.

Step 3: Let $HL_2'(l)$ denote the set of buckets in $HL_2(l)$ whose global count is not less than $s\%|D|$. At each site i, broadcast $HL_2'(i)$ to the other sites; and then build up H_2 once it receives all $HL_2'(j)$ from the other sites j; H_2 gives the binary information of whether the global count of a hash bucket obtained using hash function H is less than $s\%|D|$.

$COMP(\cup_i LC_j^i)$ is implemented in a similar way. Next we present an example to explain the algorithm MNA.

3.3 An Illustration of MNA

In this subsection, we first use one example to illustrate the algorithm MNA. Suppose that there are three sites together with three component databases, as depicted in Figure 1. Let $N = 2$ and $s = 40$.

Assume that we define the hash function as

$$H(x, y) = \{(\text{order of } x) \times 10 + \text{ order of } y\} \bmod 7.$$

After the first iteration, $L_{1,2} = \{A, B, C, D, F\}$, and the global hash table is illustrated in Figure 2. Note that:

$$\begin{aligned} L_{1,2} \bowtie L_{1,2} = \{&(A, B), (A, C), (A, D), (A, F), \\ &(B, C), (B, D), (B, F), (C, D), \\ &(C, F), (D, F)\}. \end{aligned}$$

Note that the pruning technique as stated in "if" phase of MNA_gen is not applicable in the second iteration. If we use the pruning technique in the "else" phase of MNA_gen, (A, B) and (A, F) will be removed from $L_{1,2} \bowtie L_{1,2}$. If we use the hash pruning technique as described in MNA, the 2-itemsets (B, D), (C, F), and (D, F) will be eliminated from $L_{1,2} \bowtie L_{1,2}$. Therefore, a combination of these two pruning techniques, as suggested in our algorithm MNA, will remove $\{(A, B), (A, F), (B, D), (C, F), (D, F)\}$ from $L_{1,2} \bowtie L_{1,2}$, that is, there are only five elements left in $C_{2,2}$ using MNA.

Site1

L_TID	Items
1	A, C, E
2	A, D, F
3	B, C, D

Site2

L_TID	Items
1	A, B, C, D, F
2	A, C, D
3	B, F

Site3

L_TID	Items
1	A, B, C, D
2	B, C, F
3	D, F

Fig. 1. An example database

Using the network communication method introduced in section 3.2, we obtained that $L_{2,2} = \{(A, C), (A, D), (B, C), (C, D)\}$. Using MNA, $C_{3,2} = \emptyset$. Thus, the algorithm MNA terminates.

The Theorem 1 and the fact that $C_{k,N} \supseteq L_{k,N}$ guarantee that the algorithm MNA is correct.

4 Experiment Results and Discussions

For performance evaluation purpose, the algorithm DMA, the algorithm PDM, and our algorithm MNA have been implemented on a 10M shared nothing LAN, which connects four Pentium/133. The four connected PCs run Windows NT system; and each of them has 64M main memory. We implement these algorithms using Java/RMI. The database was generated using the benchmark generation software downloaded from the web site of IBM Almaden Research Center, and the software uses the parameters as shown in Figure 3

In our experiments, to generate a distributed database, we first use the benchmark generation software to generate $|D|$ and then randomly partion $|D|$ into four component database D_i for $1 \leq i \leq 4$. The component databases are respectively stored in the local disks.

In the first experiment, we choose $M = 1000$, $|I| = 4$, $|D| = 400,000$, and $|T| = 15$. Besides, we choose $30,000$ as the bucket number to build a hash table H_2, and choose 300 as the number of children in each hash tree index

Bucket_id	Count	2-itemsets contained
0	5	(A, D), (C, E)
1	3	(A, E), (C, F)
2	6	(A, F), (B, C)
3	3	(B, D)
4	3	(D, F)
5	5	(A, B), (B, F)
6	8	(A, C), (C, D)

Fig. 2. H_2

|D|	Number of transactions
|T|	Average size of transactions
|I|	Average size of the maximal potentially large itemsets
|L|	Number of potentially large itemsets
M	Number of items

Fig. 3. Parameters

on candidate sets. In this experiment, the algorithm MNA runs the parameter $N = 1$; this corresponds to the problem of mining association rules.

The experiment results are depicted in Figure 4. From Figures 4 (c) and (d), we can see that the algorithm MNA is faster than both PDM and DMA. There are two interesting phenomena occurred - while the minimum support value goes smaller,

Ph1: a reduction effect on the execution time of BMA by MNA is decreasing, and

Ph 2: a reduction effect on the execution time of PDM by MNA is increasing.

Note that while the minimum support value goes smaller, more large itemsets will be formed, and thus, the effect of local processing cost reduction by PLO-DHP is reduced. This is why Ph1 occurs. Further, while the minimum support value goes smaller, more candidates will be generated, and thus network communication costs are increased. Since DHP applies a broadcast communication protocol, the communication costs increase in DHP is much lager than that in MNA. This is

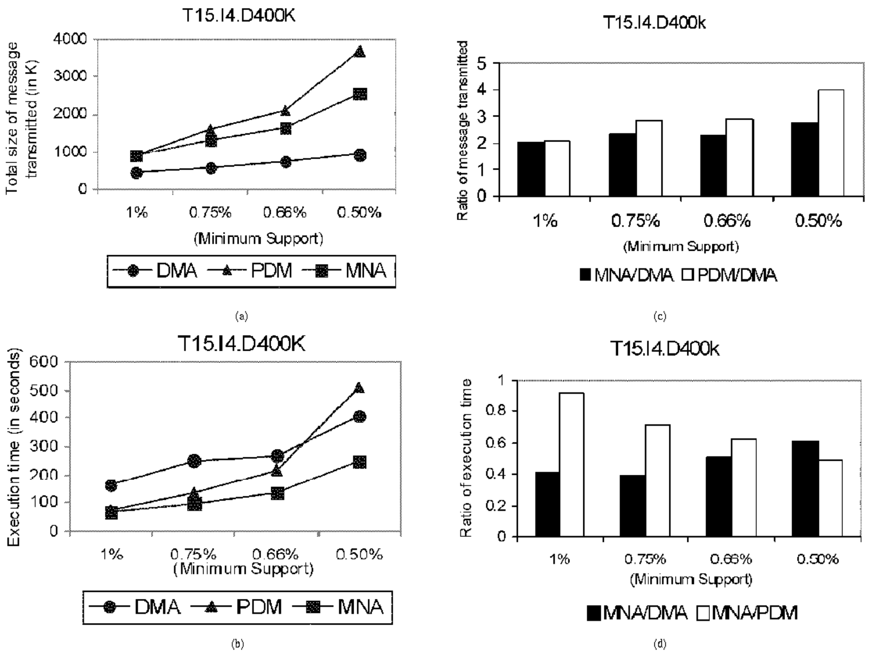


Fig. 4. Results of the experiment 1

why Ph2 occurs. We should also note that the execution time of MNA may be as less as 50% of that of PDM and 40% of that of DMA.

The experiment shows that the total message size of MNA is larger than that of DMA. This is because an extra demand of network communication for building H_2 happens in the iteration two of MNA. However, the total message size of MNA is smaller than that of PDM. This is because that the communication protocol used in MNA is better than PDM in a distributed environment.

In our second experiment, we try to show the speed up caused by an increase of N. The experiment results are shown in Figure 5, where $M = 1000$ and $|D|$ ranges from $100,000$ to $1,000,000$.

Note that both DMA and PDM can also be naively extended to solve the N common association rules. While applying DMA when $N > 1$, the number of candidates generated in each iteration is smaller than that in the corresponding iteration for $N = 1$. This is also true for MNA and PDM. Therefore, given a minimum support value s and $N > 1$, the time relationship among DMA, PDM, and MNA should be similar to that among DMA, PDM, and MNA for a larger value s_0 (i.e., $s_0 > s$) and $N = 1$. This means that MNA should be most efficient as well when $N > 1$.

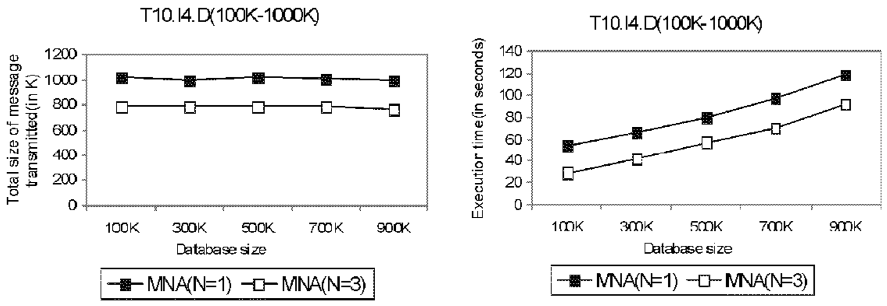


Fig. 5. Results of the experiment 2 ($s\% = 0.5\%$)

5 Conclusion

In this paper, we studied the problem of mining distributed association rules, a generalized problem of mining association rules. We proposed an efficiently distributed algorithm MNA to solve this problem. Our experiment results suggest that if restricted to the association rules problem, the algorithm MNA is more efficient than the existing algorithms DMA and PDM.

We recently also developed Web access pattern mining algorithms based on PLO-DHP [6]. As a future study, we will develop efficient distributed algorithms to solve Web data mining problem.

References

1. R. Agrawal and J. C. Shafer, Parallel Mining of Association Rules: Design, Implementation, and Experience, *IEEE Transactions on Knowledge and Data Engineering*, 8(6), 962-969, 1996. 109, 111
2. R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, *Proceedings of the 20th VLDB Conference*, 487-499, 1994. 109, 111, 113, 114
3. R. Agrawal and R. Srikant, Mining Sequential Patterns, *Proceedings of the 11th International Conference on Data Engineering*, 1995.
4. D. W. Cheung, Vincent T. Ng, Ada W. Fu and Y. Fu, Efficient Mining of Association Rules in Distributed Databases, *IEEE transactions on Knowledge and Data Engineering*, 8(6), 884-897, 1996. 109, 110, 111, 112, 113, 114
5. U. M. Fayyad, G.Piatetsky-Shapiro, P.Smyth, and R.Uthurusamy. *Advances in Knowledge Discovery and Data Mining*, AAA/MIT Press, 1996.
6. X.Lin , Y.Li and C. P.Tsang, Applying On-Line Bitmap Indexing to Reduce Counting Costs in Mining Association Rules, *Information Science*, 120 (1-4), 197-208, 1999. 109, 110, 112, 113, 114, 119
7. Y.Li, On Efficient Computation of Association Rules, *Master Thesis*, University of New South Wales, 2000. 110, 111, 112, 114
8. R. Ng and J. Han, Efficient and Effective Clustering Method for Spatial Data Mining, *Proc Int'l Conf. Very Large Data Bases*, 144-155, Santiago, Chile, Sept, 1994.

9. J. S. Park, M. S. Chen, and P. S. Yu , An Effective Hash-Based Algorithm for Mining Association Rules, *Proc. ACM-SIGMOD Int'l Conf. Management of Data*, 175-186, 1995. 109, 110, 111, 112, 113, 114
10. J. S. Park, M. S. Chen, and P. S. Yu, Efficient Parallel Data Mining for Association Rules, *Proc. Int'l Conf. Information and Knowledge Management*, 1995. 109, 110, 111, 113
11. A. Savasere, E.Omiecinski, and S.Navathe, An Efficient Algorithms for Mining Association Rules in Large Databases, *Proc. Int'l Conf. Very Large Data Bases*, 432-444, Zurich, Sept. 1995. 109, 111

Mining Association Rules with Negative Items Using Interest Measure[⋆]

Haofeng Zhou, Pan Gao, and Yangyong Zhu

Department of Computer Science, Fudan University
Shanghai, 200433, P.R.China
haofzhou@fudan.edu.cn

Abstract. In this paper, we analyze some potential problems in the existing mining algorithms on association rules. These problems are caused by only concerning about its support and confidence, while neglecting to what extent the rule will interest people. At the same time, the existing definition and mining algorithms of association rules does not take into account any negative items, therefore many valuable rules are lost. We hereby introduce the concepts of interest measure and negative item into the definition and evaluation system. Then we modify the existing algorithms so as to use interest measure to generate rules with negative items. At the end of this paper we analyze the new algorithm and prove it to be efficient and feasible.

1 Introduction

Mining association rules from database is mining rules with the following implication: some events occur or not because of the occurrence or nonoccurrence of other things at the same time [1]. It is quite useful in decision support systems, expert systems and so on. While database systems are playing more and more important roles in all kinds of fields, the urge for data mining applications are becoming more and more eager. Before we begin to explain our work in detail, we have to clarify some basic concepts.

Definition 1. *Association Rule*
Let $I = \{i_1, \cdots, i_m\}$ be a set of positive literals, where $i_j (1 \leq j \leq m)$ is called an item, $D = \{T_1, \cdots, T_n\}$ be the transaction set, where $T_i \subseteq I (1 \leq i \leq n)$ is a transaction. The association rule is an implication formula like:

$$X \Rightarrow Y \ , \tag{1}$$

where $X \subset I$, $Y \subset I$ and $X \cap Y = \emptyset$.

In order to evaluate the association rules, usually we need two additional parameters, which are support and confidence, respectively. Let us define the projection of a transaction set first.

[⋆] This paper was funded by the National 863 Projects of China No. 863-306-ZT02-05-1.

Definition 2. *Projection*
Given literal set I and transaction set D, the projection of D on itemset $X \subseteq I$ is:

$$D_X = \{T | T \in D \wedge X \subseteq T\} . \tag{2}$$

Definition 3. *Support of an association rule*
Given transaction set D and an association rule $X \Rightarrow Y$ on it, the support of $X \Rightarrow Y$ is:

$$s = \frac{|D_{X \cup Y}|}{|D|} , \tag{3}$$

while the domain of s is M_s.

Definition 4. *Confidence of an association rule*
Given transaction set D and an association rule $X \Rightarrow Y$ on it, the confidence of $X \Rightarrow Y$ is:

$$c = \frac{|D_{X \cup Y}|}{|D_X|} , \tag{4}$$

while the domain of c is M_c.

It can be shown easily using formula (3) and (4) that

$$c(X \Rightarrow Y) = s(X \Rightarrow Y)/s(X) . \tag{5}$$

Usually we are only interested in association rules with support and confidence high enough. Only rules with support and confidence higher than given thresholds are qualified. Such kind of rules are called strong association rules.

Definition 5. *Strong association rule, minimal support, minimal confidence*
Given transaction set D and an association rule $X \Rightarrow Y$ on it, $minconf \in M_c$, $minsupp \in M_s$, we call $X \Rightarrow Y$ a strong association rule if and only if $c \geq minconf \wedge s \geq minsupp$. We call $miniconf$ the minimal confidence, and the $minsupp$ the minimal support.

2 Description of the Potential Problems

However, we may face such problems as shown in Table 1, when we apply the current definition to practice.

Table 1. A possible analysis result of a transaction set

%	Buy Coffee	Do not Buy Coffee	Subtotal
Buy milk	20	5	25
Do not Buy Milk	70	5	75
Subtotal	90	10	100

Suppose we have $I = \{Coffee, Milk\}$, transact set D, and get the analysis result as shown in Table 1. If we set $minsupp = 0.2$,$minconf = 0.6$, we will get the following association rule using the existing mining algorithms:

$$Buy\ Milk \Rightarrow Buy\ Coffee \quad s = 0.25 \quad c = 0.8\ , \tag{6}$$

which means that *80% people who buy milk also buy coffee*. This is quite reasonable and understandable in logic. On the other hand, we can draw the conclusion from the table that *90% people buy coffee*. In other words, the stimulative effect of buying milk on buying coffee (80%) is not as much as we expect (90%). Instead, the rule

$$Buy\ Coffee \Rightarrow Do\ not\ Buy\ Milk \quad s = 0.7 \quad c = 0.78 \tag{7}$$

has a support s of 0.7 and confidence c of 0.78, thereby is much more instructive in business.

We can conclude from the example that the current evaluation system of association rules based on support and confidence has some problems. The existing mining algorithm can only generate rules having forms like (6), while cannot generate rules having forms like (7), which has negative item such as *do not buy milk* appearing. Sometimes, as shown in the example, this kind of knowledge has more potential value. Therefore, we need to improve or redefine the existing system of association rules.

3 Redefinition of Association Rules and Its Evaluation System

There has been some research work [3] done related to the usage of negative items in association rules. It pointed out that we can compute the literal set I's corresponding literal set $NI = \{\bar{i}_1, \cdots \bar{i}_m\}$, then define $I' = I \cup NI$ as the new literal set, from which we can do further computing. Other works do not modify I, but introduce negative items into the T in D. Neither changes the formal definition of the association rules, thus bringing the exceptional phenomenon as described below:

Suppose we have $I' = I \cup NI$, given proper $minsupp$ and $minconf$, it is quite possible that we will generate association rules having forms like:

$$\bar{i}_1 \cdots \bar{i}_j \Rightarrow \bar{i}_{j+1} \cdots \bar{i}_k \qquad i_j \in I\ . \tag{8}$$

All items appearing in the rule are negative, for example: *Do not buy milk* ⇒ *Do not buy coffee*, which is hard to understand and has little practical value when decision makers try to use it. The similar situation occurs if rules like *Do not buy milk* ⇒ *Buy coffee* are generated. The only explanation of it would be that customers may be interested in coffee, while no reasonable inherent relations between milk and coffee can be found. Therefore, to define the association rules, we should take into account some knowledge when formalizing it, instead of doing nothing except simply introducing the negative items.

3.1 Redefinition of Association Rules

We present the reasonable definition of association rules which takes account of negative items:

Definition 6. *Association rules (with negative items)*
Let $I = \{i_1, \cdots, i_m\}$ *be a set of positive literals, while each* $i_j (1 \leq j \leq m)$ *is called a positive item, its corresponding* $\overline{i}_j (1 \leq j \leq m)$ *is called a negative item;* $D = \{T_1, \cdots, T_n\}$ *be the transcation set, while each* $T_i \subseteq I(1 \leq i \leq n)$ *is a transaction. The association rule is an implication formula like:*

$$XY \Rightarrow Z \ , \tag{9}$$

where $X_i \subset I$ *and* $X \not= \emptyset$,$Y = \{\overline{i} | i \notin X \wedge i \in I\}$, $Z = \{i | i \in I \vee \overline{i} \in I\}$,$(X \cup Y) \cap Z = \emptyset$.

This definition introduces negative literals (called negative items) to the association rules. Such negative items can appear in both head (right side) and body (left side). And there should be at least one original positive items appearing at the body. Positive items can either appear at head or do not appear at head. Thus we can have such rules as:

$Buy\ computer \wedge Buy\ Windows\ 98 \wedge Do\ not\ buy\ mouse \Rightarrow Buy\ track\ ball$

The new definition prevents the meaningless rules like formula (8) from being generated, for it forces at least one positive item appearing at the body. Further more, usually the transaction sets in real world only record what happened instead of what did not happen. Experts and decision makers can only predict the future based on what happened. Therefore, in practice we can ignore the Y part in the definition so that the definition reduces to $X \Rightarrow Z$.

If we really recorded negative events in transaction set, while literal set is still positive set, we can guarantee that positive event and its corresponding negative event will not co-exist in the same transaction record, otherwise we think it illegal to be deleted. Then we can convert the negative event (item) to a special positive event and add it to the original literal set, forming a new literal set. Then we can apply the current existing algorithms to it and generate rules. We should point out that we should not modify any transaction in a transaction set, especially we should not add events that do not appear in the original record. We can consider what we really know, but should not assume or guess the unknown things. In practice, only in rare occasions we record what does not happen.

If not mentioned explicitly, the *association rules* appearing in the remainder part of this paper follow the reduced definition of $X \Rightarrow Z$.

Let's consider what we can do to improve the evaluation system of association rules since we have introduced negative items.

In the example mentioned in the previous part, we generate the incorrect rule of $Buy\ milk \Rightarrow Buy\ coffee$, partly because we didn't taking into account to what extend the rule will interest people (compared with other rules, such as $Buy\ milk \Rightarrow Do\ not\ buy\ coffee$, or compared with the extent people expect). It is necessary to introduce the interest measure to the evaluation system of association rules.

3.2 Introducing Interest Measure to the Evaluation System

Many related works mentioned the concept of interest measure, presenting different definition formulas. The one in Reference [2] is:

$$I_R = \frac{C_R - S_Y}{\max\{C_R, S_Y\}} \ . \tag{10}$$

The denominator $\max\{C_R, S_Y\}$ is just a standardization factor, which makes $|I_R| < 1$. The author of the paper thinks that the more a rule's interest measure is higher than 0, the more we are interested in that rule. On the contrary, the more it is lower than 0, the more we are interested in it's reversed rule. But this paper did not define the reversed rule, thus it is still a key problem how to generate the potential more valuable rule when we found $I_R < 0$. At the same time, the interest measure as defined in the formula presented is not quite understandable and lacks of intuitive meaning. So we'll redefine the interest measure after we analyze these problems.

For convenience of narrating, we use P to replace support s. $P(X)$ mean the percentage of transactions that contains X appearing in the transaction set D. It is called frequency in the Probability. When the sample data set is large enough, the frequency approximates the probability. So we can use frequency $P(X)$ to replace the probability of a transaction containing X. We can also easily derive that the confidence $c(XY \Rightarrow Z) = P(XY)/P(X) = P(Y|X)$, which is the conditional probability of Y occurring under the condition of X.

In formula (10), the denominator $\max\{C_R, S_Y\}$ is a standardization factor, which can be ignored, so:

$$\begin{aligned} I'_R = C_R - S_R = P(Y|X) - P(Y) &= (P(XY)/P(X)) - P(Y) \\ &= (P(XY) - P(X)P(Y))/P(X) \ . \end{aligned}$$

We are considering the relation between I'_R and 0, which is the same as the relation between $P(XY) - P(X)P(Y)$ and 0, or $P(XY)/P(X)P(Y)$ between 1. We base our definition of interest measure on this relation.

Definition 7. *Interest measure*
Given transaction set D and the association rules $X \Rightarrow Y$ on D, the interest measure of $X \Rightarrow Y$ is

$$i = \frac{P(XY)}{P(X)P(Y)} \ , \tag{11}$$

while the domain of i is M_i.

This definition reminds us the concept of independence between events. In fact these two concepts are similar. When two event happen independently, it is only accidental if they happen at the same time, and people cannot find reasonable explanation for it. Reference [4] already point out that when $P(XY) \approx P(X)P(Y)$, the corresponding association rule is meaningless.

Obviously that $i = 1$ if and only if $I_R = 0$.

Similarly, the more a rule's interest measure is higher than 1, the more we are interested in that rule. On the contrary, the more it is lower than 0, the more we are interested in it's reversed rule. i will never be lower than 0. The reason is shown as below:

Suppose we have association rules R_1: $X_1 \Rightarrow Y_1$ and R_2: $X_2 \Rightarrow Y_2$, their corresponding old and new interest measures are I_{R_1}, I_{R_2}, i_1, i_2, respectively. Suppose $I_{R_1} > I_{R_2}$, which does not lose generality, we can derive that $i_1 > i_2$ as following.

When $I_{R_1}, I_{R_2} > 0$,we have $\max\{C_{R_1}, S_{Y_1}\} = C_{R_1}$,$\max\{C_{R_2}, S_{Y_2}\} = C_{R_2}$, so $(C_{R_1} - S_{Y_1})/C_{R_1} > (C_{R_2} - S_{Y_2})/C_{R_2} \Rightarrow C_{R_1}/S_{Y_1} > C_{R_2}/S_{Y_2}$,therefore $P(X_1Y_1)/P(X_1)P(Y_1) > P(X_2Y_2)/P(X_2)P(Y_2)$, which is the same as $i_1 > i_2$.

When $I_{R_1}, I_{R_2} < 0$, we have $\max\{C_{R_1}, S_{Y_1}\} = S_{Y_1}$,$\max\{C_{R_2}, S_{Y_2}\} = S_{Y_2}$, so $(C_{R_1} - S_{Y_1})/S_{Y_1} > (C_{R_2} - S_{Y_2})/S_{Y_2} \Rightarrow C_{R_1}/S_{Y_1} > C_{R_2}/S_{Y_2}$, we also can get $i_1 > i_2$.

Now we reconsider the previous example of buying milk and coffee, taking account into the interest measure. We list all rules which may be generated and its corresponding support s, confidence c and interest measure i, as shown in Table 2.

Table 2. All possible association rules

	Rules	s	c	i
1	$Buy\ milk \Rightarrow Buy\ coffee$	0.2	0.8	0.89
2	$Buy\ coffee \Rightarrow Buy\ milk$	0.2	0.22	0.89
3	$Buy\ milk \Rightarrow No\ not\ buy\ coffee$	0.05	0.2	2
4	$No\ not\ buy\ milk \Rightarrow Buy\ milk$	0.05	0.5	2
5	$No\ not\ buy\ milk \Rightarrow Buy\ coffee$	0.7	0.93	1.037
6	$Buy\ coffee \Rightarrow No\ not\ buy\ milk$	0.7	0.78	1.037
7	$No\ not\ buy\ milk \Rightarrow No\ not\ buy\ milk$	0.05	0.067	0.67
8	$No\ not\ buy\ milk \Rightarrow No\ not\ buy\ milk$	0.05	0.2	0.87

Here we only need to consider the first, second, third, and sixth rules, because of the constraint we mentioned before. Because $i_1, i_2 < 1$, they do not have much value in practice. Because $i_3, i_6 > 1$, they can be further considered.

Just the same as the support and confidence, interest measure has threshold, too. After we decide the threshold, we can redefine the *Strong Association Rules*.

Definition 8. *Strong association rules with negative items and the minimal interest measure*
Given the transaction set D and an association rule $X \Rightarrow Y$ on it, $minconf \in M_c$, $minsupp \in M_s$, $minint \in M_i$, we call $X \Rightarrow Y$ a strong association rule if and only if $c \geq minconf \wedge s \geq minsupp \wedge i \geq minint$. We call minint the minimal interest measure.

It should be noted that $minint > 1$, as explained in the previous part.

From this definition, we can see that to generate this kind of association rules, we should take into account of the interest measure, as well as support and confidence.

4 Algorithms Dealing with Negative Items

4.1 Modifying Existing Algorithms

Many existing mining algorithms are based on positive literals. We should solve 2 problems if we want to generate rules with negative items.

1. How to import negative items to the algorithm to generate such association rules.
2. How to assess the association rules using it's interest measure.

A common and trivial method to solve the first problem is to expand the literal set, and reconstruct the transaction set using the new literal set containing negative items, as described below: Given $I = \{i_1, i_2, \cdots, i_m\}$, $D = \{T_1, T_2, \cdots, T_n\}$, $T_j \subseteq I$, let $\bar{I} = \{\bar{i}_1, \cdots, \bar{i}_m\}$, $T'_j = T_j \cup \{\bar{i} | i \notin T_j \wedge \bar{i} \in \bar{I}\}$, $D' = \{T'_1, \cdots, T'_n\}$, then, we can apply existing mining algorithms to D' to generate association rules.

But its disadvantage is to expand the sample space so much that item number in each transaction in the transaction set is increased to a much larger number. For instance, suppose $|I| = 1000$,$|D| = 10000$, averagely $|T| = 5$. If we use this method, we will have $|T'| = 1000$, which is 200 times larger than before. This will be an extremely heavy burden. Therefore we do not recommend to import negative items at the beginning of the algorithm.

As for the second problem, we can use interest measure as a threshold to filter the rules after their support and confidence have computed and passes the corresponding thresholds. The rules with lower interest measure as compared with the minimal interest measure are filtered. But what else can we do with these rules, since we have already shown that lower interest measure is a bridge leading to rules with negative items? It is really a waste if we only use it as a filter to prune unqualified rules.

On the other hand, many existing algorithms have matured from years' of proving and practicing, which are worthy of being made full use of. We can get twice the result with half the effort if we alter these existing algorithms so as to solving the 2 problems pointed out instead of beginning from scratch.

Recalling the existing algorithms, we can discover that it can be divide into 2 phases, in which the first phase during which frequent sets are produced are of much importance. We develop our algorithm here, dividing the mining work into 3 phases.

1. Produce the frequent sets that contain positive literals only. We can use the existing algorithms to finish this job.

2. Use the minimal confidence and minimal interest measure as the thresholds to filter candidate rules. The rules which passes the filtering are those ordinary rules without negative items.
3. For those rules which are filtered because of low interest measure, considering if association rules with negative items can be produced from them. Check the confidence and interest measure of these new candidates, if qualified, output them as rules with negative items.

For the convenience of narration, we need to introduce a new concept called *negative itemset.*

Definition 9. *Negative Itemset*
Given a positive literal set $I = \{i_1, \cdots, i_n\}$*, a itemset* $S \subseteq I$*, we call* $S_N = \{i'_1, \cdots, i'_m\}$ *a negative itemset of* S *if and only if* $i'_j \in S \vee \bar{i}'_j \in S (1 \le j \le m \le n)$

In other words, S_N is an itemset that is obtained when we replace some of the positive literals with their corresponding negative ones in S.

Negative itemset also has its own support, but if we use the same method to compute it, we will have to re-scan the database, which will cause a great impact on the computing speed. Suppose we have $|S| = n$, we get altogether $2^n - 1$ possible negative sets. It is unendurable to re-scan the database for so many times. Fortunately we do not have to re-scan the whole transaction set D in order to obtain its support. Instead, we will show that we can compute the support of a negative set based on the support of positive literal sets we have already computed.

Theorem 1. *Given frequent set* X *and its support* $P(X)$*, we can compute the support of each one of its negative sets* X' *using the support of* X *and* X*'s subsets.*

Proof. Suppose $X = B \cup A$, while B is the set constructed by all positive literals appearing in X', $A = X - B$, and $A \cap B = \emptyset$, $A = \{A_1, \cdots, A_n\}$.

1. When $B = \emptyset$, $P(X') = P(\bar{A}_1 \cdots \bar{A}_n)$. It is obvious that

$$P(\bar{A}_1 \cdots \bar{A}_n) = 1 - \sum_{i=1}^{n} P(A_i) + \sum_{i<j} P(A_i A_j) - \cdots + (-1)^n P(A_1 \cdots A_n) \ , \quad (12)$$

which proves our conclusion.

2. When $B \neq \emptyset$, we will show by induction that

$$\begin{aligned} P(B\bar{A}_1 \cdots \bar{A}_n) = P(B) - \sum_{i=1}^{n} P(BA_i) + \sum_{i<j} P(BA_iA_j) \\ - \cdots + (-1)^n P(A_1 \cdots A_n) \ . \end{aligned} \quad (13)$$

   (a) When $n = 1$,$P(B\bar{A}_1) = P(B) - P(BA_1)$, the equation (13) holds.

(b) Suppose equation (13) holds when $n = k$,so when $n = k + 1$

$$\begin{aligned} & P(B\bar{A}_1 \cdots \bar{A}_k \bar{A}_{k+1}) = P(B\bar{A}_1 \cdots \bar{A}_k) - P(B\bar{A}_1 \cdots \bar{A}_k A_{k+1}) \\ & = P(B\bar{A}_1 \cdots \bar{A}_k) - [P(BA_{k+1}) - \sum_{i=1}^{k} P(BA_{k+1}A_i) \\ & \qquad + \sum_{i<j} P(BA_{k+1}A_iA_j) - \cdots + (-1)^k P(BA_{k+1}A_1 \cdots A_k)] \\ & = P(B\bar{A}_1 \cdots \bar{A}_k) - P(BA_{k+1}) + \sum_{i=1}^{n} P(BA_{k+1}A_i) \\ & \qquad - \sum_{i<j} P(BA_{k+1}A_iA_j) - \cdots + (-1)^{k+1} P(BA_{k+1}A_1 \cdots A_k)] \\ & = P(B) - \sum_{i=1}^{k+1} P(BA_i) + \sum_{i<j} P(BA_iA_j) - \cdots + (-1)^{k+1} P(A_1 \cdots A_n) , \end{aligned}$$

which means that equation (13) still holds.

(c) Now we can derive that for each n, equation (13) holds, which means that negative itemset X''s support $P(X')$ can be calculated using the support of X and X's subsets.

3. Either $B = \emptyset$ or $B \neq \emptyset$, we reach our conclusion.

Recalling the 3 phases to produce association rules with negative items, we can use any existing algorithm to produce the frequent sets in phase 1. The key of our algorithm is the phase 2 and 3. We now present it as below, using the similar structure as the one in the Apriori algorithm. For a detail explanation of the original Apriori algorithm itself, see Reference [5].

Algorithm 1. Negative rule generating algorithm (NRG)
Input: literal set I, transaction set D, minimal support *minsupp*, minimal confidence *minconf*, minimal interest *minint*, and large itemsets L.
Output: association rule set R
Method:
L_k: large k-itemsets with 2 fields (itemset,support_count)

```
For all large $k$-itemset $l_k \in L_k$  $k \geq 2$ do
  Begin
    $H_1$=consequents of rules form $l_k$ with one item in the consequent;
    Call ap-genrules($l_k$,$H_1$);
  End
$R = \cup_k R_k$;

Procedure ap-genrules($l_k$: large k-itemsets,$H_m$:set of m-item consequents)
  Begin
    If ($k > m + 1$) Then
      Begin
        $H_{m+1}$=apriori-gen($H_m$); //Refer to the original Apriori algorithm
        For all $h_{m+1} \in H_{m+1}$ do
          Begin
```

```
            conf=support($l_k$)/support($l_k - h_{m+1}$);
            If (conf< $minconf$) then Delete $h_{m+1}$ from $H_{m+1}$;
            Int=support($l_k$)/(support($l_k - h_{m+1}$)*support($h_{m+1}$));
            If Int> $minint$ then
              If (conf≥ $minconf$) then
                $R_k = R_k \cup \{(l_k - h_{m+1}) \Rightarrow h_{m+1}\ with\ support(l_k), conf, Int\}$
            Else
              If Int< 1 then
                For all negative itemsets $h_{m+1_N}$ of $h_{m+1}$ do
                  Begin
                    s=support($(l_k - h_{m+1}) \cup h_{m+1_N}$);
                    c=s/support($l_k - h_{m+1}$);
                    i=c/support($h_{m+1_N}$);
                    If s≥ $minsupp$ AND c $\geq minconf$ AND i $\geq minint$
                    Then $R_k = R_k \cup \{(l_k - h_{m+1}) \Rightarrow h_{m+1_N}\ with\ s, c, i\}$
                  End
          End
        Call ap-genrules($l_k$,$H_{m+1}$);
      End;
    End;
```

The key of this algorithm is that when interest measure of combination of $l_k - h_{m+1}$ and h_{m+1} reaches the specified threshold, this combination becomes a potential rule whose confidence is to be checked. Otherwise we come to consider if one of its negative itemsets can construct a strong association rule. These negative itemsets forms a series of potential frequent sets. Evaluate the 3 parameters of each potential frequent set, we can decide if it can produce its corresponding association rule. The process of obtaining negative itemsets reduces to the process of obtaining subsets. Each negative itemset's support can be computed using the support of the original positive itemset and it's subsets, using the method presented in Theorem 1. We should note that here in our NRG algorithm, $H_k(k \in 1, 2, \cdots)$ has the same meaning as before. When we generate the rule with only one consequence, we first use minimal confidence to filter all candidate rules, the consequences of those passing the filtering form H_1, then we use interest measure to filter the candidate rules, if necessary, generating rules with negative items. Note that the elements in H_1 need not corresponding rules. For example, we may have a candidate rule $ABC \Rightarrow D$, whose confidence and support are both high enough. So we put D into H_1. But when we examine its interest measure, we may find it unqualified. So it is possible that D appear in H_1, but we don't have any rule with a consequence or right hand side D. After we generate rules with one consequence and H_1, in subroutine apriori-gen we produce H_{m+1}. We delete h_{m+1} from H_{m+1} only when we find it does not have enough confidence, which is the same as in the original Apriori algorithm. Therefore in our algorithm, for all $k, k \in \{1, 2, \cdots\}$, H_k have exactly the same meaning as the original Apriori algorithm.

4.2 Evaluating the Algorithm

Compared with other generic mining algorithms, this algorithm will automatically introduce negative itemsets to produce potentially interesting rules when the rules founded are not interesting. This is its functional feature. Compared with the algorithms mentioned in Reference [3], since it does not expand the original literal set or modify the transaction set, it has a much better performance as well as a lighter data preparation burden. Compared with the algorithm mentioned in Reference [7], it is simpler because it can do further analysis to the uninteresting rules directly without the help of field knowledge.

It is obvious that its performance is not as good as those generic mining algorithms such as Apriori, AprioriTID and DHP, because this algorithm inherits the steps used to produce frequent sets in other algorithms such as Apriori intactly, while adding steps to deal with interest measures and produce negative itemsets. In order to have a better performance, we can adopt DHP or AprioriTID algorithm to produce positive frequent sets.

The most complicated part of it is to compute the support of the negative itemsets. Since we base our computation on the already computed support of the positive literals, we can use the existing results when we compute these positive literals, so that re-scan database is avoided, which is far more efficient or, at least, more feasible.

Compared with the general Apriori, we need to do additional computation to produce negative itemsets when we find the interest measure is lower than our criteria. As stated above, this process comes down to a process of computing subsets. When an itemset B has $|B| = n$, it can have up to $2^n - 1$ negative itemsets. The count of itemsets containing k negative items is C_n^k $(1 \leq k \leq n)$. For each negative itemset containing k negative items, we need $C_k^0 + \cdots + C_k^k = 2^k$ support values to produce its support. Therefore, every time we meet an itemset having a low interest measure, totally $\sum_{j=1}^{n} C_k^j 2^j$ computations are needed to obtain all frequent sets it imports. But since this part of computation mainly involves accessing the Hash tree in memory, it does not affect the performance or the computing speed a lot.

Besides considering how much computation we'll do when uninteresting rules are encountered, we should also consider whether computing negative itemsets is a quite probable thing. To illuminate this question, we did a small-sized test. We applied Apriori algorithm to a transaction database with $|D| = 5000$ and $|I| = 200$. The data in it comes from stock exchanges. We do the data preparation as following: if one person buys or sells stocks within 60 minutes, we think these actions belong to the same transaction. Each action of buying or selling can only belong to one transaction. Even so, many transactions T in this database only contain 1 item. Setting *minsupp* and *minconf* to 1 and 0, respectively, we apply our algorithm to them and check the interest measures of the rules we got. If we met a rule with interest measure lower than 1, we increment the counter in order to find out the possibility of occurance of this kind of rules. In order to get more frequent sets, we decrease the value of *minsupp* step by step and the final result is shown in Table 3.

Table 3. Analysis of the probability of occurance of negative itemsets

Minimal support	0.008	0.007	0.006	0.005	0.004	0.003
Number of possible candidate rules	4	8	18	34	56	72
Number of rules lack interest	0	0	2	2	4	4
Percentage(%)	0	0	11.1	5.9	7.1	5.6

We find out from the analysis that there is few need to compute negative itemset in practice. And even when the computation is needed, we do not need to re-scan the database, so the efficiency is acceptable. In the testing environment mentioned above, the scale of D is a little too small, we are considering to increase the scale under a better hardware and software platform to re-do this test in order to get more accurate result. At the same time, we find that Reference [6] pointed out that in general, the possibility of $P(XY) \approx P(X)P(Y)$ is quite small, using a hypothesis testing method. We are trying to analyze the probability in theory.

5 Conclusion

In this paper, we analyze some potential problems existing in the existing association rules mining algorithms. These problems are caused by only concerning about its support and confidence, while neglecting to what extent the rule will interest people. At the same time, the existing definition and mining algorithms of association rules does not take into account any negative items, thus many valuable rules are lost. We hereby introduce the interest measure and negative items into the definition and evaluation system of association rules. Then we modify the existing algorithms so as to use interest measure to generate rules with negative items. At last we analyze the new algorithm and prove it to be efficient and feasible.

References

1. Agrawal, R., Imielinski, T., Swami, A. N.: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference (1993) 207-216 121
2. Zhou, X., Zhu, Y., Shi, B.: A Method for Mining Association Rules Using Interest Measure. ICYCS (1999) 125
3. Zuo, W., Liu, J.: Mining Association Rules Involving Positive and Negative Attributes. NDBC (1999.8) 288-292 123, 131
4. Piatetsky-Shapiro, G.: Discovery, Analysis, and Presentation of Strong Rules. Knowledge Discovery in Database. AAAI/MIT Press (1991) 229-248 125
5. Agrawal, R., Mannila, H., Srikant, R.,Toivonen, H., Verkamo, A. I.: Fast Discovery of Assoication Rules. In: Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996) 307-328 129
6. Megiddo, N., Srikant, R.: Discovering Predictive Association Rules. KDD (1998) 274-278 132
7. Savasere, A., Omiecinski, E., Navathe, S. B.: Mining for Strong Negative Associations in a Large Database of Customer Transactions. ICDE (1998) 494-502 131

Probabilistic Approach to Association Rules in Incomplete Databases

Marzena Kryszkiewicz

Institute of Computer Science, Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
mkr@ii.pw.edu.pl

Abstract. In the paper we list a set of properties that characterize a legitimate approach to data incompleteness. An example of a legitimate probabilistic approach, which is based on attribute distribution, is presented. We also review and compare three other approaches to incompleteness: the one that ignores missing values, the approach applying only certain information, and the approach based on valid databases. All the three approaches turn out to be invalid.

1 Introduction

The problem of association rules discovery was introduced in [1] for sales transaction database. An exemplary association rule will state that 80% of customers who buy fish buy white wine as well. The problem of data incompleteness did not occur for a transaction database, however, it is often unavoidable in relational databases. Missing data may result from errors, measurement failures, changes in the database schema etc.

Several solutions to this problem have been proposed in the classification context in the area of artificial intelligence. The simplest among solutions consist in removing examples with unknown values or replacing unknown values with the most common values. More complex approaches were presented in [2, 4]: [2] uses a Bayesian formalism; [4] predicts the value of an attribute based on the value of other attributes of the example, and the class information.

The papers [3, 5] address the data incompleteness in the context of association rules. It was offered in [3] how to compute pessimistic and optimistic estimations of support and confidence of an association rule. Additionally, there were introduced notions of expected values of support and confidence based on known attribute values in the database. The main idea presented in [5] was to cut a database into several valid databases without missing values.

In the paper we list a set of properties that characterize a legitimate approach to incompleteness. If at least one of the specified properties does not hold then the approach cannot be treated as valid. It is presented in the paper an example of a legitimate probabilistic approach, which is based on attribute distribution. We also review some other approaches to incompleteness, namely: an approach that ignores the presence of missing values, the notions of expected support and confidence as introduced in [3], and the approach based on valid databases. All of them turn out to be invalid.

2 Association Rules in Complete Relational Databases

Let us consider a table D = (O, AT), where O - is a non-empty finite set of *tuples* and AT is a non-empty finite set of *attributes*, such that a: O $\rightarrow V_a$ for any $a \in AT$, where V_a denotes the domain of a. Any *attribute-value pair* (a,v), where $a \in AT$ and $v \in V_a$ will be called an *item*. A set of items will be called *itemset*. Statistical significance of an itemset X is called *support* and is denoted by $sup(X)$. $sup(X)$ is defined as the percentage (or the number) of transactions in D that contain X.

An *association rule* is an expression of the form: $X \Rightarrow Y$, where X and Y are items and $X \cap Y = \emptyset$. Statistical significance (*support*) of a rule $X \Rightarrow Y$ is denoted by $sup(X \Rightarrow Y)$ and is defined as $sup(X \cup Y)$. Strength of the rule $X \Rightarrow Y$ is called *confidence* and is denoted by $conf(X \Rightarrow Y)$. $conf(X \Rightarrow Y)$ is defined as $sup(X \cup Y) / sup(X)$.

3 Uncertainty Caused by Data Incompleteness

Computation of real support of itemsets and real confidence of association rules is infeasible in the case of databases with missing attribute values. However, one can always calculate possible least and greatest values of support and confidence as well as try to predict their most probable values. We will apply the following notions in order to express the properties of data incompleteness:

- Missing values will be denoted by "*" ,
- The *maximal set of tuples that match an itemset X necessarily* is denoted by $n(X)$ and is defined as follows: $n(X) = \{t \in D | \forall (a,v) \in X: a(t)=v\}$,
- By $m(X)$ we denote the *maximal set of tuples that may match the itemset X* in D, i.e. $m(X) = \{t \in D | \forall (a,v) \in X: a(t) \in \{v,*\}\}$,
- The set-theoretical difference $m(X) \setminus n(X)$ is denoted by $d(X)$,
- By $n(-X)$ we denote the *maximal set of tuples that certainly do not match the itemset X* in D, i.e. $n(-X) = D \setminus m(X)$,
- By m(-X) we denote the *maximal set of tuples that may not match the itemset X* in D, i.e. $m(-X) = D \setminus n(X)$.

(See [3] for properties of these notions.)

Example 3. Given the incomplete database D presented in Fig. 1, we will illustrate the introduced notions of necessary and likely itemsets matching.

Id	X1	X2	X3	X4
1	*	a	a	c
2	a	a	b	*
3	a	b	c	c
4	a	b	d	c
5	*	b	e	d
6	b	b	f	*
7	b	c	g	*
8	b	c	h	*

Fig. 1. Exemplary database with missing values

Fig. 2 contains the respective sets of tuples that match exemplary itemsets.

Itemset *X*	*n*(*X*)	*m*(*X*)	*d*(*X*)	*n*(-*X*)	*m*(-*X*)
{(X1,a)}	{2,3,4}	{1,2,3,4,5}	{1,5}	{6,7,8}	{1,5,6,7,8}
{(X4,c)}	{1,3,4}	{1,2,3,4,6,7,8}	{2,6,7,8}	{5}	{2,5,6,7,8}
{(X1,a),(X4,c)}	{3,4}	{1,2,3,4}	{1,2}	{5,6,7,8}	{1,2,5,6,7,8}
{(X1,a),(X2,b)}	{3,4}	{3,4,5}	{5}	{1,2,6,7,8}	{1,2,5,6,7,8}
{(X2,b),(X4,c)}	{3,4}	{3,4,6}	{6}	{1,2,5,7,8}	{1,2,5,6,7,8}
{(X1,a),(X2,b),(X4,c)}	{3,4}	{3,4}	∅	{1,2,5,6,7,8}	{1,2,5,6,7,8}

Fig. 2. Tuples matching itemset *X*

Let us note that e.g. tuple 5 in Fig. 1 does not match the itemset {(X1,a),(X4,c)} certainly, though the value of attribute X1 is missing. □

Let *least possible support* of an itemset *X* be denoted by *pSup*(*X*) (*pessimistic* case) and *greatest possible support* be denoted by *oSup*(*X*) (*optimistic* case). Clearly,

$$pSup(X) = |n(X)| \,/\, |D| \,, \tag{1}$$

$$oSup(X) = |m(X)| \,/\, |D| \,. \tag{2}$$

Let $X \subset Y$. It is easy to observe that, $n(X) \supseteq n(Y)$ and $m(X) \supseteq m(Y)$. Hence, $pSup(X) \geq pSup(Y)$ and $oSup(X) \geq oSup(Y)$.

Let $pConf(X \Rightarrow Y)$ and $oConf(X \Rightarrow Y)$ denote *least possible confidence* and *greatest possible confidence* of $X \Rightarrow Y$, respectively. These values can be computed according the following equations (see [3] for proof):

$$pConf(X \Rightarrow Y) = |n(X) \cap n(Y)| \,/\, [|n(X) \cap n(Y)| + |m(X) \cap m(\text{-}Y)|] \,, \tag{3}$$

$$oConf(X \Rightarrow Y) = |m(X) \cap m(Y)| \,/\, [|m(X) \cap m(Y)| + |n(X) \cap n(\text{-}Y)|] \,. \tag{4}$$

The difference between optimistic and pessimistic estimations for rules can be high. It would be useful to be able to predict values of support and confidence close to real (though unknown) ones. There were proposed several definitions of expected support and expected confidence in the literature. One can argue which definition is better or when should be applied. Whatever is the definition of a most likely support and confidence in incomplete dataset, the properties listed in Fig. 3 must be preserved:

The required set of properties
$sup(X) \in [pSup(X), oSup(X)]$
$sup(X) \geq sup(Y)$ for $X \subset Y$
$conf(X \Rightarrow Y) = sup(X \cup Y) \,/\, sup(X)$
$conf(X \Rightarrow Y) \in [pConf(X \Rightarrow Y), oConf(X \Rightarrow Y)]$
$\Sigma_{X \in Instances(A)}, sup(X) = 1$

Fig. 3. The required set of properties hold by a legitimate approach to data incompleteness. *X* and *Y* are itemsets, $A \subseteq AT$, and *Instances*(*A*) is the set of all itemsets that can be built for *A*

In the next section we introduce the concepts of a *probable support* and *probable confidence* that hold all the properties above.

4 Probabilistic Approach to Incompleteness

Let μ_t^a: $V_a \rightarrow [0,1]$ denote an attribute distribution for a tuple $t \in$ D and an attribute $a \in AT$. If μ_t^a is the same for all tuples, we denote it by μ^a. In particular, one can assume that $\mu^a(v)$ corresponds to the frequency with which value v occurs for the attribute a in the database D decreased by the number of tuples for which a is unknown (i.e. $\mu^a(v) = |n(a,v)| / |\mathrm{D} - d(a,v)|$). Another common assumption is to partition tuples into classes wrt. some criterion and fix μ_t^a for each class of tuples separately. So, tuples within a class will have the same attribute value distribution, whereas tuples belonging to different classes may have different distributions.

Based on the notion of μ_t^a, we define probability $probSup_t$ with which a tuple $t \in$ D supports an item (a,v):

$$probSup_t(a,v) = \begin{cases} 1 & \text{if } a(x) = v \\ \mu_t^a(v) & \text{if } a(x) = * \\ 0 & \text{otherwise.} \end{cases} \quad \textbf{(5)}$$

Now, we can define probability ($probSup_t$) with which a tuple $t \in$ D supports an itemset $X = \{(a_1,v_1),..,(a_k,v_k)\}$:

$$probSup_t(X) = probSup_t(a_1,v_1) * \ldots * probSup_t(a_k,v_k)\ . \quad \textbf{(6)}$$

Probable support (*probSup*) of an itemset X in the database D is defined as follows:

$$probSup(X) = [\Sigma_{t \in \mathrm{D}}, probSup_t(X)] / |\mathrm{D}|\ . \quad (7)$$

Property 1. Let X be an itemset in D, t be a tuple in D and $A \subseteq AT$. Then,
1. If $t \in n(X)$ then $probSup_t(X) = 1$,
2. If $t \in d(X)$ then $probSup_t(X) \in [0, 1]$,
3. If $t \in n(-X)$ then $probSup_t(X) = 0$,
4. $probSup(X) = [|n(X)| + \Sigma_{t \in d(X)}, probSup_t(X)] / |\mathrm{D}|$,
5. $probSup(X) \leq [|n(X)| + |d(X)|] / |\mathrm{D}|$,
6. $probSup(X) \in [pSup(X), oSup(X)]$,
7. $probSup_t(X) \geq probSup_t(Y)$ for $X \subset Y$,
8. $probSup(X) \geq probSup(Y)$ for $X \subset Y$,
9. $\Sigma_{X \in Instances(A)}, sup_t(X) = 1$,
10. $\Sigma_{X \in Instances(A)}, sup(X) = 1$.

Probable confidence (*probConf*) of a rule $X \Rightarrow Y$ is defined in usual way:

$$probConf(X \Rightarrow Y) = probSup(X \cup Y) / probSup(X)\ . \quad \textbf{(8)}$$

Property 2. Let $X \Rightarrow Y$ be a rule in D. $probConf(X \Rightarrow Y) \in [pConf(X \Rightarrow Y), oConf(X \Rightarrow Y)]$.
Proof: Let us assume that there are k tuples which match both X and Y in reality, but this fact is not derivable from D (i.e. these tuples are subset of $d(X \cup Y)$). Let l be the number of tuples that in reality match X, but this fact is not derivable from D (i.e. these tuples are subset of $d(X)$). Then the confidence of the rule $X \Rightarrow Y$ can be expressed as a function $conf(k,l) = (|n(X \cup Y)| + k) / (|n(X)| + l)$, where a) $k \in [0, |d(X \cup Y)|]$

and b) $l \in [0, |d(X)|]$. Clearly, the value $conf(k,l) \in [pConf(X \Rightarrow Y), oConf(X \Rightarrow Y)]$ for any values of k and l satisfying a) and b), respectively.

We will prove that $probConf(X \Rightarrow Y) \in [pConf(X \Rightarrow Y), oConf(X \Rightarrow Y)]$ by showing that $probConf(X \Rightarrow Y)$ is equal to $conf(k,l)$ for some values of k and l satisfying conditions a) and b).

Property 1.4 allows us to write the following equation: $probConf(X \Rightarrow Y) = [|n(X \cup Y)| + \Sigma_{t \in d(X \cup Y)},\, probSup_t(X \cup Y)] \;/\; [|n(X)| + \Sigma_{t \in d(X)},\, probSup_t(X)]$. Let $k_0 = \Sigma_{t \in d(X \cup Y)},\, probSup_t(X \cup Y)$ and $l_0 = \Sigma_{t \in d(X)},\, probSup_t(X)$. Then, $probConf(X \Rightarrow Y) = [|n(X \cup Y)| + k_0] / [|n(X)| + l_0] = conf(k_0, l_0)$. Applying Property 1.2 one can also observe that $k_0 \in [0, |d(X \cup Y)|]$ and $l_0 \in [0, |d(X)|]$, which means that k_0 and l_0 satisfy conditions a) and b), respectively. Hence, $probConf(X \Rightarrow Y) \in [pConf(X \Rightarrow Y), oConf(X \Rightarrow Y)]$. □

5 Review of Selected Approaches to Incompleteness

In this section we review three other approaches to incompleteness, namely:

- Ignoring missing values,
- Computing expected support and confidence based on certain information only,
- Applying only some complete (called *valid*) parts of the database.

The simplest approach to incompleteness in a relational database is just to ignore the presence of unknown values. In this case, the support (*iSup*) of an itemset X in D is equal to the pessimistic support $pSup(X)$ and the confidence (*iConf*) of a rule $X \Rightarrow Y$ is equal to $pSup(X \cup Y) / pSup(X)$.

The next approach was introduced in [3]. *Expected support* denoted by $eSup(X)$ for an itemset X was defined as $pSup(X) / [pSup(X) + pSup(\text{-}X)]$. *Expected confidence* (*eConf*) of a rule $X \Rightarrow Y$ was defined as $eSup(X \cup Y) / eSup(X)$.

The approach based on valid databases to missing values was presented in [5] and is based on the notions of *disabled tuples* and *valid databases*. A tuple $t \in$ D is *disabled* for X in D, if it contains missing values for at least one item in X (i.e. $\exists(a,v) \in X$: $a(t)=*\}$. $Dis(X)$ denotes the subset of D disabled for X. The *valid database* $vdb(X)$ for an itemset X is defined as D \ $Dis(X)$. Support (*vSup*) of an itemset X is computed in $vdb(X)$ and is defined as $|n(X)| / |vdb(X)|$. Confidence (*vConf*) of a rule $X \Rightarrow Y$ is computed in $vdb(X \cup Y)$ and is defined as $|n(X \cup Y)| / [|n(X)| - (|n(X)| \cap |Dis(Y)|)]$.

Figures 4-5 illustrate how values of support and confidence differ for all approaches discussed in this paper.

Itemset *X*	*pSup*, *iSup*	*oSup*	*probSup*	*eSup*	*vSup*
{(X1,a)}	3/8	5/8	3/6	3/6	3/6
{(X4,c)}	3/8	7/8	3/4	3/4	3/4
{(X1,a),(X4,c)}	2/8	4/8	13/32	2/6	2/2
{(X1,a),(X2,b)}	2/8	3/8	5/16	2/7	2/6
{(X2,b),(X4,c)}	2/8	3/8	11/32	2/7	2/4
{(X1,a),(X2,b),(X4,c)}	2/8	2/8	2/8	2/8	2/2

Fig. 4. Supports of exemplary itemsets in D from Fig. 1

$X \Rightarrow Y$	*pConf*	*oConf*	*probConf*	*iConf*	*eConf*	*vConf*
{(X1,a)} ⇒ {(X4,c)}	2/3	4/4	13/16	2/3	2/3	2/2
{(X4,c)} ⇒ {(X1,a)}	2/6	4/4	13/24	2/3	4/9	2/2
{(X1,a),(X2,b)} ⇒ {(X4,c)}	2/3	2/2	4/5	2/2	7/8	2/2
{(X2,b),(X4,c)} ⇒ {(X1,a)}	2/3	2/2	8/11	2/2	7/8	2/2

Fig. 5. Confidences of exemplary rules in D from Fig. 1

Fig. 6 shows which properties of a legitimate solution to the data incompleteness problem are held by approaches presented in the paper.

Required property \ Aproach	Probabilistic	Ignoring *	Expected	Based on *vdb*
$sup(X) \in [pSup(X), oSup(X)]$	+	+	+	-
$sup(X) \geq sup(Y)$ for $X \subset Y$	+	+	+	-
$Conf(X \Rightarrow Y) = sup(X \cup Y) / sup(X)$	+	+	+	-
$Conf(X \Rightarrow Y) \in [pConf(X \Rightarrow Y), oConf(X \Rightarrow Y)]$	+	+	+	-
$\Sigma_{X \in Instances(A)}, sup(X) = 1$	+	-	-	+

Fig. 6. Comparison of four approaches to incompleteness wrt. the set of required properties. "+" – indicates that the property is held; "-" – indicates the opposite

Among the four presented approaches only the probabilistic approach is valid.

6 Conclusions

In the paper we specified a notion of a legitimate approach to data incompleteness by listing a set of simple properties that should be held. We presented in detail a probabilistic approach based on attribute distribution and reviewed briefly three other approaches to incompleteness, namely: an approach that ignores the presence of missing values, the approach introduced in [3], and the approach based on valid databases. We concluded that only the probabilistic approach is legitimate.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Associations Rules between Sets of Items in Large Databases. In: Proc. of the ACM SIGMOD Conference on Management of Data. Washington, D.C. (1993) 207-216
2. Kononenko, I. , Bratko, I. , Roskar, E.: Experiments in Automatic Learning of Medical Diagnostic Rules. Technical Report. Jozef Stefan Institute, Ljubljana, Yugoslavia (1984)
3. Kryszkiewicz, M.: Association Rules in Incomplete Databases: In: Lecture Notes in Artificial Intelligence 1574, Methodologies for Knowledge Discovery and Data Mining, Proc. of Third Pacific-Asia Conference, PAKDD '99, Beijing, China, 26-28 April 1999, Springer (1999) 84-93
4. Quinlan J.R.: Induction of Decision Trees. In: Shavlik J. W., Dietterich T.G. (eds.): Readings in Machine Learning. Morgan Kaufmann Publishers (1990) 57-69
5. Ragel A., Cremilleux B.: Treatment of Missing Values for Association Rules. In: Proc. of Second Pacific Asia Conference, PAKDD '98. Melbourne, Australia. LNAI 1394. Research and Development in Knowledge Discovery and Data Mining. Springer (1998) 258-270

Discovering Sequential Patterns from Non-uniform Databases
(Extended Abstract)

Du Dang and X. Sean Wang

ISE Department, George Mason University
Fairfax, VA 22030, USA
xywang@gmu.edu
http://www.ise.gmu.edu/~xywang

Abstract. Databases, when used to collect data in the dynamic world, change over time. As a consequence, different parts of a database that come into the database at different times may contain activities with different characteristics. In other words, the data in the database are "non-uniform". This non-uniformity of the database gives rise to the idea of a Divide and Conquer strategy that divides a very large and non-uniform database into component parts for better mining. Experiments conducted show that the method is effective under various situations.

1 Introduction

Most of the current mining methods treat a database as a single uniform block of data. In other words, these methods do not look into the characteristics of the data to find out whether there are disparities among these characteristics and then adapt the mining process to deal with these disparities.

Take the example of a grocery chain located in a dynamic environment where the population varies significantly over time, both in number and in social and ethnic structures. The grocery chain tries to discover the buying behaviors or patterns of its customers. Based on the past experience, a group of grocery items that is purchased together by more than 50% of the activities (or transactions) is considered as frequent and will be used as guideline for future replenishment supply. The mining process, operated on the whole database, provides the following statistics: the group of grocery items {salt, pepper, sugar} is procured together in more than 90% of the activities, whereas the group of grocery items {soy sauce, tofu, lomein} is purchased together in only 18% of the activities. Based on this finding, only the group of grocery items {salt, pepper, sugar} is frequent. However, if we look into the details of the activity structures, we find that about 20% of the activities are somewhat closely related, namely, only these 20% of activities include one or more items in the group {soy sauce, tofu, lomein}. (These related activities may originate from customers of a particular ethnic group.) Using this group of "related activities" as basis for computation, we find that about 90% of these activities include the group of grocery items {soy sauce, tofu, lomein}. So, this group of grocery items should also be considered as frequent and should be purchased together in future replenishment supplies. This interesting behavior may not be discovered if the database of customer activities is considered as uniform.

In order to discover some interesting behaviors or patterns, a database should be treated as "non-uniform", that is, at its different parts, there exist activities with different characteristics. Non-uniformity is not restricted to the example presented above. In our dynamic world, databases, that collect data on activities and behaviors, are supposed to change significantly over time. The changes relate not only to the number of activities that increases continuously, but also to the characteristics of these activities. As a consequence of these changes, different "parts" of a database would contain activities with different characteristics. Addressing the non-uniformity issue in databases is helpful in many respects.

1. As shown in the above example, recognizing the disparity of activities in various parts of a database would help discover some interesting behaviors or patterns not discoverable if considering a database as a uniform block.
2. By identifying the parts with different characteristics, we will be able to divide the database following the lines of these parts and mine each one separately. This is the idea behind our Divide and Conquer strategy.

The rest of this paper is organized as follows: After some definitions, we will present the characteristics of the non-uniform environment (section 2). We then discuss methods that are applied to non-uniform databases, especially the Divide and Conquer strategy (section 3). The effectiveness of these methods will be shown through some experiments (section 4). Section 5 discusses related work. This paper concludes with a Conclusion and Future Work section.

2 The Non-uniform Environment

An **event** is represented as $e = (a_1, a_2, ..., a_m, T)$ where each a_i $(i = 1, 2, ..., m)$ is an attribute value and T is a time stamp value. An event is often abbreviated as $(\bar{a}, T)$ where $\bar{a}$ represents $a_1, a_2, ..., a_m$. A finite set of events is called an **activity**. An event with no time stamp value is an **event type** (represented as $\bar{a}$). An **event sequence** is defined as $s = (\bar{a}_1, T_1), (\bar{a}_2, T_2), ..., (\bar{a}_n, T_n)$ where $T_i \leq T_{i+1}$ for $i = 1, ..., n-1$. A sequence of event types is called a **sequence type**. A sequence type is a **sequential pattern** if it satisfies a frequency requirement.

The **non-uniformity level** $h(a_1, a_2)$ between two activities a_1 and a_2 is the ratio of the number of distinct disparate event types to the total number of distinct event types in these activities. For two activities A and B that have i and j distinct event types respectively, and p and q are the number of distinct event types that can be found in only one of the activities (p is specific to A and q is specific to B), we have $h = (p+q)/(i+q) = (p+q)/(j+p)$.

The non-uniformity level $h(c_1, c_2)$ between two component parts[1] c_1 and c_2 is the minimum of the levels of non-uniformity between the pairs of activities within

[1] A component part is a subset of activities in the database.

these component parts, i.e., $h(c_1, c_2) = min[h(a_1, a_2)]$ where $a_1 \in c_1,\ \ a_2 \in c_2$. The non-uniformity level for the whole database is the average of the h's for respective pairs of component parts.

A **non-uniform database**, with minimal non-uniformity level h, is a database that can be divided into at least two component parts so that, for each pair of component parts, the non-uniformity level is at least h.

The **Database frequency** of a sequence type is obtained by dividing the number of activities that support the sequence type by the total number of "related activities" in the database. An activity *supports* a sequence type if the activity contains all the event types in the sequence type and these events are in the same order as the event types in the sequence type. For a given activity α_1, another activity α_2 is said to *relate* (with respect to $1 - h$) to α_1 if the non-uniformity level is more than h.

3 Divide and Conquer Strategy

The following observations, that will be substantiated later in this paper, are used to support the Divide and Conquer strategy: (1) The partition of a database into component parts does not affect the accuracy of the mining but only the efficiency of this process. (2) A sequence type is frequent in a database only when it is frequent in at least one of its component parts. (3) Time for mining a database, in some cases, can be reduced significantly by splitting the database into smaller component parts for separate processing. (4) In some cases, whole database mining process can be avoided by only updating the frequencies of some already discovered sequential patterns.

Assumptions and Methodology. The methodology underlying the Divide and Conquer strategy is to split a very large and non-uniform database into smaller and more manageable component parts in order to perform the mining process more efficiently. Under this strategy, after dividing the database, we do the mining for each component part separately to discover the sequence types that are frequent in those components. We then scan the rest of the database to see if the sequence type discovered for a given component part is frequent in the whole database. We call this step "updating". Because the updating step is usually much less time consuming than performing all the steps of the mining process, some saving in processing time can be expected under certain conditions.

Dividing the Database. We will see, in the experiment section, that the efficiency of Divide and Conquer depends on the non-uniformity level of the database. It also depends on the number of component parts that we use. There is usually an optimal number of component parts for each database. Above this number, savings in mining time either decrease significantly or even become negative (i.e., overhead is greater than the savings).

Mining the Components. We apply the Agrawal's apriori algorithm [2] to perform local mining with the modification that when frequency is calculated, only the related activities are considered (with respect to an input value $1 - h$).

The Updating Algorithm. This updating algorithm utilizes the frequent sequence types discovered in one component part (the current component part) and updates the frequency of these sequence types against the rest of the database (the old component part). It is designed based on the above-mentioned observations and consists of the following steps:

Step 1. Eliminates, in the current component part, all sequence types that are known to not exist in the rest of the database.

Step 2. Scan the rest of the database to check for the frequency of the maximal sequence types. For those that are frequent, eliminate all sequence types that are subsequences of these maximal sequence types. This is because we only want to discover maximal sequence types.

Step 3. Scan the rest of the database for one-item sequence types, two-item sequence types, and so forth. Any time a sequence type is found to be not frequent, eliminate all longer sequence types in the following steps that contain this sequence type.

Step 4. Scan the rest of the database as follows:

(1) A sequence type discovered in a component part d_i of a database has also been frequent in the rest d_j of the database. Since the sequence is now frequent both component parts of the database, it is frequent in the whole database. Indeed, we have that $f_j/F_j > k$ and $f_i/F_i > k$ implies $(f_j + f_i)/(F_j + F_i) > k$. In this case, there is no need to scan the rest of the database.

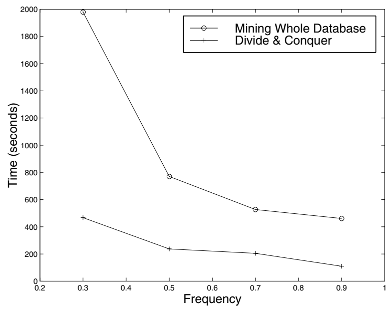
(2) A sequence type discovered in d_i is not frequent in the rest d_j of the database. In this case we have to scan all the rest d_j of the database and combine the frequencies. The sequence type is frequent if $(f_j + f_i)/(F_j + F_i) > k$.

4 Experiments

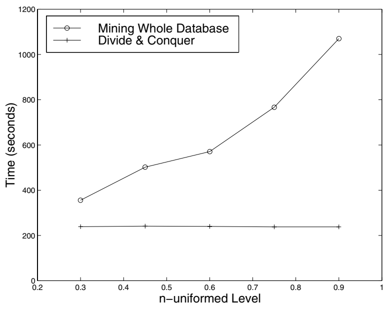
The experiments we performed include the creation of a number of data sets and the running of local mining and updating algorithms on these different data sets. We also run the algorithm to divide the database into component parts. These algorithms are designed using Visual C++ software and the executable codes run on a Pentium 200 MHZ micro computer.

We generate databases with pre-determined characteristics. The inputs to this generating algorithm are the number of distinct event types per component part, the non-uniformity level, the number of event types per activity, the number of activities per component part, and the number of component parts.

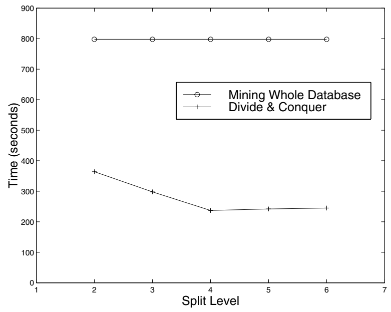
Using the data sets generated through the above procedure, we apply our Divide and Conquer algorithm to different scenarios such as mining with different frequencies, different non-uniformity levels, different split levels and different database sizes. We run 4 experiments. For each experiment, we randomly generate 3 or more databases, subject each database to the same algorithms and parameters and then average the results. The use of more than one database aims at eliminating some results that might be obtained by chance rather than through some given method. For each experiment, we compare time to mine the whole database to mining time using Divide and Conquer.



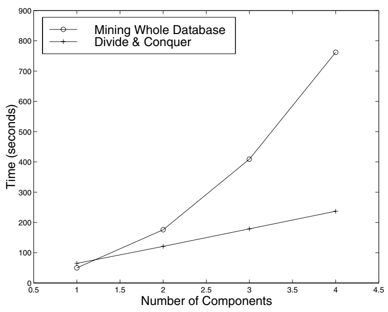
(a)



(b)



(c)



(d)

In the first experiment, we vary the frequency thresholds. See figure (a). (In the figures, when other parameters are not mentioned, we used the following: split level=4, uniformity level=75%, size of each component=500 activities, frequency threshold=0.5.) Even mining time decreases when frequency increases, the percentage of savings by using Divide and Conquer is still about 70%. In the second experiment, we vary the non-uniformity levels. See figure (b). As this level increases, the percentage savings due to Divide and Conquer also increases (from 32% at uniformity level 0.30 to 78% at uniformity level 0.90). In the third experiment, we fix the database size but try to observe data mining time when the number of component parts we split the database into changes. See figure (c). As expected, there is an optimal number of component parts. In our last experiment, we change database sizes in terms of the number of non-uniform component parts, while each component part has a fixed size. See figure (d). Because adding new non-uniform component parts affects the non-uniform level, time savings due to Divide and Conquer also increases.

The experiment results can be summarized as follows: (1) A non-uniform database can be efficiently split into smaller component parts for separate processing; (2) Savings increase with the non-uniformity levels; (3) No savings can be expected from splitting a uniform database due to updating overhead; (4) An efficient updating method is needed to support the Divide and Conquer strategy.

5 Related Work

To deal with very large databases that continue to expand over time, researchers have devoted efforts to devise efficient mining techniques aiming at reducing processing time. Data mining is an active field of research. Here we only relate our work to three papers. The first one is Agrawal and Srikrant [2], which designed methods for mining sequential patterns over large databases of customer activities. Mannila et al. [3] analyzed frequent episodes, that were defined as collections of events in a given partial order that occurred within a time interval of a given size.

The afore-mentioned data mining methods differ from the techniques proposed in this paper in that the referenced methods were used for mining a database in its entirety. This paper, on the contrary, emphasizes the differences in the database. Splitting databases into their component parts for more efficient processing is the focal point of the Divide and Conquer strategy.

Recently, Aggarwal and Yu [1] used correlation between items to replace the absolute support as recommended by [2]. The correlation notion is also used in our paper to define how two activities are considered as "related". However, the definition of this correlation as well as its application are quite different.

6 Conclusion and Future Work

In this paper, we have raised the observation that many databases are non-uniform, that is they contain activities with disparate characteristics. By identifying these characteristics, especially by grouping activities with similar characteristics in separate parts of a database, we can discover new and interesting patterns, and also apply methods, such as Divide and Conquer and incremental data mining, to improve mining efficiency. The idea has also been used for incrementally discover patterns in a large database.

In this paper, we considered the sequence mining problem. It is conceivable however, many mining problems may benefit the Divide and Conquer strategy. It will be interesting to study these different mining problems. Also, in the paper, we used Agrawal's apriori algorithm for local mining. It is clear that different algorithms may be used for different component parts to achieve the most benefit.

References

1. Charu C. Aggarwal and Philip S. Yu, "A new framework for itemset generation," PODS, 1998, pp. 18-24. 144
2. Rakesh Agrawal and Ramakrishnan Srikant, "Mining Sequential Patterns," IEEE Eleventh International Conference on Data Engineering, IEEE Computer Society Press, 1995. 141, 144
3. Heikki Mannila, Hannu Toivonen and A.Inkeri Verkamo, "Discovering Frequent Episodes in Sequences," First Conference on Knowledge Discovery and Data Mining, Montreal, California, August 1995. 144

An Effective Approach to Mining Exception Class Association Rules[1]

Fang Yu and Wen Jin

Department of Computer Science
Fudan University
200433, Shanghai, P.R.China
{fyu, jinwen}@fudan.edu

Abstract *Class association rule (CAR)*, which integrates the techniques of classification and association, is of great interest recently. However, it has two drawbacks: one is that a large part of *CAR*s are spurious; the other one is that some important *exception class association rule (ECAR)*s are difficult to find. In this paper, we propose a new method to find CAR, remove the *spurious class association rule (SCAR)*s, and generate ECARs effectively. According to our approach, user can retrieve useful information from *useful class association rule (UCAR)* and know the influence from different conditions by checking corresponding ECARs. Experimental results demonstrate the effectiveness of our proposed approach.

1 Introduction

C*lass association rule (CAR)* was first proposed by Bing Liu, Wynee Hsu, and Yiming Ma in 1998 [1]. It is a small subset of association rules whose right hand sides are restricted to the class label. For example, when we are doing experiments on the mushroom dataset, we have the rule as follows: {beautiful mushroom => poisonous}. *CAR* can also be used to generate a classifier [1].

Although CAR is very useful, a lot of *spurious rules* will appear [2]. The following is an example. Consider a report of breakfast cereal that surveys 5000 students' activities in the morning. The data shows that 3000 students play basketball, 3750 eat cereal, 3750 drink milk, 2750 both drink milk and eat cereal, 2000 students both drink milk and play basketball, 2000 students both play basketball and eat cereal, and 1000 students do all the three things. For a minimum support of 40% and minimum confidence of 50%, we find the following two class-association rules:

- Play basketball => eat cereal,
- Drink milk, play basketball =>eat cereal.

The first class-association rule is a *spurious rule* because the overall percentage of students eating cereal is 75%, which is even larger than the confidence of the rule that

[1] This project was funded in part by "Chun-Tsung scholar" Foundation, 973 Projects Foundation of China, and the National Doctoral Subject Foundation of China.

is 67%, meaning that being involved in playing basketball decreases the chance of being involved in eating cereal.

The second association rule is a *spurious rule* because the confidence of {Drink milk => eat cereal}, which is 73%, is higher than that of {Drink milk, play basketball =>eat cereal}, which is 50%. Thus, playing basketball has a negative effect on drinking milk and eating cereal.

Although CAR mining is able to find a lot of rules, some important rules are still not considered. They are referred to as *exception classification rule (ECAR)*s**,** which have high confidence but low support [3,4]. For example, if we set the parameter support to 50%, we have a rule {jobless => not granted} with support 70%, but we don't have another two important rules {jobless, women => granted} and {jobless, relative of clerk => granted} with support 30%. These rules are interesting because they are contrary to the rules generated by the association-rule mining and may give us some extra knowledge.

There are some people working on finding exception rules. In the work by Balaji Padmanabhan and Alexander Tuzhilin [5], an initial set of beliefs is used. These beliefs need domain knowledge and are relatively hard to get. In the work by Huan Liu and Hongjung Lu [4], a deviation analysis approach is proposed to discover all rules. In their work, a separated step of deviation calculation is needed to focus on the interesting attributes. In general, it scans the database many times to generate the window.

If a CAR does not belong to either kind of spurious rules, we call them useful classification association rule (UCAR)s. In this paper, we propose a new approach to finding UCARs and correspond ECARs effectively. The most important contributions of our technique are:

- It can filter the spurious rules and generate UCAR only.
- It can generate ECARs for each UCAR and tell us when applying this rule and what may affect the UCAR's result.
- It can provide an easy way to find a rule.

2 Problem Statement

In our work, we treat all the attributes uniformly and all the possible values are mapped to a set of consecutive positive integers. With these mappings, database can be treated as a set of (attributes, interger_value) pairs and a class label. Each (attributes, interger_value) pair is called *item*. The aggregation of some items is called *itemset*.

Definition 1. ***Class association rule (CAR)*** Let D be the dataset, I be the set of all items in D, and Y be the set of class labels. A *CAR* is an implication of form X=>y, where $X \subseteq I$ and y Y. A rule X=>y holds in D with confidence c if c% of cases in D that contain X is labeled with class y. The rule X=>y has support s in D if s% of the cases in D contain both X and y.

Definition 2. ***Spurious class association rule (SCAR)*** *SCAR* {X=>y} has two forms. One is support (y) > confidence {X=>y}. The other one is that if $X' \subseteq X$, confidence {X'=>y} > confidence {X=>y}.

Definition 3. ***Useful class association rule (UCAR)*** If a *CAR* does not belong to either kind of SCARs, it is called *Useful class association rule (UCAR)***.**

Definition 4. ***Exception class association rule (ECAR)*** Let $X \Rightarrow y_1$ be a UCAR, an ECAR has a form of $\{X, Z \Rightarrow y_2\}$, $X, Z \in I$, $X \cap Z = \varnothing$, $y_1, y_2 \in Y$, and $y_1 \neq y_2$. The confidence of ECAR satisfies the minimum confidence requirement, but does not satisfy the minimum support.

3 The Effective Approach to Mining ECAR

The process of discovering ECAR is divided into three steps as illustrated below.

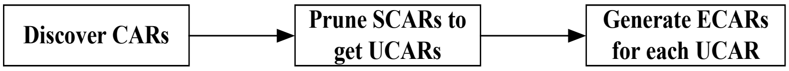


Fig. 1. The generation process of ECAR.

3.1 Discovery of classification association rule

In this section, we generate all of the *CARs*. We assume that all the possible values are mapped to a set of consecutively positive integers. The q classes are mapped to the first q number from 0 to q-1. CARs are divided into q groups from *CAR[0]* to *CAR[q-1]*. Each rule in the same group *CAR[y]* has the same class head y. We employ the framework of Apriori to generate CAR candidates, but only the itemsets in the same groups are joined.

Lemma: The candidate k+1 itemset of this group can be generated by joining the k itemsets of this group.

Prove: Let $\forall\ x_1x_2x_3...x_n \in \mathrm{CAR}_n[y]$ denote that the support of $x_1x_2x_3...x_ny$ is higher than the minimum support $s\%$. Then $x_1x_2x_3...x_{n-1}y$, $x_1x_2x_3...x_{n-2}x_ny$ must have a higher support than s%. They must be in the (n-1) frequent itemsets of $\mathrm{CAR}_{n-1}[y]$. Therefore $x_1x_2x_3...x_ny$ can be generated by joining within the same class group.

3.2 Prune the spurious rule

In this section, we use a forest structure to store the frequent itemsets. The forest has q trees: *tree[1],tree[2],…,tree[q]*. Nodes in each tree are frequent itemsets with class i. The confidence of a specific node and the maximum confidence of its antecedence and decedents are all stored in one node.

Suppose a node has k items in an itemset $\{<a_1, v_1>,<a_2, v_2>, \ldots, <a_k, v_k>\}$, then it has k pointers linking to its parent nodes$\{<a_2, v_2>, \ldots, <a_k, v_k>\}$, $\{<a_1, v_1>,<a_3, v_3>, \ldots, <a_k, v_k>\}$, …, $\{<a_1, v_1>,<a_2, v_2>, \ldots, <a_{k-1}, v_{k-1}>\}$. There is another set of pointers linking to its children. Figure 2 shows one tree structure.

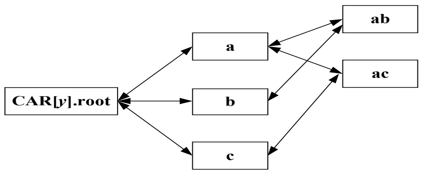


Fig. 2. The tree model for pruning SCAR.

The first kind of SCAR, where support (y) is larger than confidence {X => y}, is easy to be identified. Given q class, the support of class i in each tree CAR[i] can be stored. The confidence of each node is computed while scanning the tree once and is compared to the support of y. If it is smaller than the support of y, we prune this node.

The second kind of spurious rule is $X' \in X$ and confidence {X'=>y} > confidence {X=>y}. It is somewhat difficult to be deleted because all the possible confidences of {X'=>y| $X' \in X$} need to know. There are two steps to delete them.

1. In the first step, tree CAR [y] is traversed once by the Deep First Scan algorithm. Suppose an n's itemset is X, then n's max_antecedence is maximum confidence of {X'=>y | $X' \in X$}. This can be calculated in a recursive way for the maximum confidence of {X'=>y | $X' \in X$} = max {confidence {X'=>y}, max confidence of {X' | $X' \in X$, |X|=|X'|+1}}.
2. The second step is to calculate the maximum decedent's confidence and delete the second kind of spurious rule. Maximum decedent's confidence is calculated in a similar way as the max_antecedence. Each node's confidence is compared to its max_antecedence to determine whether it is a second kind of spurious rule. If so, we first mark this rule as spurious. Finally, if the n's confidence is higher than any of its decedents, all its subtrees are deleted because all of them are spurious rules.

3.3 Generate exception rule

In this section we find the ECAR. As described above, we have the CAR and have a small set of UCAR after pruning the SCAR. Suppose there is a UCAR {*X=>y}*, then the question is how to derive the exception rule {*X, Z=>y'}*. We have two phrases to generate the ECARs.

Phrase 1. Windowing
In this phase, if we have a UCAR *X=>y,* we get all the instances that contain X and y to generate a window.

Phrase 2. Rule finding

a. Find longest itemset in the window satisfying $\text{sup}_{\text{min.}}$
b. Search for reference rule {*Z*=>*y*'}
c. For each pair found, we have an except candidate: *X,Z->y'.*
d. For all the exception candidates, we scan the whole data once to check their confidence requirements. For those that satisfy minimum confidence, they are exception rules.

Although some of our work is based on [4], we make some important improvements. One is that deviation analysis is not needed. We can avoid setting the parameter of deviation threshold and still find interesting attributes. Another one is that we find all the exception rule with the same strong pattern A(UCAR{X=>y}), (see reference 4 for definition of A). In this way, the relationship between UCAR and ECAR is established. When applying UCAR, user can see what may affect the result of UCAR by looking at its corresponding ECARs.

4 Application

To demonstrate the effectiveness of our proposed approach, it was tested using several datasets. A lot of interesting exception rules were generated. We show the results based on the mushroom and credit dataset from the UCI repository.

4.1 CARgen

As we only want to generate frequent itemset with class label, we join frequent *k* itemset within the same class to generate candidate *k+1* frequent itemset. A lot of time can be saved. Following is the time usage of our CARgen method and Apriori with different minimum support when applied to the mushroom data set.

Table 1. Time comparison of CARgen and Apriori used for CAR generation

Support	Generate all the frequent itemsets (Apriori)	Generate frequent itemsets with class only
50%	6	2
40%	16	4
30%	51	14
20%	717	144
10%	28695	2236

4.2 SCAR prune method

In our experiment a lot of SCAR are pruned. Following chart shows the number of SCAR pruned compared to the original CAR.

Table 2. SCAR prune ratio

Support	Number of CAR	Number of SCAR	Number of UCAR	Prune ratio
40%	16	9	7	56.25%
30%	686	618	68	90.08%
20%	16364	16812	203	98.88%
10%	192036	191293	743	99.61%

4.3 ECAR generation

In our experiment, we find a lot of interesting rules. For example, from the mushroom dataset, a UCAR can be derived as follows:

{*(gill_zie) = [b], (stalk-surface-above_ring)= [s]*} => {*(type) = [e]*}.

Its confidence is 93.986% and its support is 41.55%. When user has this rule, one may be satisfied with its support and confidence, and may make a conclusion that whenever *(gill_size) = [b]* and *(stalk-surface-above_ring) =[s]*, mushroom must be edible. But unfortunately it is false. Described below is an ECAR:

{*(gill_size) = [b], (stalk-surface-above_ring)= [s],(cap_supface)=[s] , (stalk_root)=[b]*} => {*(type) = [p]*}.

This ECAR rule can be used together with the above UCAR rule. That is, before applying the above UCAR, we should check whether it satisfies ECAR above. If it does, it is classified as poisonous; otherwise, it is labeled as edible. In this way, the overall confidence of two rules is 98.99%. The deviation value of the UCAR above is –0.38, which is not very negative. Compared to the work in [4], if we set the parameter δ below –0.4, this important and interesting rule will not be found. These indicate that our proposed approach has big advantage.

5 Conclusion

This paper proposes a new approach to finding, removing the SCARs, and generating ECARs effectively. According to our scheme, user can retrieve useful information from UCAR and know the effect of different conditions by checking the corresponding ECARs. The experiment results demonstrated the effectiveness of our proposed approach. The approach we presented can find the rules, which are hard to be identified by others. In addition, it provides a good way for user to better understand the rule.

References

1. Liu, B., Hus, W., and Ma, Y.: Integrating Classification and Association Rule Mining. In the Proc. of the fourth International Conference on Knowledge Discovery and Data Mining New York city, (1998).
2. Aggarwal, C. C. and Yu, P. S.: A New Framework for Itemset Generation. In the Proc. of PODS 98, Seattle, (1998) 18-24.
3. Suzuki, E.: Autonomous Discovery of Reliable Exception Rules. In the Proc. of the International Conference on Knowledge Discovery and Data Mining, Oregon (1996).
4. Liu, H., Lu, H. J., Feng, L., and Hussan, F.: Efficient Search of Reliable Exceptions. In the Proc. of PAKDD99, Beijing, (1999) 104-203.
5. Padmanabhan, B. and Tuzhilin, A.: A Brief_driven Method for Discovering Unexpected Patterns. In the Proc. of the fourth International conference on Knowledge Discovery and Data Mining, New York city, (1998) 27-32.

Integrated Data Management and Enterprise Models

Alain Bazan[3], Florida Estrella[1], Zsolt Kovacs[1], Paul Lecoq[2], Jean-Marie Le Goff[2], Richard McClatchey[1], Steve Murray[3], Tony Solomonides[1], and Jean-Pierre Vialle[3]

[1]Centre for Complex Cooperative Systems, UWE
Frenchay, Bristol BS16 1QY UK
`Richard.McClatchey@cern.ch`
[2]CERN, Geneva, Switzerland
`Jean-Marie.Le.Goff@cern.ch`
[3]LAPP, IN2P3, Annecy-le-Vieux, France
`Steve.Murray@lapp.in2p3.fr`

Abstract. Enterprises need to cope with increasing volumes of complex and evolving data and at the same time to reduce 'time-to-market' for products. As data volumes increase and user communities grow and change with time, enterprise systems must be able to provide access to the enterprise data appropriate to multiple application viewpoints. In addition, the enterprise model must be flexible, adaptable and secure and be designed to maximise reusability of code, to cope with distribution of the enterprise activities and to inter-operate with legacy systems. The era where business rules are buried deep within the application code is coming to an end. Today users themselves seek to dynamically change their business rules and they need systems which can adapt to their evolving business needs, meet their requirements and scale to large installations. This paper outlines how an enterprise model that integrates process and product data modelling has been constructed following a description-driven design approach for the management of large-scale scientific apparatus.

1 Introduction

The management of product information from creative design through to manufacture has become increasingly important in recent years. Traditionally design engineers have employed Product Data Management (PDM) systems to control access to documented versions of product designs. In production management commercial Workflow Management Systems (WfMS) are beginning to be employed for coordinating the more complex and repeatable work processes of production. The integration of PDM with WfMS through a common data model would provide a platform in which the data and processes inherent in production planning and design can be more easily controlled. Large-scale engineering and scientific projects may, in addition, demand systems which require integration and/or distribution over many separate organisations. The integration of such 'islands of information' is heavily dependent on the flexibility and accessibility of the data model describing the enterprise's repository. The model must provide interoperability, extendibility and reusability so that a range of applications can access the enterprise data. Making the repository self-describing ensures that knowledge about the repository structure is available for applications to interrogate and to navigate around for the extraction of application-specific data. This paper examines a large-scale production environment which employs a self-describing data model to provide interoperability, reusability and integration between PDM and WfMS.

Product data management (PDM) tools, based on commercially available products, have been used for some time by manufacturing companies such as Mercedes-Benz to manage the data and documents accumulated in the design of products. PDMs [1] have been successfully employed to control the data and documents emerging from the creative and collaborative stages of product design when access to documents needs to be controlled between groups of designers. However their use in supporting the unstructured processes inherent in product development is somewhat limited [2]. PDM systems also provide few facilities for activity definition and no facilities for the enactment of production activities.

Workflow management systems (WfMS) [3] conversely, allow managers to coordinate and schedule the activities of organisations to optimise the flow of information between the resources of the organisation. Emerging commercial workflow systems seem to be appropriate tools for supporting the enactment of defined production operations. Such systems are, however, weak at handling the dynamic evolution of process and product definitions which occurs during design and also during the enactment of workflow processes.

Typically, in manufacturing systems, engineers use a PDM and production managers use Production Planning and Control Systems and/or WfMS software. Design control and production control are separated and there is little or no cross-talk between the two: the provision of continuity from design to production data is therefore a high priority. The integration of PDMs with WfMSs to provide consistency and continuity seems appropriate in next generation manufacturing systems. The PDM can then manage the definitions of the product and workflow data and the WfMS can cater for the instantiation, scheduling and enactment of those definitions. Agent software could potentially be used both to enact the workflow activities and to interact with the PDM definitions. Up to now the integration of product data and workflow worlds has only been proposed for the capture of design documentation in [4]. This paper outlines how PDM and WfMS can be integrated via a data model to facilitate the support of the full product development lifecycle in manufacturing from design to production.

The example used in this paper is that of the CRISTAL system which is being developed at CERN, Geneva to manage the data collected and the dynamic processes invoked in the construction and assembly of the Compact Muon Solenoid [5] experiment. CRISTAL is a distributed and integrated product data and process management system. It is based on an enterprise ontology and a middleware which integrates multiple enterprise product and process systems. It has an active object model which can be customised for use in manufacturing, telecommunications, healthcare, scientific and engineering domains. The active object model (or ontology) defines the objects, their states, the events and the conditions under which the objects change state. If the object model is changed, the system changes its behaviour. Consequently, suitably privileged people can customise the object model with minimal programming. CRISTAL provides flexibility in defining system configuration, in controlling device construction and supports the operation of the constructed device for applications. Further detail on CRISTAL can be found [6]. This paper opens by discussing the capture of product and task definitions and investigates the integration of PDM and WfMS technologies via a UML [7] model.

2 Product and Task Definitions

The development lifecycle of a large high-energy physics detector is much like any other large-scale construction activity in that it follows a design-prototype-implement cycle. It does differ from industrial production in that it is highly iterative and consequently dynamic in execution. Any changes in design need to be permeated through from conceptual design to physical construction as quickly as possible. The importance of rapidly reflecting design changes in production activities is typical of many examples of manufacturing engineering.

Conceptual design is a collaborative activity with designers checking-out and checking-in documents and diagrams of components under some policy of configuration management. The data vault aspects of a PDM lend themselves well to this creative design process. Product breakdown is often strictly hierarchical in form and attributes can be assigned to each part or sub-part. Objects in the product hierarchy can go through several stages of development so that "state" can be assigned to a part and can be managed by the PDM.

PDM systems provide a document change management service which can be used by engineering applications to assess, control and minimise the impact of material, product and process changes that occur in complex manufacturing lifecycles. According to [8] any PDM system used for the engineering of large-scale one-of-a-kind facilities (such as CMS) should hold the descriptions of both the PBS and the work breakdown structure (WBS). The PBS saves information pertaining to projects, sub-projects, documents, items etc. The WBS holds information about the organisation of tasks (or activities) to be performed and the resources required for each task. The WBS defines the activities which enable the engineers to build the production line and the production line can be viewed as a collection of (versioned) workflows. The WBS holds the definitions of the production line and can be mapped onto workflows (see following section).

The purpose of building a PBS for industrial applications in a PDM is to facilitate the capture of a design hierarchy of parts. As the product becomes more complex in structure, however, a parts explosion can take place in the PDM and data management becomes an issue. In CRISTAL, the CMS detector will be constructed from millions of individual parts, many of which will be identical in nature. It is simply infeasible (and, with current technology, practically impossible) to enter and manage these parts individually in the PDM. Instead, a concept of meta-data management could be followed where definitions of parts are captured in the PDM and instantiations of these definitions are used to form the PBS. In developing a meta-data concept to the management of production data an object modelling approach was used in CRISTAL.

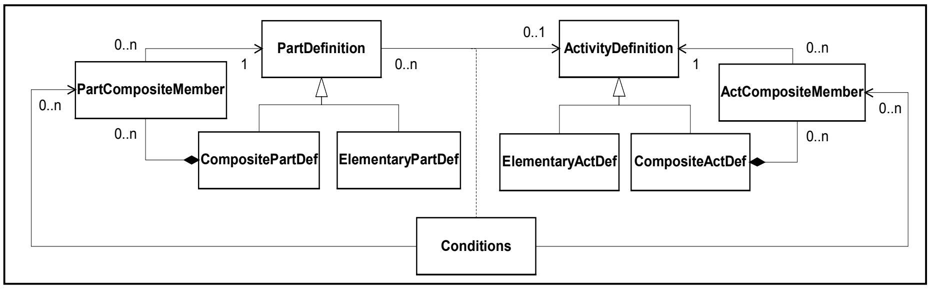


Figure 1: A subset of the CRISTAL Model showing meta-objects.

Figure 1 shows an abridged UML object model of the product breakdown structure which underlies CRISTAL. For data management the model is generalised into Elements and their definitions. Part definitions are one example of an Element definition and may be either elementary or composite in nature. Activity definitions (an instance of which may be performed on an instance of a part definition) are another example of an Element definition. Composite elements are made up of other elements and the CompositeElementMember object reflects membership of an element in a composite. In implementing this meta-data management the traditional hierarchical representation of assembly breakdown is transformed into a graph representation [9]. Individual parts (of the same type) are then associated with common definitions which eases data management.

The data model developed for CRISTAL has been designed to be generally applicable to production management environments. In fact the model is sufficiently generic in nature to describe many applications in which both data descriptions and activity descriptions (and their inter-relationships) are captured in a database and in which traceability is required of each execution of an activity on a part or product.

The structure of the CRISTAL data model is based on so-called directed acyclic graphs describing physical and catalogue aggregation as defined by [10]. However, the object model proposed in [10], although also based on directed acyclic graphs, does not provide semantics for the relationship between physical and catalogue aggregation nor does it explicitly capture the membership of one object in its aggregate, as in the CRISTAL model. The CRISTAL model is rich in semantics and, consequently, could be applied to general aggregation-based data management systems. The next section investigates how meta-objects as defined in CRISTAL also facilitate the integration of PDMs with WfMSs.

3 Integrating PDMs with Workflows

A WfMS is a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic. The main components of a workflow management system are a workflow application programming interface and a workflow enactment service. The workflow application programming interface (API) allows for the specification of workflows and workflow activities.

The workflow enactment service consists of an execution interface and an execution service provided by a so-called workflow engine. The engine is the component that executes the static workflow (production) descriptions which were defined through the programming interface. Commercial WfMS products normally subsume both the specification and enactment of workflows.

To achieve integration between the PDM and workflow components of CRISTAL, the PDM can be used to store a set of definitions of both the parts and the tasks that need to be executed on the parts. The workflow definitions are mapped onto the WBS and the workflow instances can therefore be derived from their definitions residing in the PDM. The PDM acts as the reference database both for the activation and enactment services of the production workflow system and manages (versions of) the PBS and the WBS.

Just as the PBS has been modelled using meta-objects, the workflow definitions (WBS) of CRISTAL can also be modelled. Figure 1 shows the overall data model. Each part definition has at most one workflow definition assigned to it. Workflows can be reused for several part definitions. A part definition is an item of the PBS, the workflow definition is the workflow meta-object (WBS). A workflow meta-object is an ordered collection of workflow activity definitions which will be executed on every part registered in the CRISTAL system with its corresponding part definition. Workflow (activity) definitions can be either elementary or composite in nature.

CRISTAL has two distinct functions: one of product data management and one of workflow management. To achieve integration between the PDM and workflow components of CRISTAL, the PDM can be used to store descriptions of parts and of activities that need to be executed on the parts. The PDM can then manage the definitions of the product and workflow data and the Workflow software can cater for the instantiation, scheduling and enactment of those definitions. The PDM acts as the database for the activation and enactment services of the production workflow and handles versioning of the definitions as the schema evolves. CRISTAL, consequently, controls and tracks parts through the manufacturing life cycle from a PDM-resident design to the Workflow-controlled construction and assembly of the CMS detector.

The CRISTAL data model has been designed so that each assignment of a Workflow Definition to a Part Definition is declared for a specific purpose. Each purpose has associated with it some of Production Conditions (see figure 1): the data model captures the definition the production conditions required for each assignment of a workflow definition to a part definition. In integrating PDM with WfMS, these are essentially the part objects, commands and locations required as prerequisites for the initiation of a workflow on a part. This method of integrating PDM and WfMS through the definition of meta-objects and their mutual assignment is very powerful. It allows other links to be made between aspects of the overall CRISTAL data model. For instance, the same mechanism can be used to assign agents to workflow definitions for the purposes of enactment or the assignment of agents to part definitions for the purposes of data management.

This technique can be generalised for other applications e.g, the association of a maintenance activity to a part will require quite different conditions to be captured than when the detector was constructed. Another example is calibration activities, parts and calibration-specific conditions.

In other words, the identified association between the process and part description worlds carries rich semantics. It allows many other links to be made between aspects of the overall CRISTAL data model: the same mechanism can be used to assign agents to activity definitions for the purposes of enactment or the assignment of agents to part definitions for the purposes of resource management.

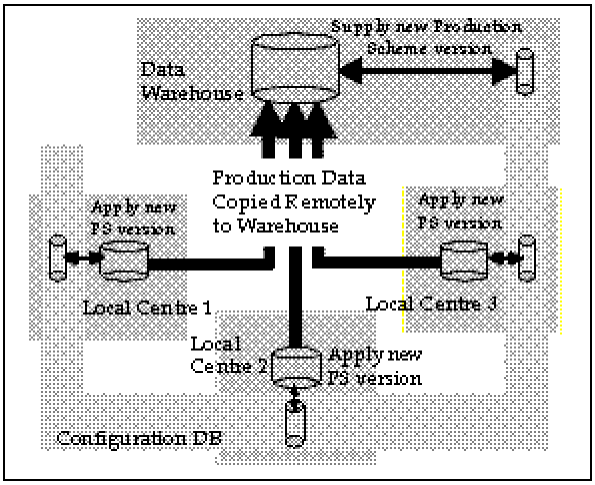


Figure2: The CRISTAL data warehouse and data distribution.

A CRISTAL system comprises one or more distributed data gathering centres (figure 2), each of which is federated into the system. These centres are a single Central System and Local Centres in which the CRISTAL software will run. The CRISTAL system uses a set of roles to define user access to its software and data. Each Local Centre will also have a set of Instruments defined in the database in terms of the commands that each instrument uses and the data formats expected as outcomes from the execution of activities by instruments [11].

4 Description Driven Systems and Enterprise Models

With increasing complexity in the organisation, data and functions of enterprises, information systems need to be increasingly flexible and extendible, simple to interrogate and to navigate around and to be interoperable with existing legacy systems. The integration of product and process models in a unified data model provides the means by which information can be shared across an enterprise throughout the system lifecycle from design to production. Large-scale engineering and scientific projects may, in addition, demand systems which require integration and/or distribution over many separate organisations. The integration of such 'islands of information', that will form the basis of so-called 'virtual enterprises', is heavily dependent on the flexibility and accessibility of the data model describing the enterprise's repository. Making the repository self-describing, based on descriptive structures, ensures that knowledge about the repository structure is available for applications to interrogate for the extraction of application-specific data.

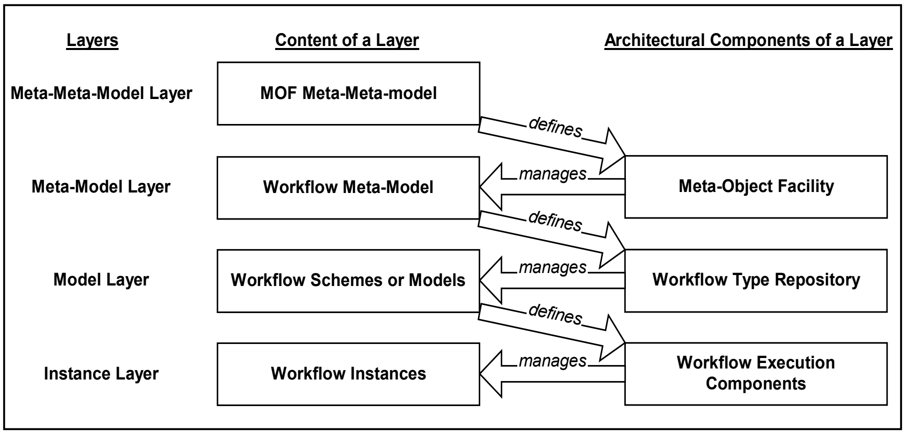


Figure 3. A 4-layer meta-modeling architecture.

One important aspect in providing for enterprise models concerns ways of making components and systems self-describing. To encourage flexibility and interoperability, systems should be designed to be able to retain knowledge about their dynamic structure and for this knowledge to be available to the rest of the distributed infrastructure through the way that the system is plugged together. There is no doubt that as the CMS construction process gets underway production schemes and definitions of production activities and parts will continue to change. Clearly, these changes in definition must be folded into all viewpoints derived from the construction data. To cope with this the production management system used must, ideally, be able to support dynamic self-reconfiguration. One step in achieving this is for the system to make a representation of itself available for manipulation. A system which can make modifications to itself by virtue of its own computation is called a reflective system [12]. In order to facilitate inter-operation with future systems and in order to adapt to reconfigurations and versions of itself large systems (such as those found in High Energy Physics) should become self-describing or reflective. The representation needed for self-description is often termed meta-data. In CRISTAL the concept of a meta-model is introduced to provide for interoperability, flexibility and to reduce system complexity.

'Description-driven systems' can be defined as systems in which the definition of a domain-specific configuration (such as healthcare) is captured in a computer-readable form. This definition can be interpreted by applications to achieve domain-specific goals. In a description-driven system definitions are separated and managed independently from instances. This allows the definitions to be specified and to evolve asynchronously from instantiations (and executions) of those definitions. Description-driven systems require computer-readable models both for definitions and for instances. One example of the use of description-driven systems is in a WfMS where the business process model defines the instantiated workflows and the definitions are managed separately from the instantiations. WfM systems are often built on a multi-layer architecture [13].

In WfM systems the workflow instances (such as activities or tasks) correspond to the lowest level of system abstraction - the instance layer (see Figure 3). In order to instantiate the workflow objects a workflow scheme is required. This scheme describes the workflow instances and corresponds to the next layer of abstraction - the model layer. In order for the workflow scheme itself to be built, a further model is required to store the semantics for the generation of the workflow scheme. This model (i.e. a model describing another model) is the next layer of system abstraction - the meta-model layer (see Figures 3). The meta-meta-model layer is the layer responsible for defining a general modelling language for specifying all meta-models. This top layer is the most abstract and must have the capability of modelling any meta-model.

Multi-layer systems provide reusability, complexity handling, schema evolution and facilitate interoperability The CRISTAL meta-model is comprised of so-called meta-objects each of which is defined for a class of significance in the data model: e.g part definitions for parts, activity definitions for activities, and agent definitions for agents. In the model information is stored at specification time for types of parts or part definitions and at assembly time for individual instantiations of part definitions. At the design stage of the project information is stored against the definition object and only when the project progresses is information stored on an individual part basis. This meta-object approach reduces system complexity by promoting object reuse and translating complex hierarchies of object instances into (directed acyclic) of object definitions graphs (see [10]). Meta-objects allow the capture of knowledge (about the object) alongside the objects themselves, enriching the model and facilitating self-description and data independence. It is believed that the use of meta-objects provides the flexibility needed to cope with their evolution over the extended timescales of CRISTAL production and the flexibility required to cope with ad-hoc activity specification.

CRISTAL systems can be used to track processes and products for a variety of purposes. The CRISTAL model is sufficiently generic in nature to describe any applications in which both product descriptions and process descriptions (and their inter-relationships) are captured in a database and in which traceability is required of each execution of a process on a product. For example, the association of a sales activity to a product would require sales-specific conditions, rather than, say, engineering-specific conditions, to be captured in the database. In other words the association between the process and product descriptions carries rich semantics and the model allows associations to be made between subsets of the overall data model. As an example, this mechanism can also be used to assign agents to process descriptions for the purposes of enactment or agents to part descriptions for the purposes of resource management. Consequently, the CRISTAL model is generally applicable to large-scale production management environments

Once information has been captured in a CRISTAL database it will be accessed by end-users for a variety of purposes, from a variety of viewpoints. The description-driven nature of the CRISTAL model facilitates the extraction of data for products via a mechanism which can navigate the model, can interpret its structures and can present the data in a form meaningful to the end-user. This extraction facility comprises a set of processes (or Agents) which can be invoked either by a domain-specific application or by a domain non-specific application. The agents either navigate the enterprise model to project out domain- or viewpoint-specific data or they navigate the model to correlate effects between viewpoints. In the viewpoint-specific case the agents perform the traversal of the enterprise description, following selected products and extract the relevant data for the application. In the inter-viewpoint case the agents are used to determine the effect of a system-wide change on individual viewpoints or sets of viewpoints.

In High Energy Physics (HEP), the environment in which CRISTAL was developed, the viewpoint-specific applications could include calibration systems, alignment systems for detectors, slow controls applications or indeed any physics analysis application which requires access to data captured during the construction of the detector. In the example of car production, used elsewhere in this paper, viewpoint-specific applications could include car assembly and maintenance and there are clear correlations between these viewpoints in terms of identified car parts. A further example of a viewpoint in car production is that of an engine management system. The car assembly viewpoint refers to engine parts for the purposes of assembly and testing, whereas the engine management viewpoint considers engine parts for the purposes of engine operation and its management.

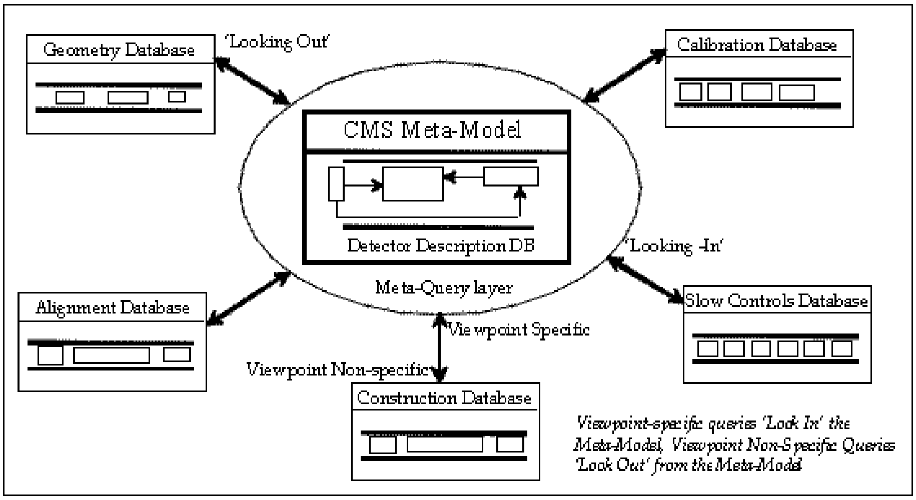


Figure 4: Viewpoints and the Enterprise Model

Figure 4 shows the architecture of a descriptive model-based system which encompasses multiple viewpoint databases (e.g Personnel, Products, Suppliers). In each case data has been extracted from a general descriptive model (or enterprise description database) via the extraction facility. This generalised extraction facility can navigate the description, from an application-specific viewpoint, looking for data associated with a set of defined elements.

The descriptive model of figure 4 therefore acts as a repository of knowledge against which queries can be issued to locate and extract data across multiple databases. Agent processes are used to 'look into' the descriptive model and to extract data from a user-specified viewpoint and to 'look out' from the model to correlate effects between viewpoints. The overall effect is to produce an integrated set of cooperating databases, or enterprise model, accessed through an extraction facility. Hence 'islands' of disparate information (such as maintenance, calibration, alignment) are eliminated. Such an approach could reasonably be applied to organisations developing technologies for 'virtual enterprises' where collections of autonomous databases could be related via a central enterprise model.

5 Other Related Research

Recently a considerable amount of interest has been generated in meta-object description languages. The OMG has produced the Meta Object Facility (MOF) [14] that manages meta-models which are relevant to the OMG Architecture. This meta-modelling approach will facilitate further integration between product data management and workflow management thereby providing consistency between design and production and speeding up the process of implementing design changes in a production system.

The experience of using meta models and meta objects at the analysis and design phase in the CRISTAL project has been very positive. Designing the meta model separately from the runtime model has allowed the design team to provide consistent solutions to dynamic change and versioning and to support data extraction via user-defined viewpoints. The object models are described using UML which itself can be described by the OMG Meta Object Facility and is the candidate choice by OMG for describing all business models.

The current phase of CRISTAL research aims to adopt an open architectural approach, based on a meta-model and a meta-query facility to produce an adaptable system capable of interoperating with future systems and of supporting views onto an engineering data warehouse. The meta-model approach to design reduces system complexity, provides model flexibility and can integrate multiple, potentially heterogeneous, databases into the enterprise-wide data warehouse. A first production system for CRISTAL based on CORBA [15], Java and Objectivity technologies has been deployed in the autumn of 1998 [16]. The second phase of research will culminate in the delivery of a production system in 2000 supporting meta-queries and the definition, capture and extraction of data according to user-defined viewpoints.

The concept of using meta-data to reduce complexity and aid navigability of data resident in a database is well known. Also its use in minimising the effect of schema evolution in object databases has been stated many times elsewhere. In the CRISTAL project meta-data are used for these purposes and, in addition, meta-models are used to provide self-description for data and to provide the mechanisms necessary for developing a meta-query facility to navigate multiple data models. Using meta-queries, data can be extracted from multiple databases and presented in user-defined viewpoints. The CMS meta-model of figure 4 therefore acts as a repository of knowledge against which meta-queries are issued to locate and extract data across multiple databases. Agent processes are used to ‘look in’ the meta-model and extract data from a user-specified viewpoint and to ‘look out’ from the model to correlate effects between viewpoints. The overall effect is to produce an integrated set of cooperating databases accessed through a meta-query facility. Hence ‘islands’ of disparate information (such as maintenance, calibration, alignment) are eliminated. Such an approach could reasonably be applied to organisations developing technologies for ‘virtual enterprises’ (such as in [17]) where collections of autonomous databases could be related via a central enterprise meta-model.

Acknowledgments

The authors take this opportunity to acknowledge the support of their home institutes. The support of P. Lecoq, J-L. Faure and M. Pimia is greatly appreciated. N. Baker, A. Bazan, T. Le Flour, S. Lieunard, L. Varga, G. Organtini and G. Chevenier are thanked for their assistance in developing the CRISTAL prototype.

References

1. Philpotts, M. (1996) An Introduction to the Concepts, Benefits and Terminology of Product Data Management, Industrial Management & Data Systems, Vol 4 pp11-17
2. Pikosz, P. and J. Malmqvist (1996) Possibilities and Limitations when Using PDM Systems to Support the Product Development Process, In proc of NordDesign’96, Helsinki, Finland.
3. Georgakopoulos, D., M. Hornick and A. Sheth (1995) An Overview of Workflow Management: from Process Modelling to Infrastructure for Automation, Journal of Distributed and Parallel Database Systems Vol 3 (2), pp119-153.
4. Ramanathan, J. (1996) Process Improvement and Data Management, IIE Solutions, 1996 28 12 pp24-27.
5. Technical Proposal. The CMS Collaboration (1995), ftp://cmsdoc.cern.ch/TPref/TP.html
6. McClatchey, R. et al. (1997) A Distributed Workflow and Product Data Management Application for the Construction of Large Scale Scientific Apparatus. NATO ASI Series F : Computer & Systems Sciences Vol 164.pp 18-34.
7. Fowler M. and K. Scott.(1997) UML Distilled - Applying the Standard Object Modelling Language. Addison-Wesley Longman publishers.
8. Bachi, G and A. Hameri (1995) What to be Implemented at the early Stages of a Large-Scale Project. CERN MT/95-02 (DI) LHC Note 315.
9. Baker, N. et al. (1998) An Object Model for Product and Workflow Data Management. In proc. of the 9th ACM Int Workshop & Conf on Database and Expert System Applications (DEXA’98) pp 731-738.
10. Blaha M. and W. Premerlani (1999) Object-Oriented Modelling and Design for Database Applications. Prentice Hall publishers.

11. Le Goff, J-M et al. (1998) Detector Construction Management and Quality Control: Establishing and Using a CRISTAL System. CERN CMS NOTE 1998/033.
12. Kerherve, B. and A. Gerbe (1997) Models for Metadata or Metamodels for Data ? In proc. of the 2nd IEEE MetaData conference, 1997.
13. Schulze. W., „Fitting the Workflow Management Facility into the Object Management Architecture“. Proc of the Business Object Workshop at OOPSLA'97.
14. OMG (1997 & 1999) Meta-Object Facility RFP TC Doc cf/96-02-01, Evaluation Report TC Doc ad/97-08-14 and ad/99-09-04
15. OMG (1992) The Common Object Request Broker: Architecture & Specifications, OMG Publications.
16. Bazan A. et al. (1999) The Use of Production Management Techniques in the Construction of Large Scale Physics Detectors. IEEE Trans on Nuclear Sci. 1999 Vol 46 No 3 pp 392-400 ISBN 0018-9499
17. Hardwick, M., et al. (1996) Sharing Manufacturing Information in Virtual Enterprises. Communications of the ACM Vol 39 (2) pp 46-54.

A Virtual Private Network for Virtual Enterprise Information Systems [*]

Lina Wang, Ge Yu, Guoren Wang, Xiaochun Yang , Dan Wang, Xiaomei Dong, Daling Wang, and Zhe Mei

School of Information Science and Engineering,
Northeastern University, Shenyang 110006, P.R. China
{Lnwang,Yuge,Wanggr}@mail.neu.edu.cn

Abstract The security requirements of a virtual enterprise information system are investigated and a virtual private network (VPN) is designed to ensure the security, taking the advantages of the integrated functionality of encryption, key management, user authentication, firewall, and certificate authority with low cost. Several novel techniques are adopted in the VPN: 1) An effective chaotic encryption algorithm by using Logistic chaotic mapping; 2) a chaos theory based user's "fingerprint" authentication algorithm by using mixed optically bistable chaotic model; 3) a reliable authentication mechanism by integrated Guillou-Quisquater digital signature with RSA encryption; 4) an efficient Certificate Authority (CA) and digit certificate mechanism. The details of the VPN are also presented, including a cryptography server, an auditing server, a digital signature server and an authenticating server.

1 Introduction

A virtual enterprise (VE) is a dynamic alliance organized by many partners for common benefits[1,2]. Usually, a virtual enterprise has a coordinator who is the initiator and is in charge of constructing the alliance and selecting participants. The information system is an important infrastructure for running a virtual enterprise, in which the heterogeneous, autonomous and distributed information are to be transferred, exchanged and shared on Internet in a secure way[3]. However, Internet lacks secure architecture because the underlying TCP/IP protocol is not secure, and the components like operating systems, application systems, router software hide many insecure flaws. The typical threats faced by VE information systems include:

- Tampering – unauthorized updating such as malicious alternation by hackers.
- Leakage – uncovering confidential information without proper protection, etc.
- Repudiation – refusing to acknowledge or accept an obligation, etc.
- Fraud – forging a document or signature, etc.
- Stealing – illegally utilizing the resource without any payment.

Therefore, the security requirements of VE information systems include:

[*] The work was supported by the Fok Ying Tung Education Foundation and the National 863/CIMS Research Grant No. 863-511-946-003.

- Access control – any access from outside of the VE and any access to the protected services should be validated against unauthorized users.
- Identification authentication – any user should be identified when s/he enters the system.
- Confidentiality – the sensitive information should be protected strictly against illegal users.
- Integrity – the transferred information should not be changed in the communication.
- Audit management – it is necessary to guarantee non-repudiation, and to trace illegal activities.

Computer security and cryptography have been widely researched and applied in the emerging applications [4,5], for example, the secure electronic transfer protocol (SET) is proposed to support network based trading of electronic commerce. VPN[6] is a new Internet technique that provides private network on open Internet, by placing security and routing functionality at the borders of the public network to ensure secure connections. They can deliver business-to-business communications as Extranet, and replace wide-area backbone network like Intranet and dial-in remote access networks. For those businesses not yet entirely moving their production networks to the Internet, a VPN is a cheaper alternative solution than adding more investment for dedicated circuits.

Until now, two generations of VPN products have been developed. The first generation products include Intranet VPN, Remote Access VPN and Extranet VPN. Since they have restrictions to applications, the second generation products are developed, such as the VPN Concentrator of 3COM. In this paper, a VPN dedicated for the VE information system is designed and implemented by adopting some novel approaches.

The rest of the paper is organized as follows. Section 2 introduces a VE information system -- ViaScope, which is taken as the research background, and discusses the security requirements and strategy. Section 3 presents the design of the VPN for the ViaScope, including the information encryption algorithm, the authentication algorithm, the key management, the Guillou-Quisquater protocol based identification authentication and digital signature, and Certificate Authority and digit certificate management. Section 4 describes the implementation issues of the security sever in ViaScope, including the cryptography server, the auditing server, the digital signature server and the authenticating server. Finally, Section 5 concludes the paper and gives the future works.

2 Security Requirement Analysis and Strategy for a VE Information System

2.1 Overview of ViaScope and its Security Requirement

ViaScope[7] developed at Northeastern University is a CORBA/IIOP based VE information system. ViaScope is designed with two goals: 1) information integration within a VE, and 2) information service for VE members. To implement the first goal, the data sources and their related common services in a VE environment are wrapped as CORBA objects to be registered into the VE repository, and a federated database is built. To implement the second goal, the information services are provided by the CORBA ORB mechanism. Fig. 1 shows the brief architecture of ViaScope.

A VE environment often involves many different kinds of database systems like Oracle and SQL server, operating systems like Unix and Windows, middleware systems like Orbix and OrBus, and application systems like C++ or Java or even legacy systems. There are many potential security problems at every boundary between these components. For example, considering the interface at ORB-to-ORB and at other levels of the system, the CORBA architecture attempts to provide

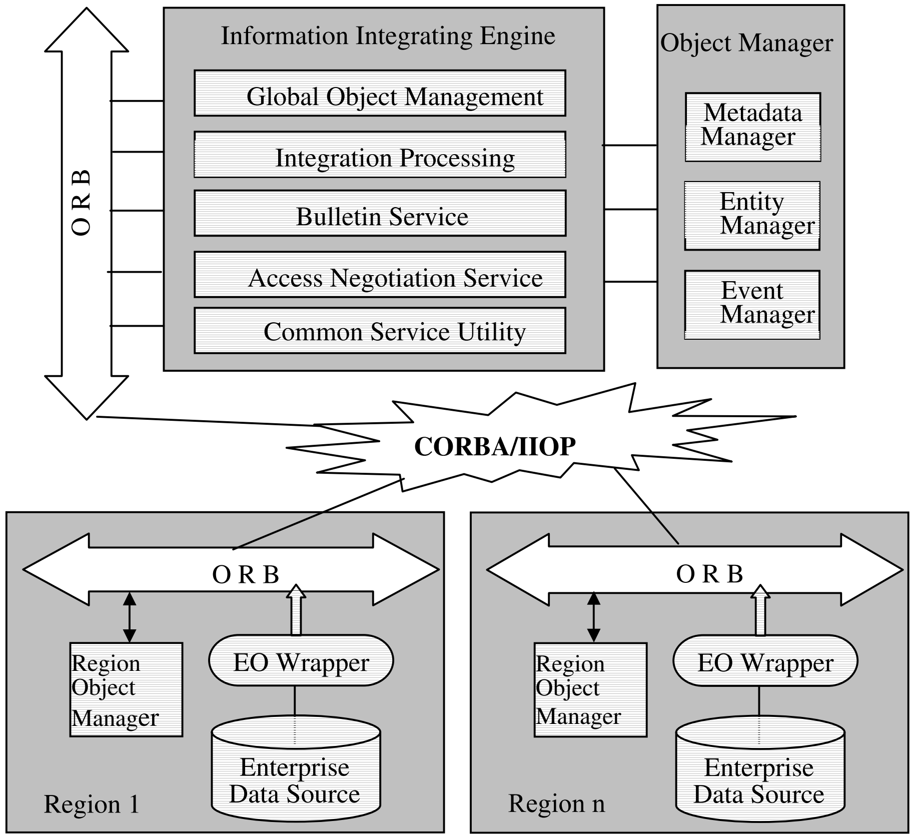


Figure 1 Architecture of ViaScope

transparent object location and activation, while the security policy may require a client to know that it is communicating with a trusted object.

According to the structure of ViaScope given in Fig.1, the security requirements are analyzed for different components as follows.

(1) CORBA ORB mechanism. It is the foundation to build a VE information system with its distributed object management facility. However, the ORB ability to activate objects transparently leads to a new form of the Trojan horse scenario. Because the ORB can activate objects automatically, it might be tricked into transparently activating a Trojan horse object instead of the intended object. Since CORBA has not standardized the implementation repository, it is hard to control the problem across multiple ORBs. Moreover, object references embedded in application programs also may lead to some security issues. For example, the existence of an embedded reference to an object can be hidden from a user. Therefore, how to detect and control these types of object references is to be solved. Many CORBA based application architectures utilize multiple levels of nested invocation (such as when a client calls an implementation that calls another implementation and so forth). It is necessary to ensure how and when security credentials should be delegated, and how servers that can work on behalf of many users are handled.

(2) Enterprise Object (EO) wrapper. It generates enterprise data objects and application service objects. The former are the wrapped data sources such as databases and files and can provide query interface for the data sources, and the latter are the wrapped applications such as transaction processing and can provide many kinds of services. It is necessary to guarantee the security for sharing, transferring and accessing these EOs on network. Therefore, the EO access control mechanism is needed to define access right, and the encryption mechanism is needed for secure communication.

(3) Global object management. Generally, trust and shared non-competitive objectives are the basis for forming VE. When an object interacts with another object, it usually expects that the other will confirm to some behavior. This expectation represents a form of trust between the objects. In situations where interaction will involve a high degree of uncertainty, the contract becomes uncertain in interactions between the objects. Therefore, the security policies should be added into the behaviors of the objects in terms of permissions, prohibitions and obligations. The permissions define the actions allowed to do, prohibitions define the actions not allowed to do, and the obligations define the action that must do.

(4) Enterprise bulletin service. VE-oriented information integration, management and service aim at for information sharing on Internet. Enterprise bulletin service is provided for the system or a member to release the information to all involved VE members efficiently. It is necessary to guarantee the trust of the bulletins information.

(5) Access negotiation service. One challenge of a VE is for a coordinator how to select correct partner or for a participant how to be selected by the best partner. It is necessary to guarantee the trust between the coordinator and the participant by defining proper accesses right and access level. At the same time, the autonomy of local systems should be kept when authorizing users.

(6) VE object management. It consists of metadata management, entity management, and event management. The CORBA interface repository and object broker services make the data available to other applications. These services might reveal information that should be restricted by security. The trade-off must be made between interoperability and security. Since the information contained in the VE

objects is fatal for the success of the VE, it is necessary to keep them secure by encrypting those critical objects.

2.2 Security Strategy of VE Information System

One of good choices for security solution of a VE information system is to construct a VPN, since the VPN can integrate the functionality of encryption, authentication and digital signature to implement a secrete tunnel on Internet to let legal VE members securely visit VE objects and services. In addition, most VEs are built among small and medium-size enterprises and a VPN is a wise selection for them due to its low investment requirement. The following are the major security strategy to be adopted in the VPN.

(1) Confidentiality. It ensures that the information such as an enterprise contract in the network transfer remains private. This can be accomplished through encryption to prevent illegal user to modify the contents of the contract. That is, before the data is sent to a receiver, the plaintext of the data are firstly encrypted into the ciphertext, and then sent to the receiver. The ciphertext received by the receiver is decrypted to the original plaintext. Plaintexts may be customer's contracts, orders, digital money or other sensitive information like VE object information.

In a VE information system, a cryptography server is needed to support confidentiality of VE objects and services. Symmetric key (or private key) algorithm and asymmetric key (or public key) algorithm can be used in a integrated way.

(2) Integrity. It ensures that a message can not be modified during transmission. For example, when a contract is sent via communication network, the transfer must be protected from unauthorized alteration or destruction. To do it, one-way hash function may be used.

(3) Authentication. It ensures that the sender of a message is whom he claims to be. An authentication server is needed to provide for the user authentication and system authentication. For example, it is required for the authentication between a coordinator and a participant. A signature server is needed to provide digital signature to ensure the integrity and authentication of information.

In this paper, the issues of authentication and integrity are considered together. A sender makes the signature and encryption for the hash value by using his private key and the receiver's public key. In the receiver site the encrypted hash value is decrypted by using the receiver's private key and verified by using the sender's public key.

(4) Non-repudiation. It ensures that the originator of the message cannot deny that he sent the message. For example, whether a contract is received or verified. The authentication should ensure that contract indeed comes from the true trade partner. Digital signature is used to verify the authentication of the contract.

3 Design of Virtual Private Network

In the VE information system, a VPN is designed as shown in Fig. 2. The VPN consists of seven elements:

<VPN>::＝{<Encryption>, <Key-Management>, <User-Authentication>, < Digital Signature>,<Access Control>, <Certificate-Authority>, <Net-Management>}
<Encryption>::=<DES>|<3-DES>|<IDEA>|<Chaos based Encryption Algorithm>
<Key-Management> ::=<RSA based Internet Key Exchange>
<User-Authentication>::=< Chaos Theory based User's "Fingerprint" Authentication by Using Mixed Optically Bistable Chaos Model>|< Guillou-Quisquater Protocol based Identification Authentication>
< Digital Signature>::=< Guillou-Quisquater Protocol based Digital Signature with RSA Encryption >
<Access Control> ::= <Access Control Matrix>

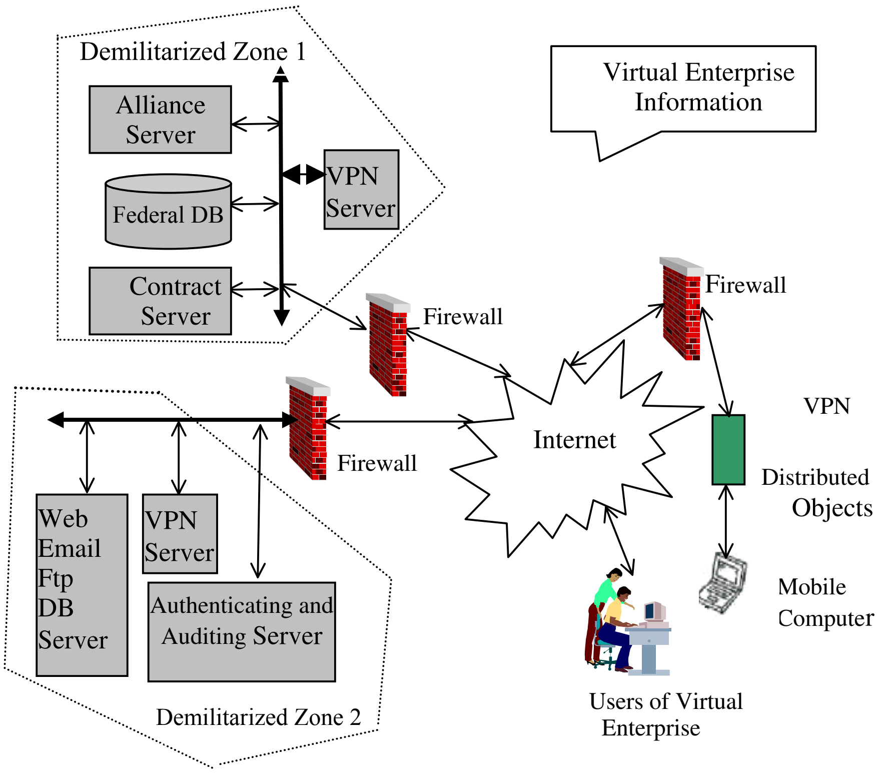


Figure 2 VPN Architecture

<Certificate-Authority>::=<Self-signed CA>
<Net-Management>::=<HTTP based>|<SNMP based>

3.1 Chaos based Information Encryption

The DES algorithm was developed in the 1970's by the IBM Corporation and has been adopted by the U.S. government as its encryption standard. DES is fairly fast and secure in certain situations, thus DES can be adopted to encrypt VE information and network objects. On the other hands, DES has some limitations and shortcomings as such some new encryption algorithms are studied in this paper.

Recently, chaotic cryptography[8] has become a very active research area. Chaos is used to treat stochastic and unpredictable phenomena. This stochastic-like behavior that chaotic oscillations presents has been used to hide information, in order to safely transmit secret message.

Suppose $\{P_n\}$ is plaintext information sequence, $\{X_n\}$ is key information sequence, and $\{C_n\}$ is ciphertext information sequence. Then the chaos based encryption is defined as follows:

$$\{C_n\} = \{P_n\} \otimes \{X_n\}$$

The chaos based decryption is defined as follows:

$$\{P_n\} = \{C_n\} \otimes \{X_n\}$$

We use the following Logistic mapping

$$X_{n+1} = 1 - \mu X_n^2$$

While $\mu \in (1.41, 2]$, $X \in [-1,1]$, this Logistic mapping is chaotic and generates chaotic sequence $\{X_n\}$. Text and image encryption has been implemented by using the Logistic chaotic mapping in our VPN system. The security of chaotic encryption depends on the sensitivity of the chaotic systems to their parameters and initial conditions. The theory analysis and computer simulating results show that the chaos encryption algorithm not only has good cryptography strength (the cipher has good diffusion properties with the plaintext and the key) and high reliability but also is easy to implement.

3.2 Key Management

In a VPN, since the encryption algorithm is public, key management is very important. We use the RSA algorithm for key management. Its principle is described as follows.

Before A wants to send the plaintext to B, A first encrypts the plaintext to the ciphertext with the session key, and then send the ciphertext. To protect the session key, the session key should be encrypted to form the ciphertext of the session key by using RSA algorithm with B's public key. They are processed as follows.

Ciphertext = DataEncrypt(Plaintext, SecctionKey)

SessionKeyCipher = SessionKeyEncryptedByRSA(SessionKey, PublicKeyOfB)

A⇒B : <Ciphertext, SessionKeyCipher>

(Notes: $A \Rightarrow B$: <message> means the message is transmitted from A to B)

At the B site, after getting the message, B first decrypts the ciphertext of the session key with B's private key to get the plaintext of the session key, and then use the session ky to get the plaintext of the data. They are processed as follows.

SessionKey=SessionKeyDecryptedByRSA(SessionKeyCipher, PrivateKeyOfB)

Plaintext = DataDecrypt(Ciphertext, SessionKey)

3.3 Chaos based User Identification Authentication

We use the property that chaotic system is sensitive to their parameters and initial condition, and generate chaotic sequence to represent user's identity "fingerprint". The property makes every user's identity "fingerprint" be unique, and it is impossible that any other person forges the "fingerprint". The Identity "fingerprint" is dynamically changed to enforce the security.

The chaotic sequence is a pseudo-random sequence, and can be easily generated by the iterative equation and the nonlinear equation or the partial differential equation. Considering the nonlinear iterative equation

$$X_{n+1} = f(X_{n;}\, \mu_i) \tag{1}$$

The above is a one-dimensional multiple parameter iterative equation, $f(X_{n;}\, \mu_i)$ is a nonlinear function, and μ_i $(i=1,2,3\cdots)$ is a parameter. When the parameter μ_i is selected as a proper value, the chaotic sequence $\{X_n\}$ is generated by equation (1). The chaotic sequence is sensitive to μ_i at the same initial condition. For example, even if A infinitesimally changes such as little as 1.0e-6, the chaotic sequence still varies a lot. Therefore, the random chaotic sequence $\{X_n\}$ is used as the user's identity "fingerprint".

In our VPN system, we adopt the mixed optically bistable model [9] as the chaotic source. The model is represented as a one-dimensional iterative equation

$$X_{n+1} = A \sin^2(X_n - X_B) \tag{2}$$

From the equation (1) and (2), the equation (3) is obtained.

$$f(X_{n;}\, A\, ,\, X_B) = A \sin^2(X_n - X_B) \tag{3}$$

Here $A=\mu_1$ and $X_B = \mu_2$, as the parameter A and X_B change, the function will be unstable from a certain point, and will get into the chaotic state by double-period bifurcation. In the chaotic area, the function output sequence $\{ X_n \}$ is a good random code. Because random property is very high, it is almost impossible to guess the chaotic sequence.

3.4 Guillou-Quisquater Protocol based Identification Authentication between Coordinator and Participant

A smart card is designed for a participant's identification authentication to the coordinator. The participant's identity consists of a set of credentials: a bit string J consisting of the card's name, validity period, a bank account number and whatever else the application warrants. J is analogous to the public key. Other information shared by all users of the application include an exponent v and a module n where n is the product of two secret primes calculated as $n=p*q$. The private key is B that must be satisfied with this formula $JB^V \equiv 1 \bmod n$. When the credential of a participant is sent to the coordinator, identification authentication protocol is described as follows:

(1) Participant selects a random integer r, $r \in [1,n-1]$, and calculates:

$T=r^V \bmod n$

participant $\Rightarrow$ *coordinator*: $< T, r, J >$

(2) Coordinator selects a random integer d , $d \in [0,v-1]$)

coordinator $\Rightarrow$ *participant*: $< d >$

(3) Participant calculates:

$D=rB^d \bmod n$

participant $\Rightarrow$ *coordinator* : $<D>$

(4) Coordinator calculates:

$T'=D^V J^d \bmod n$

If T=T' , then participant knows B, and the authentication is successful. Since $(JB^V)=1$, the proof is as follows

$T'=D^V J^d \bmod n =(rB^d)^V J^d \bmod n = r^V B^{dV} J^d \bmod n = r^V (JB^V)^d \bmod n = r^V \bmod n = T$

A coordinator authenticates the participant's identity, in turn, the participant can authenticate coordinator's identity. Hence, trust between coordinator and participant is guaranteed.

3.5 Guillou-Quisquater Protocol based Digital Signature with RSA Encryption

The Guillou-Quisquater identification authentication can be converted into a signature scheme. The public and private key set is the same as that indicated in section 3.4. The following is the protocol.

(1) Coordinator selects a random integer r, $r \in [1,n-1]$, and calculates:

$T=r^V \bmod n$

(2) Coordinator calculates:

$d = H(M,T) \bmod n$, where M is the message to be signed, $H(X)$ is a one-way hash function, and $d \in [0,v-1]$.

(3) Coordinator calculates:

$D=rB^d \bmod n$

The signature contains the message *M*, *d*, *D* and coordinator's identity *J*.

Signature = <M, d, D, J>

coordinator ⇒ participant: EncryptByRSA(Signature, ParticipantPublicKey)

(4) Participant uses RSA algorithm and his private key to decrypt the received message to get the signature:

DecryptByRSA(*EncryptByRSA(Signature, ParticipantPublicKey), PartcipantPrivateKey)*

$T'=D^V J^d \bmod n$

$d' = H(M, T')$.

If $d = d'$ then the coordinator must know the *B*, and the signature is valid.

The security of RSA and Guillou-Quisquater relies on the difficulty of factoring large integers. Implementing above protocol can ensure that message *M* certainly is announced by coordinator. Using Guillou-Quisquater digital signature with RSA encryption can ensure confidentiality, authentication and integrity in the transfer procedure of messages.

3.6 Certificate Authority and Digit Certificate

To support our VPN system, a certificate authority (CA) is constructed. The CA stores all public keys, and is considered as a trustworthy public key server. The CA can create, distribute and manage certificates. The certificate management functionality includes automatic roll-over, suspension and revocation. When user *A* needs *B*'s public key, he sends the credential request to the Certificate Authority. The Certificate Authority checks it and then authorizes and issues the digit certificate to *A*. The following gives the structure of a digit certificate.

<Digit_certificate>::={<validity>, <issuer information>, <user information>, <digital signature algorithm>, <necessary public key>}

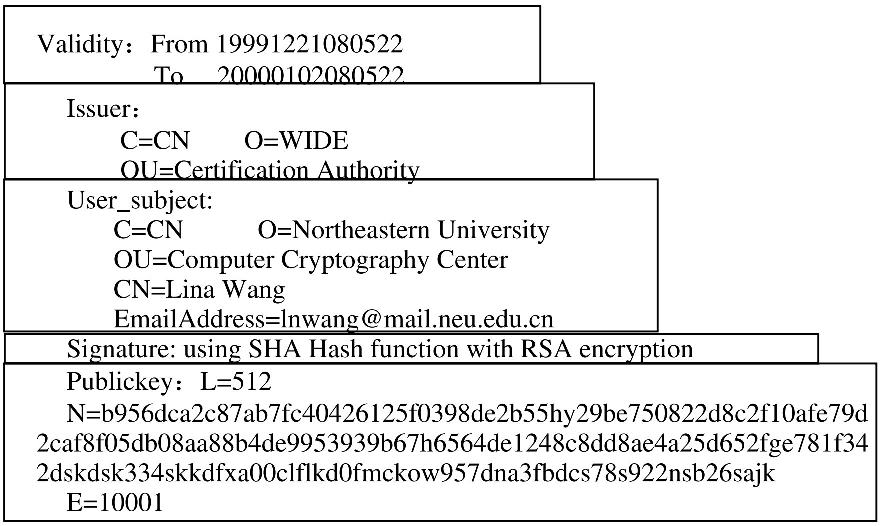
In addition, the CA should make a digital signature for the digit certificate using RSA algorithm to ensure that the digit certificate indeed comes from the CA. The processing is as follows.

Signature= SignatureByRSA(DigitCertificate, PrivateKeyOfCA)

Fig.3 illustrates an example of the digit certificate.

4 Implementation Issues of Security Server

Migration to a secure environment probably will involve substantial modifications to software. Applications that utilize secure facilities also require modification to support security interface protocols. Since security retrofitting is difficult, security issues should be designed into the system right from the start if possible. Commitments to secure capabilities must be made early in the architecture process to



Validity：From 19991221080522
To 20000102080522

Issuer：
C=CN O=WIDE
OU=Certification Authority

User_subject:
C=CN O=Northeastern University
OU=Computer Cryptography Center
CN=Lina Wang
EmailAddress=lnwang@mail.neu.edu.cn

Signature: using SHA Hash function with RSA encryption

Publickey：L=512
N=b956dca2c87ab7fc40426125f0398de2b55hy29be750822d8c2f10afe79d
2caf8f05db08aa88b4de9953939b67h6564de1248c8dd8ae4a25d652fge781f34
2dskdsk334skkdfxa00clflkd0fmckow957dna3fbdcs78s922nsb26sajk
E=10001

Figure 3 Sample of a Digit Certificate Structure

avoid the substantial risks and costs of retrofitting. Therefore, an architect should plan for the evolution of the system and the security facilities as commercial technology and standards evolution. However, a VE might contains many legacy application systems that were not designed with security in mind, or were designed to work on a secure dedicated private network. To solve the problem, the legacy application systems are wrapped as CORBA objects, and the designed encryption algorithm aims at encrypting the CORBA objects in ViaScope.

In the VPN system, the primary servers are designed as follows.

(1) Cryptography server is designed to provide access to various mechanisms and algorithms to implement confidentiality of network object data in the VE information system.

(2) Authentication server provides user authentication and system authentication.

(3) Signature server provides digital signature to ensure integrity and authentication of information.

(4) Audit server provides a centralized auditing facility that can receive messages from all parts of the system. It keeps a system secure log, and provides a centralized auditing function. The format of the log is as follows:

```
<Sec-Log>::= { <User-ID>, <Actions>, <Time>}
<Actions>::=<Actions on Object>|<Actions on Alliance>
<Actions on Object>::= <Create-object>| <Delete-object>|<Update-object>|
                 <Query-object>|<Browse-object > |<Authorize-object>
<Actions on Alliance>::= < Construct-alliance >|< Join-alliance    >
                 |<Withdraw-alliance>|<Information-authorization>
```

For implementation of the encryption, authentication, signature and interface, JavaAPI is used with its rich classes and methods, such as exponent and module calculation in BigInteger class[10].

5 Conclusion

The VE information system is designed based on ViaScope, in which distributed, heterogeneous and autonomous information system raises many security problems. In this paper we designed a VPN to ensure the security of VE. In the VPN we adopted several novel techniques, especially proposed a chaos theory based user's "fingerprint" authentication algorithm by using mixed optically bistable chaos model, and it has been proved as reliable and effective by the computer simulation experiments. The designed chaotic encryption algorithm using Logistic chaotic mapping has good cryptography strength due to the sensitivity of the chaotic systems to their parameters and initial conditions. The cryptography server and basic authentication server have been implemented as the main part of the VPN system. Further work includes the research of key techniques for chaos theory based VPN security. The chaotic cryptanalysis is still at its beginning with very few results littered among a huge ocean of chaotic cryptography. It is very necessary to develop chaotic cryptanalysis because the security of chaotic cryptography scheme can not be measured by traditional cryptanalysis methods. We need to develop the chaotic cryptanalysis which functions as the inner motivation of the further study of chaotic cryptography. We also need to provide the criteria for judging the degree of the security of chaotic cryptography.

References

[1] L.M.Camarinha-Matos, H.Afsarmanesh, C.Garita, C.Lima, Towards an architecture for virtual enterprises, Journal of Intelligent Manufacturing,, 9(2), Apr. 1998, pp.189-199.
[2] O. Richard, A.Zarli. WONDA: An Architecture for Business Objects in Virtual Enterprise. Proc. of OOPSLA'98 Workshop on Objects, Components & VE, Vancouver, 1998.
[3] E.Rdeke,J.Korzonnek, Distributed Information Management in Virtual Engineering Enterprise by GEN, Proc. of the 9th Intl Workshop on Research Issues on Data Engineering Information Technology for Virtual Enterprises, Sydney, 1999.
[4] A.D.Rubin, D.Geer, A Survey of Web Security, IEEE Computer, 1998,31(9).
[5] P.W.Dowd, J.T.Mchenry, Network Security: It is Time to Take it Seriously, IEEE Computer, 1998,31(9): pp.24-28.
[6] Tina Bird, Building VPNs: The 10-POINT PLAN, Data Communication, Jun. 1998, pp.123-130.
[7] X.Yang, G.Wang, G.Yu, D.Lee, Modeling Enterprise Objects in a Virtual Enterprise Integrating System, Lecture Notes in Computer Science 1749, pp.166-175.
[8] M.S.Baptista, Cryptography with Chaos, Physics Letters A 240(1998), pp.50-54.
[9] H.Zhang , J.Dai, P.Wang. Bifurcation and Chaos in an Optically Bistable Liquid-Crystal Device. Journal of Optical Soc Am B, 1986, 3(2).
[10] L.Wang, Y.Hamazaki, Design of New Encryption and Authentication Model For Distributed Network Environment Using Object-Oriented Approach, Proc. of ICYCS'99, Aug. 1999, Nanjing.

Adaptive Online Retail Web Site Based on Hidden Markov Model

Shi Wang, Wen Gao, Tiejun Huang, Jiyong Ma, Jintao Li, Hui Xie

Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080
{shiwang, wgao, tjhuang, jyma, jtli, xiehui}@ict.ac.cn

Abstract. There are two problems in online retail: 1) The different interest of all customers in the different commodities conflicts with the commodity classification structure of the web site; 2) Customers will simultaneously buy some commodities that are classified in different classes and levels in the web site. The two problems will make the majority of customers access overabundant web pages. To solve these problems, we mine the web data to build a hidden markov model, use association rule discovery to get the large item sets, use Viterbi algorithm to find some paths, mark the large item sets in the nodes of the paths. The large item sets will compete in the nodes for the limited space. Through this approach the web site will adjust itself to reduce the total access times of all users. This method also can be used in analyzing paths, advertisements, and reconstructing the web site.

1 Introduction

Electronic commerce is one of the most important applications in Internet. WWW is wining more and more users through its simplicity and facility. They provide the users and the merchants the ideal space of intercommunication and virtual trade. In an e-commerce environment, an online retail model [1] is depicted in Fig. 1.

The Marketing Data stores the commodity information and users' transaction information. The Web Page Data stores the web pages and their structures. The Server Data mainly store the log file.

A standard online retail classification structure of a web site is depicted in Fig. 2. In Fig. 2, each node represents a web page. The *N* nodes represent the navigation web pages or the classification web pages; the *C* nodes represent the content web pages or purchase web pages. The structure is directed graph. The designer of the web site will design many paths to a purchase web page. One commodity can belong to different classifications. Such as the electronic book, it belongs to the book and also belongs to the electronic production. The larger the scale of a web site, the more complex the web site is in the structure.

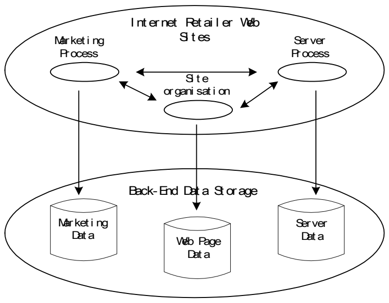


Fig. 1. Online Retail Model

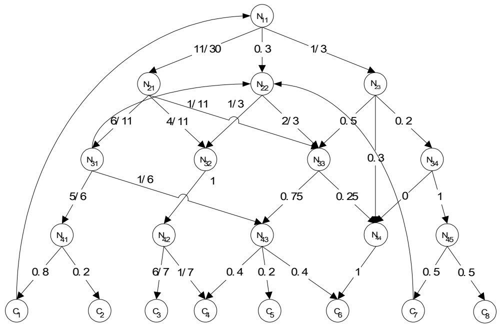


Fig. 2. A standard online retail classification structure model of a web site

There is an important problem in online retail web site. The users facing the abundant commodity information don't know how to select effectively. On the other hand, the merchant facing the users don't know which commodities are their favorites, so the merchant don't know how to adjust his service manner and commodity structures:

1. Users have different interest in retail commodities. There is a probability distribution in users' purchase interest. The interest of the users in some commodities is more than in other commodities. But the classification level design of web pages must comply with the commodity classification level structure. Because if the design doesn't comply with it, a general user won't be able to access the web site. So there is a conflict between the two aspects. The result of the conflict will make many users browse many irrelevant web pages and enter too many web levels. The method that resolves this problem is to change the navigation web page (*N* node) in Fig. 2 to the navigation and content web page (*NC* node). Then the users can buy some commodities that are often purchased in the *NC* node. Some existing methods to create the *NC* nodes are mainly according to the designer's experience or the simple statistic method. But because the interest of users is always changing and the web site become more and more complex, the two methods have their disadvantages. So we present a new approach to resolve this problem more automatically, more effectively, more rationally, and more intelligently.

2. Some commodities purchased by many users are similar to the beer and diaper, which is the classical problem in data mining. In online retail, there are great distances in some commodities' classification structure, but many users always purchase these commodities at the same time. So these users must repeatedly enter

and exit many web pages to purchase them. For these association commodity sets, we present the new approach to find them and create the *NC* nodes that contain them automatically in order to help users to access the web site effectively.

So we need to build a model and present an algorithm to mark the commodities that are purchased frequently by the majority of users in appropriate *N* nodes. The algorithm automatically changes the *N* nodes to the *NC* nodes; the original navigation relations between the *N* nodes won't be destroyed. The *NC* nodes will satisfy the users' primary requirement. These nodes will reduce the number of the users access web pages and levels.

According to the structure of the online retail web site, the log recording the user access manner, and the record of transactions, our approach can build a hidden markov model. The state nodes of the model are the web site's *N* nodes. The starting state is the root page of the web site. The final state nodes are the web site's *C* nodes. The access probability of all the users from an *N* node to another *N* node is the transition probability of a state to another state in the hidden markov model. There must be a hyperlink in the two nodes. The purchase probability that all the users buy a commodity through a web page is the observation or emission probability of this commodity in this web page. After the hidden markov model is built, we can employ the association rule discovery algorithm to find the association commodity sets (i.e. the large item sets) in the users' transaction. We find the content page which the association commodity set's center is in. Then we can use the Viterbi algorithm to find the *j* paths that have the maximal purchase probability from the root web page to the content page. We mark the association commodity set and the purchase probability of this set in the root node in the paths. We use this method to deal with all association commodity sets. The association commodity sets will compete in the nodes for the limited station. Through this method the web site will adjust itself to reduce the users' total access times. This method also can be used in analyzing paths, advertisements, and reconstructing the web site.

Buchner et al [1] firstly describe a way of combining data mining techniques on Internet data in order to discover actionable marketing intelligence. The data not only covers various types of the server and web page information, but also the marketing data and knowledge. They give a total mining frame in electronic commerce. But their approach is limited in classical mining approach. Our approach is built on their foundation. We not only apply the classical mining approach (such as association rule discovery [2]) but also use the hidden markov model to adjust the web page content.

Cooley et al [3] firstly give the definition and classification of the web mining and give a system WEBMINER that mines the web usage. Their basic idea is to process the log in a web site, organize the log data into the transactions. Then they can use the classical data mining approaches such as association rule discovery to mine the data. The result can't be used to adjust the web site's content automatically. Our approach not only gets the mining result but also uses the result to improve the web site automatically. The mining result can help the users to access the web site efficiently.

PageGather [4] approach proposed the idea of optimizing the structure of Web sites based on co-occurrence patterns of pages within usage data for the site. In essence, it takes a web server access log as input and maps it into a form ready for clustering; it then applies cluster mining to the data and produces candidate index-page contents as output. The approach will flat the web site's structure and when there are a large numbers of web pages; the number of the index page will become overabundant. Our approach is similar to this approach in the way of the mining goal. But our approach is based on the hidden markov model and our approach won't destroy the web classification structure. Our approach won't add the additional index pages; the content upgraded will appear in the appropriate nodes.

Section 2 describes the discrete output, first-order hidden markov model that we use. In section 3, we build the hidden markov model on the online retail site and we depict how we use the model to resolve the problem. Section 4 depicts the mining objects and the initialization process. In section 5, we process the experiments.

2 Hidden Markov Model

Hidden Markov model (HMM)[5], widely used for speech recognition and part-of-speech tagging, provides a natural framework for this problem. In this paper, we employ the discrete output, first-order HMM:

1. A set of states Q, with specified initial and final state q_I and q_F.
2. A set of transitions between states ($q \rightarrow q'$).
3. A discrete vocabulary of output symbols $\Sigma = \sigma_1, \sigma_2, \ldots, \sigma_M$.

The model generates a string $X = x_1, x_2, \ldots, x_l$ by beginning in the initial state, transitioning to a new state, emitting an output symbol, transitioning to another state, emitting another symbol, and so on, until a transition is made into the final state. The parameters of the model are the transition probabilities $P(q \rightarrow q')$ that one state follows another and the emission probabilities $P(\sigma \mid q)$ that a state emits a particular output symbol. The probability of a string X being emitted by a HMM M is computed as a sum over all possible paths by:

$$P(X \mid M) = \sum_{q_1,\ldots,q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k) P(x_k \mid q_k) \tag{1}$$

Where q_0 and q_{l+1} are restricted to be q_I and q_F respectively, and x_{l+1} is an end-of-string token.

A main goal of learning problems that use HMM is to recover the state sequence $V(X|M)$ that has the highest probability of having produced an observation sequence:

$$V(X \mid M) = \underset{q_1 \ldots q_l \in Q^l}{\arg\max} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k) p(x_k \mid q_k) \tag{2}$$

There is an efficient algorithm, called the Viterbi algorithm that efficiently recovers this state sequence.

3 HMM for Online Retail

3.1 Defining the Model

1. *N* or *C* node is the state *q* in the HMM.
2. There is a commodity set $\Sigma = \sigma_1, \sigma_2, \ldots, \sigma_M$. σ_i is a commodity.
3. *C* node includes a subset $(\sigma'_1, \ldots, \sigma'_m)$ of Σ.
4. *q* and *q'* are linked directly by hyperlink. They have a transition probability $P(q \to q')$
5. The purchase probability $P(\sigma | q)$ is that the users buy σ through *q*.

Through analyzing the Log file, we can build a HMM that satisfies these definitions.

3.2 Association Rule Discovery

Giving the support and confidence, we can employ the association rule discovery algorithm to discover rules on the transaction sets. Once we get the rules, we can cluster in the rules in order to get the association commodity sets.

In each association commodity sets, the commodity that has the maximal purchase probability is the center of the cluster.

3.3 Discovering the Paths

1. To each association commodity set $x_i = (\sigma'_1, \ldots, \sigma'_m)$, the root web page and the web page which the center of the association set is in are the initial and final states q_I and q_F. The commodity set is the emitting symbol from the initial state of the

sequence to the final state of the sequence. We can use the Viterbi algorithm to get the *j* state paths that have the maximal purchase probability.

2. The purchase sequence $X = x_1, x_2, \ldots, x_l$, and $x_1 = x_2 = \ldots = x_l = (\sigma'_1, \ldots, \sigma'_m)$.

3.

$$V(X \mid M) = \arg\max_{\substack{1 \ldots j \\ q_1 \ldots q_l \in Q^l}} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k) p(x_k \mid q_k) \tag{3}$$

So these paths have the large purchase probability. In general, users access the online retail web site through these paths to buy this association commodity set.

4. We mark the association commodity set and the purchase probability (in the root node) of this set in the *N* node of the paths.

5. To all association commodity sets does these steps.

6. In the end, because of the limited space in the *N* nodes the association commodity sets will compete according to their purchase probabilities in each *N* node. Only those commodities that have large purchase probabilities will be reserved.

7. The original navigation structure won't be destroyed through this method.

4 Initialization of the HMM

4.1 Mining Objects

The mining objects are stored in the Back-End Data Storage in Fig. 1.

1. The Log (the server data in Fig. 1) conforms the standard of W3C[6]:

Table 1. The Log format

Field	Description
Date	Date, time, and timezone of request
Client IP	Remote host IP and / or DNS entry
User name	Remote log name of the user
Bytes	Bytes transferred (sent and received)
Server	Server name, IP address and port
Request	URI query and stem
Status	http status code returned to the client
Service name	Requested service name
Time taken	Time taken for transaction to complete
Protocol version	Version of used transfer protocol
User agent	Service provider

Cookie	Cookie ID
Referrer	Previous page
…	…

2. The purchase transaction data (the marketing data in Fig. 1.) records the users' purchase information.

Table 2. The purchase transaction

Field	Description
TransactionID	Transaction Identification
Commodities	The Commodities that the user purchases in the transaction

3. The web page structure information (the web page data in Fig. 1.).

4.2 Computing the Transition Probabilities and the Purchase Probabilities

Before mining, we need to transform the log to the user access transactions. L is the user's access log, its each entry $l \in L$ includes: the client user's IP address *l.ip*, the client user's identification *l.uid*, the accessed URL *l.url*, and the access time *l.time*. Then the access transaction t is:

$$t =< ip_t, uid_t, \{(l_1^t.url, l_1^t.time), ..., (l_m^t.url, l_m^t.time)\} >$$

$$where, for\ 1 \le\ k\ \le m,\ \ l_k^t \in L,\ \ l_k^t.ip = ip_t,\ \ l_k^t.uid = uid_t,\ \ l_k^t.time - l_{k-1}^t.time \le C$$

C is a stationary time window. The algorithm that finds the access transaction is:

1. Pre-process the log.
2. Partition the log according to each user's IP address *l.ip* to form each user's access set.
3. Partition each user's access set according to C to find access transactions.
4. Sort all access transaction according to the time.

After we processed the log, we have two transaction sets: 1) the access transaction set *Ta*, 2) the purchase transaction set *Tt* (we can get this set directly from the marketing data).

Then we process *Ta* and *Tt*. We combine them to *Tat* transaction set. Each transaction in *Tat* not only records the purchase transaction but also records the path through which the user access the web site:

Table 3. The access and purchase transaction

Field	Description
TransactionID	Transaction Identification
Commodities	The Commodities that the user purchases in the transaction
AccessPath	The user went through the web path for buying the goods.

The *Tat* transaction set is the foundation of building HMM.

We compute the one-step transition probability of two linked nodes in *Tat*:

$$p(q_i \to q_j) \approx \frac{count(q_i \to q_j)}{count(q_i)} \tag{4}$$

The $q_i \to q_j$ represents the q_i and q_j are linked directly by hyperlink. The $count(q_i \to q_j)$ represents the number of the transaction in which users access the web site from q_i to q_j in one step in *Tat*. The $count(q_i)$ represents the number of the transactions in *Tat*, each of which has q_i.

The purchase probability $P(\sigma | q)$is that all the users buy σ through q:

$$p(\sigma | q) \approx \frac{count(\sigma \wedge q)}{count(q)} \tag{5}$$

The $count(\sigma \wedge q)$ represents the number of the transactions in *Tat*, in each of which σ and q appear in the same time. The $count(q)$ represents the number of the transactions in *Tat*, each of which has q. All probabilities are computed through the web site's structure and *Tat*.

The transition probability and the purchase probability have some correlation, but the transition of the net relies on the product of both. In this way, the correlation doesn't affect the algorithm.

5 Experiments

Our experiments have three steps: First, in the simulation phase (we build an example web site according to Fig. 2. The web site has 8 commodities. We run the web site for some times in order to get the simulative data). Through this way we want to explain two problems. 1) Why do we use the hidden markov model but not use the markov model (MM)? 2) Our approach is better than the simple statistic approach. Second, we experiment in real environment. Third, we use our approach to analyze the paths in general web site.

5.1 Comparison

5.1.1 Comparing HMM with MM

Why we use hidden markov model but not use markov model (MM)? In Fig. 2, the transition probabilities have been marked. In Table 4, the purchase probability of each commodity in each node is given (each content web page only has one commodity, C_1 has G_1, …, C_8 has G_8):

Table 4. The purchase probability of each commodity in each node

	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8
N_{11}	4/30	1/30	6/30	5/30	2/30	10/30	1/30	1/30
N_{21}	4/11	1/11	3/11	1/11	0	2/11	0	0
N_{22}	0	0	3/9	4/9	0	2/9	0	0
N_{23}	0	0	0	0	0.2	0.6	0.1	0.1
N_{31}	4/6	1/6	0	0	0	1/6	0	0
N_{32}	0	0	6/7	1/7	0	0	0	0
N_{33}	0	0	0	2/6	1/6	3/6	0	0
N_{34}	0	0	0	0	0	0	0.5	0.5
N_{41}	0.8	0.2	0	0	0	0	0	0
N_{42}	0	0	6/7	1/7	0	0	0	0
N_{43}	0	0	0	0.4	0.2	0.4	0	0
N_{44}	0	0	0	0	0	1	0	0
N_{45}	0	0	0	0	0	0	0.5	0.5

If we use MM only to compute one-step transition probability, two paths from N_{11} to C_6 are:

1. Path1(N_{11},N_{23},N_{33},N_{43},C_6) The transition probability from N_{11} through N_{23},N_{33},N_{43} to C_6:
   $P(N_{11}\rightarrow N_{23})\times P(N_{23}\rightarrow N_{33}) \times P(N_{33}\rightarrow N_{43}) \times P(N_{43}\rightarrow C_6)$=1/3×0.5×0.75×0.4=0.05.
2. Path2(N_{11},N_{23},N_{33}, N_{44},C_6) The transition probability from N_{11} through N_{23},N_{33},N_{44} to C_6:
   $P(N_{11}\rightarrow N_{23}) \times P(N_{23}\rightarrow N_{33}) \times P(N_{33}\rightarrow N_{44}) \times P(N_{44}\rightarrow C_6)$ = 1/3×0.5×0.25×1=0.125/3.

So we will choose Path1. But if we use HMM, then:

1. Path1(N_{11},N_{23},N_{33},N_{43},C_6) The probability that *G6* is always purchased in each node of the path from N_{11} through N_{23},N_{33},N_{43} to C_6:
   $(P(N_{11}\rightarrow N_{23})\times P(G_6|N_{23})) \times (P(N_{23}\rightarrow N_{33})\times P(G_6|N_{33})) \times (P(N_{33}\rightarrow N_{43})\times P(G_6|N_{43})) \times (P(N_{43}\rightarrow C_6)\times P(G_6|C_6))$ = (1/3×0.6) × (0.5×3/6) × (0.75×0.4) × (0.4×1) =0.006.
2. Path2(N_{11},N_{23},N_{33},N_{44},C_6) The probability that G_6 is always purchased in each node of the path from N_{11} through N_{23},N_{33},N_{44} to C_6:
   $(P(N_{11}\rightarrow N_{23})\times P(G_6|N_{23})) \times (P(N_{23}\rightarrow N_{33})\times P(G_6|N_{33})) \times (P(N_{33}\rightarrow N_{44})\times P(G_6|N_{44})) \times (P(N_{44}\rightarrow C_6)\times P(G_6|C_6))$ = (1/3×0.6) × (0.5×3/6) × (0.25×1) × (1×1) =0.0125.

So we will choose Path2. Only to define the transition probability but not to use the purchase probability, there will be this problem: although Path1 has the bigger transition probability than Path2, the probability of users through Path1 to buy G_6 isn't always bigger than Path2.

Through using association rule discovery algorithm, we can get the large item sets: G_6, G_3, G_4, G_1, (G_6, G_4). $G_6\rightarrow G_4$ is an association rule.

There are many paths from N_{11} to C_6, according to Viterbi algorithm, we can get 3 paths that have maximal purchase probability:

Table 5. The 3 paths that have maximal purchase probability from N_{11} to C_6

The path from N_{11} toC_6	The probability that G_6 is purchased
Path $(N_{11},N_{23},N_{44},C_6)$	$(1/3\times0.3\times1)\times(0.6\times1\times1)=0.06$
Path $(N_{11},N_{23},N_{33},N_{44},C_6)$	$(1/3\times0.5\times0.25\times1)\times(0.6\times0.5\times1\times1)=0.0125$
Path $(N_{11},N_{23},N_{33},N_{43},C_6)$	$(1/3\times0.5\times0.75\times0.4)\times(0.6\times0.5\times0.4\times1)=0.006$

Because there aren't other competitors in this example, we can mark G_6, G_4 in the nodes of these paths.

5.1.2 Comparing with the Simple Statistic Approaches

There are two kinds of the simple statistic approaches. The two kinds have some disadvantages in the non-tree web site:

Each commodity has two purchase probability, one is $P(G \mid N_l)$(G's purchase probability in the root), another is $P(G \mid N_l)$(G's purchase probability in the N_l node).

1. For an N node, we get the some commodities. These commodities are in all C nodes that can be reached from the N node. Then the commodity that has the maximal purchase probability $P(G \mid N_l)$ in the commodity set is put in the N node.

2. For an N_l node, we get the some commodities. These commodities are in all C nodes that can be reached from the N_l node. Then the commodity that has the maximal purchase probability $P(G \mid N_l)$ in the commodity set is put in the N_l node.

There is a disadvantage in the first approach. In the example in Fig. 2, in N_{31}, according to this approach, the G_6 is put in this node. But through this node, users' purchase intention is mainly G_1 not G_6. The second approach is an improvement to the first approach. We can get the result from HMM directly.

If the site is the complex net structure, in the example in Fig. 2, when the commodities compete in N nodes, the second approach will meet such problem:

Let each N node only store one commodity. According to the second simply statistic approach, G_6 is put in N_{11}, G_1 is put in N_{21}, and G_4 is put in N_{22}; But because G_3 is directly inferior to G_6 in the purchase probability, G_3 should be put in one of N_{21} and N_{22}. So there is a problem:

When the structure of the site isn't a tree structure, if a commodity is always purchased, the designer of the site will design many paths for this commodity. But these paths will reduce the purchase probability in the nodes of these paths. If there is only one path to a commodity (in general it has the lower purchase probability), because its purchase probability isn't distributed, it will be arranged ahead.

To solve this problem, we must analyze the user access paths. To get the paths, we build a hidden markov model. We use association rule discovery to get the large item set. We use Viterbi algorithm to find some paths that come from the root web page to the web page that the center of the large item set is in it. We mark this large item set in the nodes that are in the path. Through these steps, we can calculate the total item sets and mark them in these paths. The large item sets will compete in the nodes for the limited space. In the above example, we only choose the path through N_{21}. In the end, G_3 is put in N_{21}. According to our approach, G_4 is put in N_{22}, G_3 is put in N_{21}, and G_1 is put in N_{31}…. Our approach is better than the second simple statistic approach. If the site is a tree structure, our approach is equal to the second simple statistic approach.

5.2 Experiment on the Real Background

The Proxy Server is an ideal experiment environment. Through the proxy, we can get the proxy's log that records some users' access to a web site (the record is equal to the site's log) and we can also get the web pages in the site. The mp3 music web site is a good simulation of an online retail site. Because many users like to download the mp3 music, which is equal to that users purchase the commodities. So we can track a mp3 music site through the Proxy in order to get all mining data. We analyze the data of three mp3 music sites in three months. We build a HMM for mining:

Table 6. The mining result of three mp3 music sites

Site NO	Site Type	The number of the Access Transactions	The number of the large items	The total number of the paths to the large items	The number of the levels	Compare with the second simple statistic approach（j is 50% of the number of the total paths)
Site1	FTP	3124	10	10	4	Same
Site2	HTTP	1027	12	31	4	There are 3 large items that are put in irrational way; HMM hasn't the problem
Site3	HTTP	1311	13	57	5	There are 4 large items that are put in irrational way; HMM hasn't the problem

Our experiments manifest: if the structure of an online retail web site is a net structure, if the level of the site is deep, and if the structure of the site is complex, our approach will have a better result than the simply statistic approach. Our approach can process the large-scale, complex self-adaptation problem.

5.3 Experiment about Analyzing Path

Through the Proxy, we choose some complex web sites in order to analyze the paths. Some page is regarded as a C node and other pages are regarded as an N node. We can find some frequently accessed paths that are from the root page of the site to this

C node. The designer of the site doesn't perhaps notice these paths. Our approach can been used broadly in analyzing path.

6 Conclusions and Future Work

Our approach is essentially a recommendation approach based on web usage mining. Through mining the user access record and the transaction record, we recommend the mining result in order to accelerate the users' access. We firstly use the HMM to analyze the paths in online retail web site and expand the application field of the HMM. This approach resolves the self-adaptation problem of the online retail web site. It also can be used broadly in path analyzing in the web site. In the approach, building the HMM doesn't require the complex training process and the transition probability and the purchase probability are easily calculated.

There are some characteristics in our approach. 1) It is a kind of optimizing approach. 2) The mining object is the interactive action and the common interest, and the mining result faces up to the total users. 3) The result is the well self-organized *NC* nodes, which is equal to visualizing the mining results. 4) The mining result doesn't influence the existing navigation structure in *N* nodes.

The next step of our works not only explores the recommendation approaches but also explores the prediction approaches. Through combining the two approaches, we can recommend the mining result and predict the interest of the users in the future.

Reference

1. Buchner, A. G. and Mulvenna M.D. Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining. SIGMOD Record 27(4). 1998.
2. Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In Proc. Of the 20th VLDB Conference, 487-499, Santiago, Chile, 1994.
3. Cooley, R., Mobasher, B., et al. Web Mining: Information and Pattern Discovery on the World Wide Web. In Proceedings of International Conference on Tools with Artificial Intelligence, Newport Beach, IEEE, 1997.
4. Perkowitz, M., and Etzioni, O. Adaptive Web Sites: Automatically Synthesizing Web Pages. in Proceedings of AAAI98. 1998.
5. Rabiner, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2): 257-286, 1989.
6. Luotonen, A. The common log file format. http://www.w3.org/pub/WWW/, 1995.

A Protocol for Untraceable Electronic Cash

Hua Wang[1],* and Yanchun Zhang[2]

[1] Department of Computer Science
University of Information Engineering, ZhengZhou 450002 China
wang@usq.edu.au
[2] Department of Maths & Computing
University of Southern Queensland, Toowoomba QLD 4350 Australia
yan@usq.edu.au

Abstract. In this paper we develop a new protocol for untraceable electronic cash, in which the bank involvement in the payment transaction between a user and a receiver is eliminated. The user withdraws electronic "coins" from the bank and uses them to pay to a receiver. The receiver subsequently deposits the coins back to the bank. In the process the user remains anonymous, unless she spends a single coin more than once (double spend). The security of the system is based on DLA (Discrete Logarithm Assumption) and the cut-and-choose methodology.

1 Introduction

Electronic cash (or "e-cash"), can provide stronger anonymity than physical cash, since the latter clearly displays serial numbers which can be recorded to allow further tracing. Contrary to their physical counterparts, e-cash has an inherent limitation; it is easy to copy and reuse. To guard against what is traditionally called "double-spending" of electronic coins, Chaum [1] proposed for the bank to keep a list of spent coins and check every deposited coin against this list. Hence the bank has to be on-line during the payment, otherwise the shop will have no guarantee that the coins received are valid. Forcing the bank to be on-line during the payment is a very strict requirement; consider that in the current payment systems the number of payments is at least an order of magnitude larger than the number of deposits, which are performed in batch mode usually at the end of each day. Checking each payment on-line would force banks to migrate to more expensive computers which allow real-time payment verifications. Furthermore, these computers would present a single point of failure for the system since no payments could be conducted off-line.

To bypass the on-line requirement, Chaum, Fiat and Naor [2] proposed to detect, instead of preventing, double-spending. In their model, called off-line electronic cash, anonymity holds only as long as users are legitimate, whereas

* The first author now is a visiting scholar at University of Southern Queensland in Australia from University of Info. Engeering, China

double-spenders are identified. Off-line electronic cash has since enjoyed considerable attention in the research community, aiming to optimize the originally proposed system for practical applications.

Anonymity is the distinguishing property of e-cash; the user privacy is guaranteed even against collaboration of all other involved parties, i.e., bank and shops. There have been several references in the literature for anonymous-like systems, providing partial privacy under the assumption of no collaboration between involved parties [3, 4], or assuming the central bank is honest. These are very strong and potentially unjustified assumptions, especially in light of financial benefits from disclosing personal information and efficient ways of collaboration between interested parties, such as bulk transfers of electronic files.

The off-line property is equally significant in e-cash, even in scenarios where shops and bank happen to be the same party (e.g., postal services): on-line systems require constant and real-time involvement of the bank in every payment transaction, resulting in excessive communication and computation costs. In contrast, off-line systems usually operate in dual mode, verifying high-cost transactions on-line (to prevent high-volume fraud, even though the fraudulent party is identified), while the bulk of payments are processed in batch mode by the bank. Several currently employed "electronic cash" (or e-cash) payment systems, such as credit cards, bank and personal checks, operate in this dual-mode operation, showing the viability of the concept.

In this paper, we propose an untraceable, off-line electronic cash scheme which achieves provable security without the use of general computation protocols and without the requirement of a trusted party, or of a costly general computation-based initialization procedure. To illustrate the practicality of schemes that are not based on general computation protocols, we then show how to derive an efficient variant based on the random-oracle model (a random database for keys and users etc, see [13]); this variant thus achieves provable security based on DLA and the existence of random oracle-like hash functions, while its efficiency is on par with heuristic cut-and-choose based systems, such as that in [5]. Further more our untraceable electronic cash protocol is much more simple. An implication of this protocol is that truly anonymous e-cash can be implemented very efficiently without sacrificing security in comparison to existing account-based or anonymous-like systems.

This paper is organized as follows: in section 2, we give a brief historical view of electronic cash and the basic model of electronic cash is presented in section 3. In section 4, a new off-line electronic cash scheme is designed and the security analysis of our scheme is given in section 5. Finally, Conclusions are included in section 6.

2 Historical Overview

We proceed with providing a short historical view of electronic cash; this overview excludes enhancements to the basic model such as divisibility or anonymity revocation mechanisms, which will be discussed in later papers.

Off-line anonymous electronic cash was introduced by Chaum, Fiat and Naor [2]. The security of their scheme relied on some arbitrary assumptions however and no formal proof was attempted. Although hardly practical, their system demonstrated how off-line e-cash can be constructed and laid the foundation for more secure and efficient schemes to follow. Their methodology was conceptually simple: at withdrawal the bank verifies in a zero-knowledge manner that the user's identity is "embedded" (encrypted) in a randomly created (by the user) coin, and then provides the user with a blind signature on this coin; at payment users provide a distinct "hint" on their identity, such that one hint provides a computationally secure and unconditionally blinded commitment on the user's identity, whereas any two hints identify the user. Hints are verifiable by the shop, i.e., a zero-knowledge proof that the hint corresponds to the identity in the coin is given. At deposit the shop just transfers a payment transcript to the bank. Upon double-spending a coin the bank identifies the user using the two distinct "hints" on his identity. The zero-knowledge proofs during withdrawal and payment were using the cut-and-choose technique [14].

Okamoto and Ohta [5] were the first to attempt an improvement on this system. They modified the model by moving the most complex part of the functionality of the withdrawal protocol, namely the zero-knowledge proof of the user's identity, to the user setup (account establishment) protocol which is executed much less frequently. In the latter the user obtains an untraceable "license" which he uses for withdrawals of coins. Thus, anonymity is established only once, in the form of a pseudonym, instead of being "refreshed" with every withdrawn coin. Hence, a user's coins are linkable and users may be traced with conventional techniques, i.e., using locality, time, type, size, frequency of payments, or by finding a single payment in which the user identified himself. The authors suggested that a compromise between the efficiency and the unlinkability of the system can be found by running the account-establishment protocol more than once. The system relies on more reasonable and clarified assumptions but, it is faster than [2], at least when the account establishment protocol is performed infrequently; otherwise no improvement is claimed.

In 1991, Pfitzmann and Waidner [6] presented a way to base on-line electronic cash on general two-party computation protocols and zero-knowledge proofs. The purpose was not to create a practical system but to show that provably secure electronic cash does exist. To combine the security of [6] with the relative efficiency of regular cut-and-choose systems, Franklin and Yung [7] presented a provably secure scheme that is not based on general computation protocols. The security relied on the DLA and on the existence of a mutually trusted party (equivalently, a general computation protocol could substitute the trusted party's functionality). Although a cut-and-choose techniques were used and its efficiency was still not a prime consideration, Franklin and Yung were the first to illustrate how off-line e-cash can be based on the DLA, as well as the first to construct a formal security model; variations of this security model have appeared in subsequent e-cash systems [8, 9].

1995, Chan and Frankel [8] presented a provably secure off-line e-cash scheme which relies only on the security of RSA. This cut-and-choose based scheme extends the work of Franklin and Yung [7] who aimed to achieve provable security without the use of general computation protocols. In 1998, T. Yiannis and M. Yung [12] showed that the decision Diffie-Hellman assumption implies the security of the original ElGamal encryption scheme (with messages from a subgroup) without modification and they also showed that the opposite direction holds, i.e., the semantic security of the ElGamal encryption is actually equivalent to the decision Diffie-Hellman problem. They presented additions on ElGamal encryption which result in non-malleability under adaptive chosen plaintext attacks. Non-malleability is equivalent to the decision Diffie-Hellman assumption, the existence of a random oracle (in practice a secure hash function) or a trusted beacon (as needed for the Fiat-Shamir argument).

Yair Frankel, Yiannis Tsiounis and Moti Yung [10] suggested a notion of "Indirect Discourse Proofs" with which one can prove indirectly yet efficiently that a third party has a certain future capability. Employing this idea they present the concept of "Fair Off-Line e-Cash" (FOLC) system.

In this paper, we present a new untraceable electronic cash protocol. The basic model of our protocol is as similar as in [10]. We use the cut-and-choose methodology and the security of our system is based on Discrete Logarithm Assumption. Our withdrawal, payment and deposit protocols are much more simpler than those in [5].

3 Basic Model

Electronic cash (in particular off-line untraceable electronic cash) has sparked wide interest among cryptographers ([3, 4, 9, 10, 11], etc.). In its simplest form, an e-cash system consists of three parties (a bank B, a user U and a shop S) and three main procedures as shown in figure 1 (withdrawal, payment and deposit). In a coin's life-cycle, the user U first performs an account establishment protocol to open an account with bank B. To obtain a coin U performs a withdrawal protocol with B and during a purchase U spends a coin by participating in a payment protocol with the shop S. To deposit a coin, S performs a deposit protocol with the bank B.

Users and shops maintain an account with the bank, while

1. U withdraws electronic coins from his account, by performing a withdrawal protocol with bank B over an authenticated channel.
2. U spends a coin by participating in a payment protocol with a shop S over an anonymous channel, and
3. S performs a deposit protocol with the bank B, to deposit the user's coin into his account.

The system is *off-line* if during payment the shop S does not communicate with the bank B. It is *untraceable* if p.p.t. TM (probabilistic polynomial-time

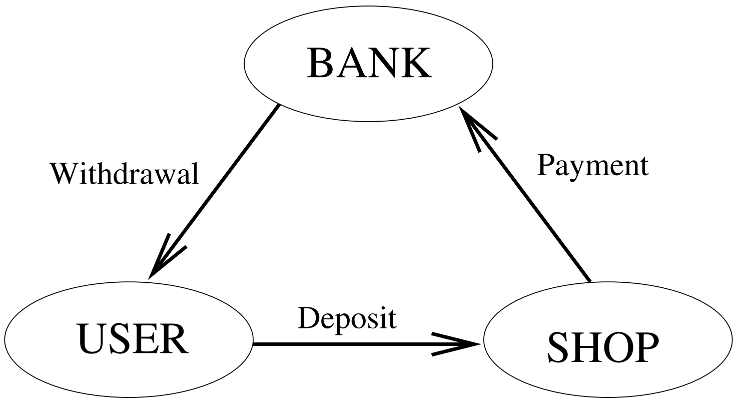


Fig. 1. model of electronic-cash

Turing Machine) M access to all bank's views of withdrawal, payment and deposit protocols. Then for any two coins C_i, C_j and withdrawals W_0, W_1, such that C_i, C_j are the coins originating from W_0, W_1, M cannot distinguish non-negligibly better (in n) than randomly guess whether C_i, C_j came from W_0, W_1. It is *anonymous* if the bank B, in collaboration with the shop S, cannot trace the coin to the user. However, in the absence of tamper-proof hardware, electronic coins can be copied and spent multiple times by the user U. This has been traditionally referred to as double-spending. In anonymous on-line e-cash, double-spending is prevented by having the bank check if the coin has been deposited before. In off-line anonymous e-cash, however, this solution is not possible; instead, as proposed by Chaum et. al. [2], the system guarantees that if a coin is double-spent the user's identity is revealed with overwhelming probability.

There are also three additional protocols such as the bank setup, the shop setup, and the user setup (account opening). They describe the system initialization, namely creation and posting of public keys and opening of bank accounts. Although they are certainly parts of a complete system, these protocols are often omitted as their functionalities can be easily inferred from the description of the three main procedures. For clarity we will only describe the bank setup and the user setup (because the shop setup is as similar as user setup) for our new scheme in the next section.

4 New Off-Line Untraceable Electronic Cash Scheme

In this section, we propose a new off-line untraceable electronic cash scheme.

Our scheme includes two system initialization protocols (bank setup protocol and user setup protocol) and three basic protocols: a new withdrawal protocol with which U withdraws electronic coins from B while his account is debited, a new payment protocol with which U pays the coin to R, and a new deposit protocol with which R deposits the coin to B and has his account credited.

4.1 System Initialization

We only describe the bank setup and the user setup based on Discrete Logarithm Assumption and random-oracle model here and omit the shop setup (because the shop setup is similar to the user setup).

Bank's setup protocol: (performed once by B)
Primes p and q are chosen such that $|p-1| = \delta + k$ for a specified constant δ , and $p = \gamma q + 1$, for a specified small integer γ. Then a unique subgroup G_q of prime order q of the multiplicative group Z_p and generators g, g_1, g_2 of G_q are defined. Secret key $X_B \in_R Z_q$ for a denomination is created. Hash function H from a family of collision in tractable (or, ideally, according to [10], correlation-free one way) hash function is also defined. B publishes p, q, g, g_1, g_2, H and its public keys $h = g^{X_B} (\text{mod } p)$, $h_1 = g_1^{X_B} (\text{mod } p)$, $h_2 = g_2^{X_B} (\text{mod } p)$.

User's setup (account opening) protocol: (performed for each user U)
The bank B associates the user U with $I = g_1^{u_1} (\text{mod } p)$ where $u_1 \in G_q$ is generated by U and $g_1^{u_1} g_2 \;/\!= 1 (\text{mod } p)$. U computes $z = h_1^{u_1} h_2 = (I g_2)^{X_B} (\text{mod } p)$.

After the user's account and the shop's account opening, we can describe the new untraceable electronic cash scheme.

4.2 New Untraceable Electronic Cash Scheme

We now describe the new off-line untraceable electronic cash scheme which includes three protocols: withdrawal protocol, payment protocol and deposit protocol.

Withdrawal protocol:(over an authenticated channel between B and U)
The withdrawal protocol creates a "restrictively blind" signature [10] of I and using cut-and-choose technology. U will signature on $(I g_2)^s$ where s is a random number (chosen by U and kept secret). Let $\oplus$ denote bitwise exclusive or and $||$ denote concatenation.

1. The user chooses $a_i, c_i, 1 \leq i \leq k$, independently and uniformly at random form the residues (mod p).
2. The user forms and sends to the bank k blinded candidates
$$B_i = H(x_i, y_i) \ (mod p) \ \ 1 \leq i \leq k$$
where
$$x_i = g^{a_i} (mod\ p), \ y_i = g_1^{a_i \oplus (I || c_i)} (mod\ p).$$
3. The bank chooses a random subset of $k/2$ blinded candidate indices
$$R = \{i_j\}, \ 1 \leq i_j \leq k, \ 1 \leq j \leq k/2$$
and transmits it to the user.
4. The user transmit $A = (I g_2)^s (mod\ p)$ and $z' = z^s (mod\ p)$ to bank.
5. The user displays a_i, c_i values for all i in R, and the bank checks them. To simplify notation we assume that $R = \{k/2+1, k/2+2, ..., k\}$

6. The bank verifies:

$$A^x = z^{'} (mod\ p)$$

   and gives the user

$$\prod_{i \notin R} B_i = \prod_{1 \leq i \leq k/2} B_i (mod\ p)$$

7. The user extracts the electronic coin

$$C = \prod_{1 \leq i \leq k/2} B_i (mod\ p)$$

Payment protocol: (performed between U and R over an anonymous channel) At payment time U supplies information to the receiver R (which is later forwarded to the bank) so that if a coin is double-spent the user U is identified. The payment protocol (U and R agree on date/time):

1. The user sends C to R.
2. R chooses a random binary string $z_1, z_2, ..., z_{k/2}$, and sends to the user.
3. The user responds as follows, for all $1 \leq i \leq k/2$:
   a. If $z_i = 1$, then sends R: a_i, y_i
   b. If $z_i = 0$, then sends R: $x_i, a_i \oplus (I||c_i)$
4. R verifies that C is of the proper form and that the user's responses fit C.
5. R later sends C and the user's responses to the bank which verifies their correctness and credits his account.

Deposit protocol:(The receiver deposit a coin to bank)
The receiver deposits the coin in an account by providing the bank with a transcript of the payment. If the user uses the same coin C twice, then the user has a high probability of being traced: with high probability, two different receivers will send complementary binary values for at least one bit z_i for which B_i was of the proper form. The bank can easily search its records to ensure that C has not been used before. If the user uses C twice, then with high probability, the bank has both a_i and $a_i \oplus (I||c_i)$. Thus, the bank can isolate the user and trace the payment to the user's account.

The relationship between various protocols can be shown in figure 2 which is different from figure 1. figure 1 is only a purchase procedure and in figure 2 the system initialization is added, which includes the security random oracle model and how to get the bank setup and the user setup. It is important for the withdrawal protocol, payment protocol and deposit protocol in our new scheme. The security of our new scheme is aslo based on the system initialization.

We have shown how to derive an efficient variant based on the random-oracle model in the bank setup and the user setup. It achieves provable security based on DLA and the existence of random oracle-like hash functions. Based on this system initialization, three new protocols with cut-and-choose methodology are designed. It is much more secure for the random-oracle model and cut-and-choose methodology used and it is much simpler in our three new basic protocols (withdrawal protocol, payment protocol and deposit protocol).

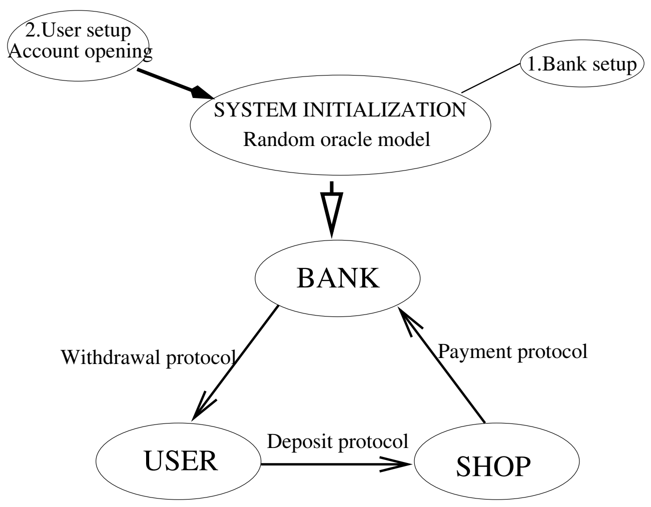


Fig. 2. Basic off-line electronic cash system

5 Security

We employ Discrete Logarithm methods; these methods have been suggested in many of the recent e-cash schemes to bind identities (an unavoidable issue in off-line e-cash). These methods were started in [6] and continued by others [8,9] as well as in [11]. The security of our protocols are based on the hardness of Discrete Logarithms [12] and the cut-and-choose technology. The cut-and-choose technology is based on zero-knowledge proof, and the scheme assumes that the hash function used is perfect (i.e., random oracles) [10].

A possible problem with the scheme is a collusion between a user U and the second shopkeeper. After having user transactions with two receivers which send the same information to the bank, the bank knows that with high probability one of them is lying, and the bank can decide the first purchase is right by the date/time in the payment but cannot trace the coin to the user's account.

To prevent the bank frame the user as a multiple spender in the scheme, we use digital signature Z^s for s is known only by the user. The user is protected against frame-up only computationally, not unconditionally. The User does not expect the signature scheme can be broken.

6 Conclusion

In this paper an untraceable electronic cash scheme is designed which is an off-line scheme and achieves provable security without the use of general computation protocols and without the requirement of a trusted party, or of a costly general computation-based initialization procedure. We have shown how to derive an efficient variant based on the random-oracle model; this variant thus achieves provable security based on DLA and the existence of random oracle-like hash functions, while its efficiency is on par with heuristic cut-and-choose based systems. The security of the system is based on DLA (Discrete Logarithm Assumption) and the cut-and-choose methodology.

References

1. D. Chaum. Blind signatures for untraceable payments. Advances in Cryptology. Proc. Crypto'82, pp:199-203.
2. D. Chaum, A. Fiat, and M. Naor. Un traceable electronic cash. In Advances in Cryptology Crypto'88, pp:319-327. Springer-Verlag, 1990.
3. D. Simon. Anonymous communication and anonymous cash. Advances in Cryptology, Proc. of Crypto'96, pp:61-73, Springer-Verlag, 1991.
4. Y. Yacobi. Efficien telectronic money. Advances in Cryptology, Proc. of Asiacrypt '94, pp: 153-163. Springer-Verlag, 1995.
5. T. Okamoto and K. Ohta. Disposable zero-kno wledge authen tication and their applications to untraceable electronic cash. Advances in Cryptology, Proc. of Crypto '89, pp:481-496, Springer-Verlag, 1990.
6. B. Pfitzmann and M. W aidner. How to break and repair a 'provably secure' untraceable payment system. Advances in Cryptology, Proc. of Crypto'91, pp:338-350, Springer-Verlag, 1992.
7. M. Franklin and M. Yung. Secure and efficient off-line digital money. In Proc. of the twentieth International Colloquium on Automata, Languages and Programming(ICALP 1993). (Lecture Notes in Computer Science 700), pp: 265-276. Springer-Verlag, 1993.
8. A. Chan, Y. Frankel, and Y. Tsiounis. An efficient off-line electronic cash scheme as secure as RSA. Research report NU-CCS-96-03, Northeastern University, Boston, Massachussets, 1995.
9. T. Okamoto. An efficient divisible electronic cash scheme. Advances in Cryptology. Proc. of Crypto'95, pp:438-451. Springer-Verlag, 1995.
10. Y. Frankel, T. Yiannis and M. Yung. Indirect Discourse Proofs: Achieving Fair Off-Line Electronic Cash. Asiacrypt'96, Lecture Notes in Computer Science 1163, pp: 286-300.
11. T. Yiannis. Fair Off-Line Cash made easy. Asiacrypt'98, Lecture Notes in Computer Science 1346, pp:240-252.
12. T. Yiannis, M. Yung. On the security of ElGamal-based encryption. In 1998 International Workshop on Practice and Theory in Public Key Cryptography (PKC '98), Japan, pp:37-49.
13. M. Bellare and P. Roga way. Random oracles are practical: a paradigm for designing efficient protocols. In First ACM Conference on Computer and Communications Security, pp:62-73, 1993.
14. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21, 2 (Feb. 1978), pp:120–126.

Materializing Web Data for OLAP and DSS

Yan Zhu, Christof Bornhövd, Doris Sautner, and Alejandro P. Buchmann

Department of Computer Science
Darmstadt University of Technology
D-64283 Darmstadt, Germany
{zhu,bornhoev,buchmann}@dvs1.informatik.tu-darmstadt.de
doris.sautner@gmx.de

Abstract. Business decisions must rely not only on company-internal data but also on external data from competitors or relevant events. This information can be obtained from the WWW but must be integrated with the data in a company's data warehouse. In this paper we discuss a system architecture for warehousing Web content for OLAP and DSS. A self-describing object model is used to make the implicit modeling and context assumptions explicit, both for the data obtained from the Web and the data already in the data warehouse. A domain-specific ontology provides a common interpretation basis for data and metadata. We propose an object-relational mapping that takes into consideration the peculiarities of relational data warehouses based on a star schema and propose a mapping rule language to describe the necessary transformation rules. The system framework described in this paper has been implemented in Java.

1 Introduction

Web warehousing is a novel and very active research area, which combines two rapidly developing technologies – data warehousing and Web technology. It provides a suitable approach to systematically discover and acquire strategic information from the Web. This information may be identified, catalogued, managed and then accessed by the end users [22], via search engines or some Web information management system [1,2,4,10,16]. Web data may be extracted, integrated, represented and materialized in a data warehouse for OLAP and DSS [6,17,24], and also may be managed by using a hybrid approach for taking advantage of the data warehousing and the virtual integration approach [3,21,23,26].

External data is important for carrying out meaningful OLAP in the context of e-commerce. For example, an online book shop manager may make use of integrating discount book information or information about new publications from other online book shops in his data warehouse, in order to analyze market trends and make new business plans. As Web technology develops, a huge amount of strategic information is already offered on the Web. Integrating and managing e-commerce data and external data in a data warehouse, implementing OLAP on them, and mining historic data to generate summary information and business rules will benefit business. However, when warehousing Web data we face three main challenges:

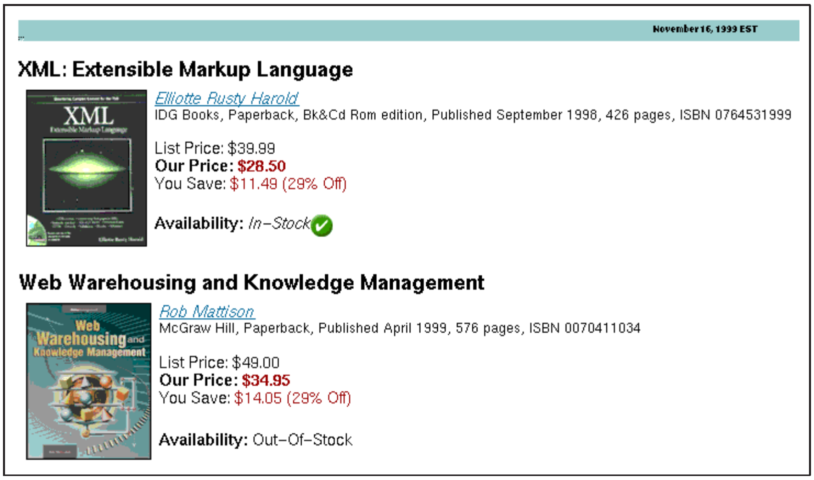
- **Data Extraction.** Web sources are dynamic and autonomous. Not only is Web-based data updated frequently but Web sources are dynamically developed as well. Every day new sources are brought on the Internet and already available sources may be changed drastically or even disappear. In addition, providers change the presentation and contents of their sources to react to new developments or to hinder automatic, exhaustive extraction of their data. All this makes determining and automatically extracting relevant data from the Web a difficult task.
- **Preparation/Integration.** E-commerce data comes from daily transaction processing on the Internet, which in most cases is in structured form. External data comes directly from some Web sources and in general is in unstructured or semistructured form. In addition, the sources available online are developed and managed by independent institutions. They represent the same or related information on the basis of different assumptions about their organization and meaning. This is because of different political and cultural contexts, or because of different intentions concerning the use of the data. Therefore, before loading data from different Web sources into the data warehouse, we have to resolve structural and semantic heterogeneity among them and between them and the data already in the warehouse.
- **Presentation/Materialization.** The paradigm of the Web is totally different from the paradigm of a data warehouse. Most data in the Web is in semistructured or unstructured form, i.e., is available as HTML or XML data or irregularly structured files, which may change frequently. In contrast, a data warehouse in general is based on a well-structured data model, in most cases a relational data model, and is comparably stable. Thus, we have to map irregularly structured, maybe frequently changing data to a predefined, restrictive database schema.

In order to tackle these challenges, we have proposed a framework for warehousing Web information [26]. We represented Web data using a special data model called MIX (***M**etadata based **I**ntegration model for data **X**-change*)[7,8]. This model represents data together with a description of their underlying interpretation context and uses domain-specific ontologies in order to enable a semantically correct interpretation of the available data and metadata. Thus, it supports the integration of Web-based information. In our previous work [8,9] some aspects of Web data preparation/integration are already discussed. In this paper we concentrate on problems of representation/materialization of Web data. The main contributions of this paper are:

- We implemented a system framework and designed a Transformation Processor to accomplish transformation from the object-oriented MIX model to a relational star schema in a data warehouse. In the Transformation Processor, an object/relational mapping approach is adopted. The object identity, complex objects and the multi-valued attribute problem are discussed in this context.
- We defined a Mapping Rule Language to specify the correspondences between MIX objects and tables in the star schema of a given data warehouse.

The rest of the paper is organized as follows: Section 2 gives a motivating example. The system architecture and the mediated data model are introduced in Section 3. Section 4 analyzes the transformation process from Web objects to the star schema of the data warehouse. The mapping rule language is explained in Section 5. Section 6 presents the related work. Section 7 provides conclusions and identifies areas of ongoing and future research.

2 A Motivating Example


November 16, 1999 EST

XML: Extensible Markup Language

Elliotte Rusty Harold
IDG Books, Paperback, Bk&Cd Rom edition, Published September 1998, 426 pages, ISBN 0764531999

List Price: $39.99
Our Price: $28.50
You Save: $11.49 (29% Off)

Availability: *In-Stock*

Web Warehousing and Knowledge Management

Rob Mattison
McGraw Hill, Paperback, Published April 1999, 576 pages, ISBN 0070411034

List Price: $49.00
Our Price: $34.95
You Save: $14.05 (29% Off)

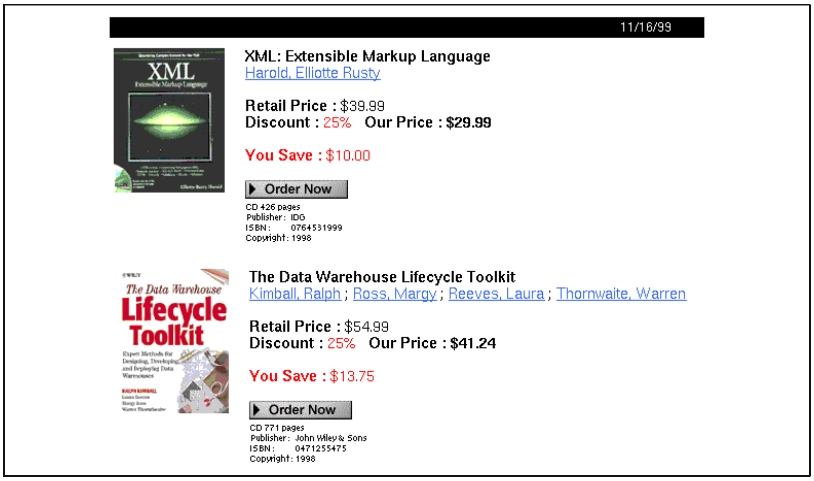
Availability: Out-Of-Stock

Fig. 1. Discount Books Information from Source A

Online book shopping is a very active e-commerce area. A large amount of customer orders is produced every day, recorded in a data warehouse for OLAP and decision making. In addition, the book shop manager may also integrate discount book information from his competitors in this data warehouse, in order to compare pricing schemes, analyze market trends and make new business plans.This information can be obtained from the related Web pages. Figure 1 and 2 show discount book information from two different online providers given as HTML pages. In Figure 3, the star schema of a simplified bookshop data warehouse is defined on the basis of the relational data model.

3 The Framework for Managing Web Data

We have implemented a Java framework that provides a platform for integrating Web data and mapping it to a data warehouse. The framework, shown in Figure 4, follows the classical mediator approach introduced in [25].



11/16/99

XML: Extensible Markup Language
Harold, Elliotte Rusty

Retail Price : $39.99
Discount : 25% Our Price : $29.99

You Save : $10.00

Order Now

CD 426 pages
Publisher: IDG
ISBN: 0764531999
Copyright: 1998

The Data Warehouse Lifecycle Toolkit
Kimball, Ralph ; Ross, Margy ; Reeves, Laura ; Thornwaite, Warren

Retail Price : $54.99
Discount : 25% Our Price : $41.24

You Save : $13.75

Order Now

CD 771 pages
Publisher: John Wiley & Sons
ISBN: 0471255475
Copyright: 1998

Fig. 2. Discount Books Information from Source B

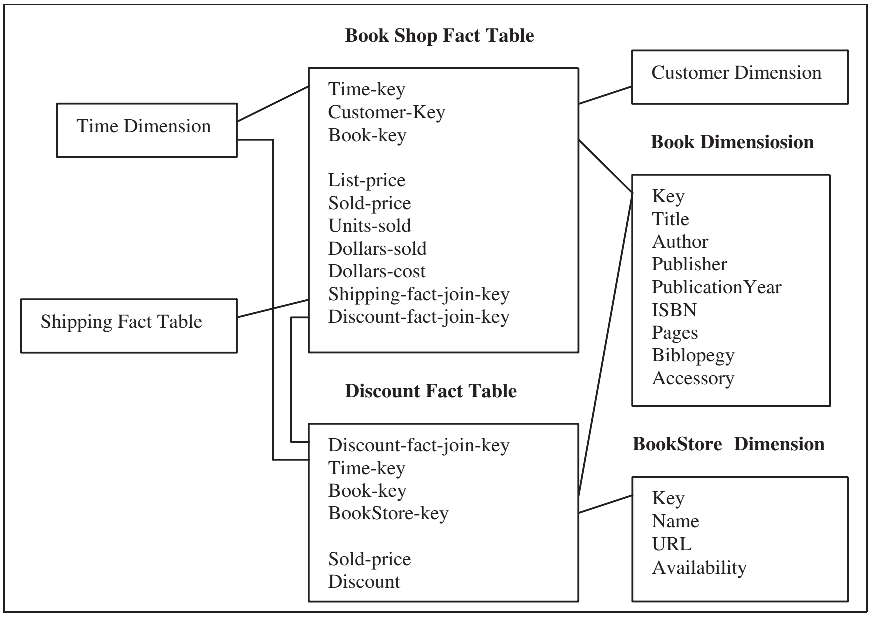


Fig. 3. Star Schema of the Online Bookshop Data Warehouse

Wrappers are used to extract relevant data from available data sources, to map it to MIX based on a common structural context and to return the MIX objects to the Federation Manager.

The ***Federation Manager*** manages the available data sources by keeping a Metadata Repository. Based on the explicit description of the underlying context the Federation Manager tries to integrate heterogeneous data by converting the data to a common semantic context.

An ***Ontology Server*** stores and manages the domain-specific vocabulary, which in our framework describes a single domain and provides a way to describe the concepts in that ontology independent of any application or data source.

The integrated data is given to the ***Transformation Processor***, which maps the MIX objects to an existing star schema, and loads the data into the data warehouse. It uses the mapping rules specified in the corresponding mapping file. The mapping rules are written using a special Mapping Rule Language (MRL). ***Function rules*** are specified as Java methods stored in the ***Mapping Functions Library*** and can be called by the Transformation Processor. The transformation process is described in more detail in Section 4.

The ***Incremental Maintenance Processor*** serves as the maintenance component of the data warehouse, when Web data is updated. In this processor a copy of the latest MIX objects is kept. The processor reads updated MIX objects, compares them with the old copies and calculates the incremental parts. These are sent to the Transformation Processor, where they are mapped to the star schema of the data warehouse.

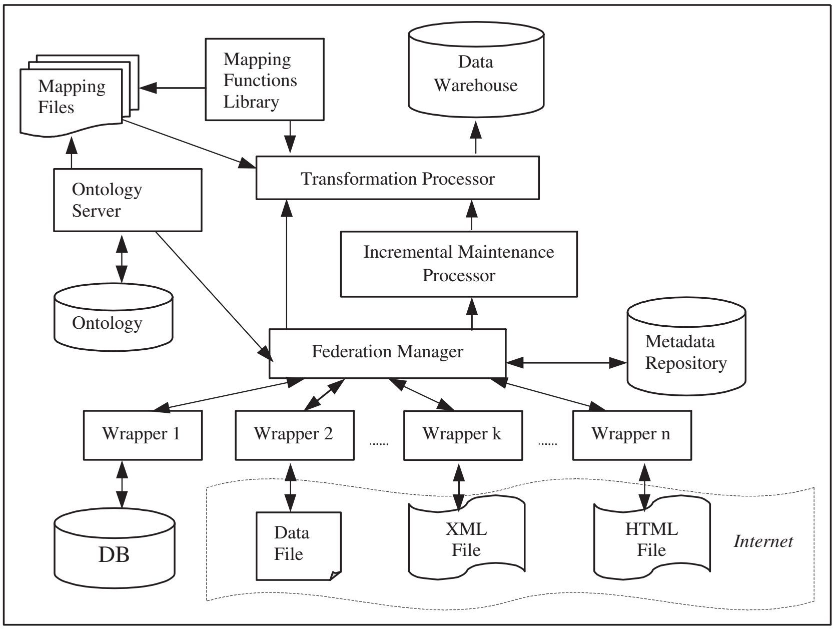


Fig. 4. System Architecture for Warehousing Web Data

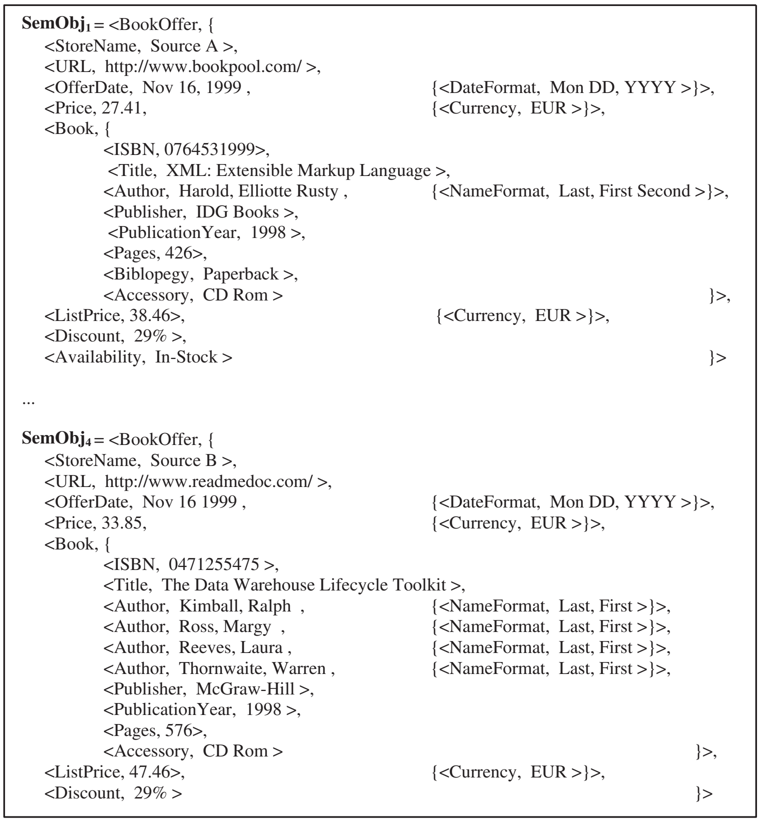
MIX is a self-describing data model and is based on the concept of a semantic object. A simple semantic object is represented as a triple of the form:

$< ontology\ concept, value, \{semantic\ context\} >$, while a complex semantic object can be represented as: $< ontology\ concept, set\ of\ semantic\ objects >$.

For example, the data given in Figure 1 and 2 can be represented as MIX objects of concept *BookOffer* based on a common structural and semantic context. In our example, we assume dates represented in the form *"Mon DD, YYYY"*, author names given by *"Last Name, First Name Second Name"*, and prices specified in EUR. Figure 5 shows the representation of the integrated data.

4 Transforming MIX Objects to Warehouse Tables

MIX is an object model, while the data warehouse in our framework is based on a star schema. Therefore, mapping MIX objects to the data warehouse tables is comparable to known object/relational mappings as discussed for example in [5,13,18,19,20].



```
SemObj1 = <BookOffer, {
    <StoreName,  Source A >,
    <URL,  http://www.bookpool.com/ >,
    <OfferDate,  Nov 16, 1999 ,                  {<DateFormat,  Mon DD, YYYY >}>,
    <Price, 27.41,                               {<Currency,  EUR >}>,
    <Book, {
            <ISBN, 0764531999>,
             <Title,  XML: Extensible Markup Language >,
            <Author,  Harold, Elliotte Rusty ,   {<NameFormat,  Last, First Second >}>,
            <Publisher,  IDG Books >,
             <PublicationYear,  1998 >,
            <Pages, 426>,
            <Biblopegy,  Paperback >,
            <Accessory,  CD Rom >                                              }>,
    <ListPrice, 38.46>,                           {<Currency,  EUR >}>,
    <Discount,  29% >,
    <Availability,  In-Stock >                                                 }>

...

SemObj4 = <BookOffer, {
    <StoreName,  Source B >,
    <URL,  http://www.readmedoc.com/ >,
    <OfferDate,  Nov 16 1999 ,                   {<DateFormat,  Mon DD, YYYY >}>,
    <Price, 33.85,                               {<Currency,  EUR >}>,
    <Book, {
            <ISBN,  0471255475 >,
            <Title,  The Data Warehouse Lifecycle Toolkit >,
            <Author,  Kimball, Ralph  ,          {<NameFormat,  Last, First >}>,
            <Author,  Ross, Margy  ,             {<NameFormat,  Last, First >}>,
            <Author,  Reeves, Laura ,            {<NameFormat,  Last, First >}>,
            <Author,  Thornwaite, Warren ,       {<NameFormat,  Last, First >}>,
            <Publisher,  McGraw-Hill >,
            <PublicationYear,  1998 >,
            <Pages, 576>,
            <Accessory,  CD Rom >                                            }>,
    <ListPrice, 47.46>,                          {<Currency,  EUR >}>,
    <Discount,  29% >                                                        }>
```

Fig. 5. Integrated Web Data in MIX Representation

However, different to those approaches, data about a real world entity is divided into facts in fact tables and descriptive attributes in dimension tables in a star schema, i.e., is spread across different tables, and the dimension tables must not be normalized. In addition, MIX provides a flexible data model for representing irregularly structured data. Semantic objects of the same concept do not necessarily have the same structure, i.e., set of attributes. For example, in Figure 5 $SemObj_1$ provides information about the availability of a book which is not given in $SemObj_4$. This must be considered when transforming MIX objects into tuples of the star schema. Finally, in contrast to most object/relational mapping approaches which create the relational schema according to the given object classes, we map to an existing data schema in the warehouse.

4.1 Basic Mapping Rules

In our mapping approach we obey the following basic rules:

1. **Aggregations.** In object/relational mapping, aggregations can be implemented using two approaches: single table aggregation or foreign key aggregation [5,19,20]. In the data warehousing context we have to distinguish two cases. When aggregation data populates a dimension table, the single table aggregation approach is favorable because of query execution performance. Thus, we follow this approach. However, its disadvantage is latent long tuple width. The foreign key approach on the other hand might snowflake the data warehouse schema, and lower query performance.
   In the case that an aggregate populates a fact table as well as dimension tables, we must decompose an aggregate and separately map subobjects of a MIX object to facts in the fact table and attributes in dimension tables.
2. **Inheritance hierarchy of concepts.** The MIX model supports class inheritance, through the inheritance of concepts in the domain ontology. However, since we have to map MIX objects to an existing data schema we see them only as objects of a concrete concept/class. Therefore, in the mapping we need not consider the inheritance relationships of concepts/classes.
3. **Object/subobject relationships between MIX objects.** In the domain ontology only the relationships between concepts for complex semantic objects and their identifying attributes are specified. Additional, non-identifying attributes may vary between objects of the same ontology concept. However, each MIX object contains concrete attributes. Thus, we have to take possible combinations of attributes of complex semantic objects of a given concept into consideration in our mapping. We achieve this by giving a mapping for all attributes of a class of objects that are of interest for our application.
4. **Handling keys.** In the MIX model each object is identified through single or multipart key attributes, similar to the relational model. When an object is implemented in Java, a unique object ID can be automatically assigned. When a dimensional table is constructed, an additional column can be designed in this table as primary key. One unique key value for each row can

be generated by the system. In the star schema of a data warehouse, the primary key of fact tables is the combination of the foreign keys that link to the corresponding dimension tables. The generation of the foreign keys is involved with connecting fact tables and dimension tables. We discuss this rule in 9) in more detail.

5. **One MIX concept to one/many relational attributes.** When mapping an ontology concept for a simple semantic object to table columns we must distinguish three cases. In the easiest case we have a direct 1:1 mapping, i.e., the values of a concept are directly assigned to the corresponding table column. In the second case, we must calculate the values of a column by applying a specified function on the corresponding concept. Finally, when values of an ontology concept must be decomposed to multiple table columns, decomposition functions have to be applied to calculate suitable values.
   Calculation rules are specified as Java methods in our framework and stored in the Mapping Function Library. They are called by the Transformation Processor. For example, objects of concept *OfferDate* in Figure 5 specify the date of a given book offer according to the format "Mon DD YYYY". In the Time Dimension of the data warehouse in Figure 3, we have attributes Day, Month and Year. Therefore, we must use decomposition functions, like: F_{Day} *("Nov 16, 1999")* $\Rightarrow$ *16*, to generate values for these columns.
6. **Many MIX concepts to one relational attribute.** When multiple concepts are mapped to one column, i.e., values from multiple attributes are used to derive a column value, an aggregation function has to be applied to calculate the value. For example, calculating the *TotalCost* paid by a customer in one book purchase is involved with MIX objects of concept *Subtotal* and *ShippingCost*. The corresponding calculation function is given by: $F_{TotalCost}(\ \mathit{Subtotal}, \mathit{ShippingCost}\ ) = \mathit{Subtotal} + \mathit{ShippingCost} \ \Rightarrow\ TotalCost$.
7. **Multiple values problem.** When different subobjects of the same concept occur, a multi-valued attribute problem arises. For example, a book may have more than one author. Most relational database systems do not support multi-valued attributes. If the maximal cardinality of such multi-valued attributes is known in advance, multiple columns can be used to store them. Otherwise, separate rows have to be used to store them. Since we map to an existing star schema, we must adapt the mapping to the given table structure. For example, multiple authors of SemObj_4 shown in Figure 5, must be mapped to multiple rows to store each author name.
8. **Default values.** When we map MIX objects to the tables of the data warehouse, we sometimes do not have values for all attributes. It would be necessary to use some default value in these places. For example, SemObj_4 shown in Figure 5 has no *Availability* item. In the transformation processing, we will use "Unknown" as the value of *Availability* in a tuple. We can suggest some default values as standard DefaultValue, such as "Null" or "Unknown".
9. **Connecting dimension tables to fact tables.** To connect dimension tables to fact tables, two cases must be considered. In one case, if a new tuple of a dimension table is populated and the primary key value is generated by

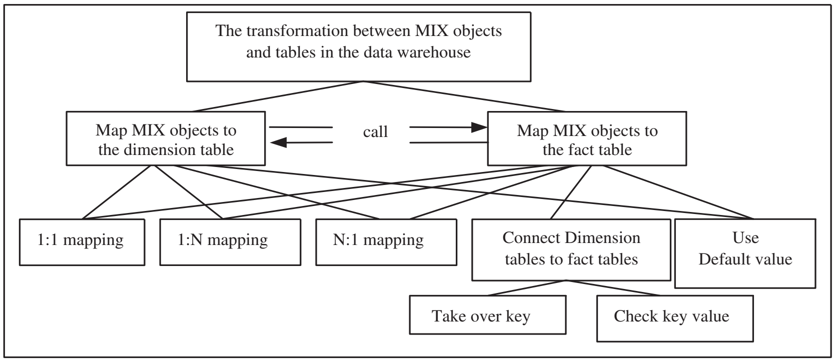

The transformation between MIX objects and tables in the data warehouse
Map MIX objects to the dimension table
call
Map MIX objects to the fact table
1:1 mapping
1:N mapping
N:1 mapping
Connect Dimension tables to fact tables
Use Default value
Take over key
Check key value

Fig. 6. Several Cases of the Transformation

the system, we can register this value in the corresponding tuple of the fact table as one foreign key value. **Example a** in Section 5 will describe this transformation. In another case, a dimension table already exists and all key values in this table are already generated before we link dimension tables to a fact table. To connect such dimension tables to the fact table we find the required key value in the dimension table and then take it over in the fact table. **Example b** in Section 5 will explain this situation.

The transformation cases are classified in Figure 6.

4.2 Transformation Processing

In the Transformation Processor shown in Figure 7, the Receiver reads MIX objects from the Federation Manager or from the Incremental Maintenance Processor and puts them in the Vector of MIX objects. The Parser reads mapping rules from mapping files that are written in MRL and parses rule documents. The parsed results are stored as a list in the ListStructure. The Mapper reads mapping rules from the ListStructure and uses mapping functions from the Mapping Function Library. Under the guide of the given rules, the Mapper transforms MIX objects to facts and attributes in the data warehouse. The data is loaded through a JDBC driver into the data warehouse.

5 Mapping Rule Language (MRL)

We defined a simple, descriptive language (MRL) to describe the mapping rules, which are specified in mapping rule files. Figure 8 gives an overview of the syntax of MRL. The following examples will explain the usage of MRL in more detail.

1. **MIX concepts to dimension tables**
   Considering the data warehouse in Figure 3 and MIX objects in Figure 5,

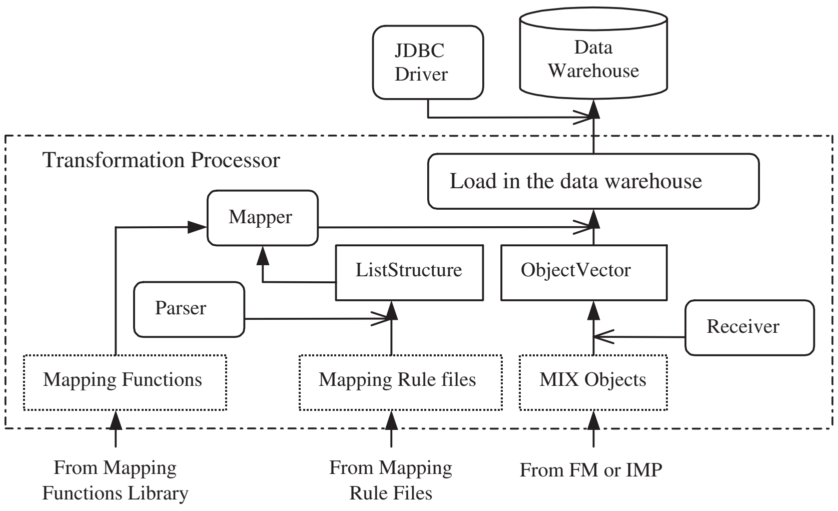


Fig. 7. The Functional Diagram of the Transformation Processor

BookOffer is a concept of MIX, while *BookStore* is the name of a dimension table. Mapping rules between MIX objects and dimension tables can be written as follows:

ClassToDimensionTable(BookOffer, BookStore)
  Key generated by the system
  Ontology.BookOffer.StoreName : BookStore.Name
  Ontology.BookOffer.URL : BookStore.URL
  Ontology.BookOffer.Availability : BookStore.Availability }

2. **MIX concepts to fact tables**
   Mapping rules between MIX objects and fact tables can be written as follows:

   ClassToFactTable(BookOffer, Discount, Time, Book, BookStore) {
     Discount-fact-join-key generated by system
     DiscountFactLinkTimeDimension : Discount.Time-key
     DiscountFactLinkBookDimension : Discount.Book-key
     DiscountFactLinkBookStoreDimension : Discount.BookStore-key
     Ontology.BookOffer.Price : Discount.Sold-Price
     Ontology.BookOffer.Discount : Discount.Discount }

3. **Connecting dimension tables to fact tables**
   Example a: By populating the *BookShop* Fact table, *Time* Dimension, *Customer* Dimension and *Book* Dimension tables using data about books purchase, after a new key value in a dimension table is just generated, we can take it from the dimension tables and assign it to the corresponding foreign key in the fact table.

DefaultValue: xxx

ClassToDimensionTable (Concept, Tables)-Block

- Generation of the Key
- 1:1-Mapping
  Ontology.Concept$_x$: ColumnName$_x$
- N:1-Mapping
  F_{agg}(Ontology.Concept$_1$, ..., Ontology.Concept$_n$) : ColumnName$_x$
- 1:N-Mapping
  F_{dec}(Ontology.Concept $_x$) : ColumnName$_1$, ColumnName$_2$, ..., ColumnName$_n$
- Call ClassToFactTable-Block (optional)

ClassToFactTable(Concept, Tables)-Block

- The generation of fact-join-key (optional)
- FactLinkDimension-Block : Key columns
- 1:1-Mapping
  Ontology.Concept$_x$: ColumnName$_x$
- N:1-Mapping
  F_{agg}(Ontology.Concept$_1$, ..., Ontology.Concept$_n$) : ColumnName$_x$
- 1:N-Mapping
  F_{dec}(Ontology.Concept $_x$) : ColumnName$_1$, ColumnName$_2$, ..., ColumnName$_n$
- Call ClassToDimensionTable-Block (optional)

FactLinkDimension({Concept,} Fact, Dimensions)-Block

- DimensionTableKey$_x$: FactTableKey$_x$
- Check (Ontology.Concept$_x$)
- FindKeyValue (DimensionTableAttributes): FactTableKey$_x$

Fig. 8. Overview of the Syntax of MRL

BookShopFactLinkallDimension(BookShop, Time, Book, Customer) {
Time.Key : BookShop.Time-key
Book.Key : BookShop.Book-key
Customer.Key : BookShop.Customer-key }

Example b: Considering the data warehouse shown in Figure 3, *Time* Dimension based on company's internal data already exists before the *Discount* Fact Table is populated with external Web data. We assume that attributes *Day*, *Month* and *Year* in the *Time* Dimension are continuous, since the books hop receives the order from customers every day. But his competitors announce their new discount book information not at every day, therefore the update date of the discount book information is not continuous. In order to connect the *Time* Dimension to the *Discount* Fact Table, we first find the time key values in those tuples in the *Time* Dimension, whose values of *Day*, *Month*, *Year* equal the *OfferDate* in the *BookOffer* object, then we write these key values in the *Discount* Fact table.

```
DiscountFactLinkTimeDimension( BookOffer, Discount, Time ) {
Check( BookOffer.OfferDate ) {
   GetDayfromDate( Ontology.BookOffer.OfferDate ) : Time.Day
   GetMonthfromDate( Ontology.BookOffer.OfferDate ) : Time.Month
   GetYearfromDate( Ontology.BookOffer.OfferDate ) : Time.Year
   FindKeyValue( Time.Day, Time.Month, Time.Year ) : Time.key
}
Time.key : Discount.Time-key }
```

The following complete example shows the mapping of a semantic object from Figure 5 to tables in Figure 3.

```
ClassToDimensionTable( BookOffer, BookStore ) {
   DefaultValue : "Unknown"
   Key generated by the system
   Ontology.BookOffer.StoreName : BookStore.Name
   Ontology.BookOffer.URL : BookStore.URL
   ClassToFactTable( BookOffer, Discount, Time, Book, BookStore ) {
      Discount-fact-join-key generated by system
      DiscountFactLinkTimeDimension( BookOffer, Discount, Time ) {
         Check( BookOffer.OfferDate ) {
         GetDayfromDate( Ontology.BookOffer.OfferDate ) : Time.Day
         GetMonthfromDate( Ontology.BookOffer.OfferDate ) : Time.Month
         GetYearfromDate( Ontology.BookOfferr.OfferDate ) : Time.Year
         FindKeyValue( Time.Day, Time.Month, Time.Year ) : Time.key }
      Time.key: Discount.Time-key }
   Ontology.BookOffer.Price:Discount.Sold-Price
   DiscountFactLinkBookDimension( BookOffer, Discount, Book ) {
      Check( BookOffer.Book.ISBN ) {
         FindKeyValue( Book.ISBN ) : Book.Key }
      Book.Key : Discount.Book-key }
   BookStore.Key : Discount.BookStore-key
   Ontology.BookOffer.Discount : Discount.Discount }
If( Ontology.BookOffer.Availability <> NULL )
   Ontology.BookOffer.Availability : BookStore.Availability
Else DefaultValue : BookStore.Availability }
```

6 Related Work

Our system is designed to extract, integrate and transfer Web data into a data warehouse and to interoperate with conventional warehouse data for carrying out meaningful OLAP and supporting the decision making process.

The WHOWEDA system [6,24] is a Web warehouse that materializes and manages useful information from the Web. Web objects, Web schema and a Web algebra are described by a so-called *Web Information Coupling Model* (WICM). They define a Node type and a Link type to refer to the coupled Web information (HTML or plain text documents, Hyperlinks) and materialize the Web objects in a set of connected, directed graphs. Their focal points are to describe the topological structure of the Web and to design a Web algebra. Our approach differs from WHOWEDA in that a metadata-based object model is used to describe Web information content rather than Web structure. This model is used as the basis to prepare and integrate the data into the Star Schema of a relational data warehouse. Some advantages are that relational data warehouses are already available in many companies and that the same set of tools can be used for data access.

Related with transferring Web data to a star schema of a data warehouse, our research can be compared with DataFoundry [14,15] and the DWQ project [11,12].

The DataFoundry project is based on a mediated data warehouse architecture with an ontology infrastructure, it makes extensive use of this infrastructure to generate mediators automatically. However, there are some important differences between our work and theirs. At first, they integrate several existing community databases in a data warehouse, in contrast, we focus on Web data. Comparing with data in scientific databases, Web data in most cases is unstructured or semistructured and has no explicit schema. In our system, MIX provides a mediated model for preparing Web data for mapping them to an existing star schema. An object/relational mapping approach is used to transform data in the mediated model to the star schema of the data warehouse.

In the DWQ project, information integration in the data warehousing environment is studied. They consider two important aspects concerning the design and maintenance of particular data warehouse applications, those are conceptual modeling of the domain, and reasoning support over the conceptual representation. Their approach provides a system-independent specification of the relationships between sources and between sources and the enterprise model at the conceptual level. But DWQ focuses only on information sources that posses an explicitly specified data schema. Their logical data model is the relational model. In our work, we not only construct domain knowledge representation at the conceptual level, but use self-describing object model at the logical level as well. This offers the possibility to integrate Web data in a semantic correct way. Besides, in [11,12] the mapping between logical and physical level is not explored. In our system, a file of mapping rules is constructed. With the aid of such a rule file and an object/relational mapping approach we can process transformations between MIX objects and tables of a data warehouse semi-automatically.

7 Conclusions

In this paper, we introduced a Web warehousing approach for integrating a company‘s own data and Web data in an existing data warehouse. We have shown the benefit of integrating local data with external data, e.g., a competitor’s pricing scheme. We discussed our framework for managing and integrating Web data. In this framework we use a self-describing object model that represents data together with metadata that makes implicit assumptions about the structure and semantics of the data explicit. This model provides the basis for integrating Web-based data.

The prepared data can then be transferred to the star schema of a given data warehouse. For this, we have developed a Transformation Processor, which maps MIX objects to dimension and fact tables using mapping rules specified in a mapping file. The system has been implemented in Java.

Current work concentrates on performance improvement. Future work will extend the functionality of the Transformation Processor and the Incremental Maintenance Processor and will address mappings to the multidimensional model.

References

1. Ambite, J. L.; Ashish, N.; Barish, G.; et al.: ARIADNE: A System for Constructing Mediators for Internet Sources, Proc. of the ACM SIGMOD International Conference on Management of Data, Seattle, USA, 1998 201
2. Anderson, C. R.; Levy, A. Y.; Weld, D. S.: Declarative Web-site Management with Tiramisu, Proc. of the International Workshop on the Web and Databases, Philadelphia, USA, 1999 201
3. Ashish, N.; Knoblock, C. A.; Shahabi, C.: Selectively Materializing Data in Mediators by Analyzing User Queries, Proc. of the International Conference on Cooperative Information Systems, Edinburgh, Scotland, 1999 201
4. Beeri, C.; Elber, G.; Milo, T.; et al.: WebSuite – A Tool Suite For Harnessing Web Data, Proc. of the International Workshop on the Web and Databases, Valencia, Spain, 1998 201
5. Bernstein, P. A.; Pal, S.; Shutt D.: Context-Based Prefetch for Implementing Objects on Relations, Proc. of the International Conference on Very Large Data Bases, Edinburgh, Scotland, 1999 206, 207
6. Bhowmick, S. S.; Madria, S. K.; Ng, W.-K.; Lim, E. P.: Web Warehousing: Design and Issues, Proc. of the International Workshop on Data Warehousing and Data Mining, Singapore, 1998 201, 212
7. Bornhövd, C.: MIX – A Representation Model for the Integration of Web-based Data, Technical Report, DVS98-1, Department of Computer Science, Darmstadt University of Technology, Nov., 1998 202
8. Bornhövd, C.: Semantic Metadata for the Integration of Web-based Data for Electronic Commerce, Proc. of the International Workshop on Advance Issues of E-Commerce and Web-based Information System, Santa Clara, USA, 1999 202
9. Bornhövd, C.; Buchmann, A. P.: A Prototype for Metadata-based Integration of Internet Sources, Proc. of the International Conference on Advanced Information Systems Engineering, Heidelberg, Germany, 1999 202
10. Calvanese, D.; Giacomo, G. De; Lenzerini, M.; Vardi, M. Y.: Query Answering Using Views for Data Integration over the Web, Proc. of the International Workshop on the Web and Databases, Philadelphia, USA, 1999 201
11. Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; et al.: Description Logic Framework for Information Integration, Proc. of the International Conference on Principles of Knowledge Representation and Reasoning, Trento, Italy, 1998 213
12. Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; et al.: Information Integration: Conceptual Modeling and Reasoning Support, Proc. of the International Conference on Cooperative Information Systems, New York, 1998 213

13. Carey, M.; Doole, D.; Mattos, N.: O-O, What Have They Done to DB2?, Proc. of the International Conference on Very Large Data Bases, Edinburgh, Scotland, 1999 206
14. Critchlow, T.; Ganesh, M.; Musick, R.: Automatic Generation of Warehouse Mediators Using an Ontology Engine, Proc. of the International Workshop on Knowledge Representation meets Databases, Seattle, WA, 1998 213
15. Critchlow, T.; Ganesh, M.; Musick, R.: Meta-Data based Mediator Generation, Proc. of the International Conference on Cooperative Information Systems, New York, 1998 213
16. Davulcu, H.; Freire, J.; Kifer, M.; Ramakrishnan, I. V.: A Layered Architecture for Querying Dynamic Web Content, Proc. of the ACM SIGMOD International Conference on Management of Data, Philadelphia, USA, 1999 201
17. Hackathorn, R. D.: Web Farming for the Data Warehouse, Morgan Kaufmann, 1999 201
18. Keller, A. M.: Persistence Software: Bridging Object-Oriented Programming and Relational Databases, Proc. of the ACM SIGMOD International Conference on Management of Data, Washington, D. C., 1993 206
19. Keller, W.: Mapping Objects to Tables, Proc. of European Conference on Pattern Languages of Programming and Computing, Kloster Irsee, Germany, 1997 206, 207
20. Keller, W.: Object/Relational Access Layers, Proc. of European Conference on Pattern Languages of Programming and Computing, Bad Irsee, Germany, 1998 206, 207
21. Labrinidis, A.; Roussopoulos, N.: On the Materialization of WebViews, Proc. of the International Workshop on the Web and Databases, Philadelphia, USA, 1999 201
22. Mattison, R.: Web Warehousing and Knowledge Management, McGraw-Hill, 1999 201
23. McHugh, J.; Widom J.: Integrating Dynamically-Fetched External Information into a DBMS for Semistructured Data, SIGMOD Record, 26(4), 1997 201
24. Ng, W.-K.; Lim, E.-P.; Huang, C.-T.; et al.: Web Warehousing: An Algebra for Web Information, Proc. of the IEEE Forum on Research and Technology Advances in Digital Libraries, Santa Barbara, USA, 1998 201, 212
25. Wiederhold G.: Mediators in the Architecture of Future Information Systems, IEEE Computer, 25(3), 1992 203
26. Zhu, Y.: A Framework for Warehousing Web Contents, Proc. of the International Computer Science Conference on Internet Applications, Hong Kong, 1999 201, 202

Hierarchically Classifying Chinese Web Documents without Dictionary Support and Segmentation Procedure[1]

Shuigeng Zhou, Ye Fan, Jiangtao Hu, Fang Yu, and Yunfa Hu

Computer Science Department, Fudan University, Shanghai, 200433, China
sgzhou@fudan.edu.cn

Abstract. This paper reports a system that hierarchically classifies Chinese web documents without dictionary support and segmentation procedure. In our classifier, Web documents are represented by N-grams ($N \leq 4$) that are easy to be extracted. A boosting machine learning approach is applied to classifying Web Chinese documents that share a topic hierarchy. The open and modularized system architecture makes our classifier be extendible. Experimental results show that our system can effectively and efficiently classify Chinese Web documents.

1 Introduction

The popularity of searching the contents of the Internet has recently increased recognition of the demand for automatic assignment of class labels to documents in large text collections. Web search engines such as *Yahoo* [1] make use of manually assigned class labels to help users understand the structure of its text collection. However, manual information is time-consuming to produce and so automated methods of class assignment are needed.

Document classification is the problem of subsuming text documents into categories or classes. There has been a great deal of research on automatic class label assignment and great strides have been made in this arena [2]. However, obvious

[1] This work is supported by the 973 High-Tech Projects Foundation of China and partially supported by a grant (No. 69933010) from NSFC.

imbalance in research effort exists, which can been seen mainly from the following aspects:

- Research of Chinese documents classification lacks behind considerably that of English documents.
- Even for English documents classification, previous research has focused on flat classification problem.
- Most document classifiers are for pure text collections, while few classifiers are developed specifically for hypertext documents.
- Conventional documents classifiers use dictionaries and segmentation procedures to extract keywords as document features, which complicates classifier systems and affects classification efficiency.

With these points in mind, in this paper, we study the problem of hierarchically classifying Chinese web documents without dictionary support and segmentation procedure. We represent the Web documents with N-grams that are easy to be extracted. A boosting machine learning approach is applied to classifying Web Chinese documents that share a topic hierarchy. The open and modularized system architecture makes our classifier be extendible.

The reminder of this paper is organized as follows. First, in Section 2 we present techniques for Chinese Web documents classification in details, including N-grams information extraction, documents features selection and the Boosting algorithm for hierarchical classification. Then in Section 3, we describe the open and modularized system architecture for documents classification. Our classifier is evaluated in Section 4. And a brief survey of related work is given in Section 5. Finally, we conclude the paper in Section 6.

2 Techniques

2.1 N-grams Extraction

Since we have decided to avoid using of dictionary and segmentation program while classifying Chinese Web documents, we represent the Web documents with N-grams that are easy to be extracted. However, for a given documents collection, the number of N-grams is very large, in which a considerable part is of little usefulness to classification. The usefulness of a N-gram item can be measured qualitatively by its occurrence frequency, distribution and centralization.

Definition 1 The frequency of N-gram item *t* occurring in document *d* is its occurrence times in *d*. We denote it as *tf*.

Definition 2 The distribution of N-gram item *t* in document class *c* is the number of documents that contain *t*. We denote it as *df*.

Definition 3 The centralization of N-gram item *t* in document collection *D* is defined to be the inversion of the number of classes that include *t*. We denote it as *ICf*.

Intuitively, a N-gram item with higher *tf*, *df* and *ICf* is more useful to classification, *i.e.* it is more distinguishable. However, there is no formal mathematical approach to guide how to extract the most distinguishable N-grams in terms of their *tf*, *df* and *Df*. In order to limit the chance of extracting less useful N-grams, we give two constraints as follows.

Constraint 1 Given a pre-specified minimum value of *tf*, denoted as *min-tf*, a N-gram item *t* in document *d* is extracted only if its *tf* is no less than *min-tf*.

Constraint 2 Given a pre-specified minimum value of *df*, denoted as *min-df*, a N-gram item *t* in class *c* is extracted only if its *df* is no less than *min-df*.

Before giving an efficient algorithm for N-grams extraction, we present the following definition and lemma.

Definition 4 For *i*-gram item t_i and *j*-gram item t_j, $i \geq j$, if t_j is contained in t_i, than we say t_j is a sub-item of t_i, and denote $t_j \subseteq t_i$.

Lemma 1 If *i*-gram item t_i meets Constraint 1 and Constraint 2, then all the sub-items of t_i meet Constraint 1 and Constraint 2 too.

Following Definition 1, 2 and 4, it is easy to prove Lemma 1. Based on Lemma 1, we give a stepwise N-grams extraction algorithm shown in Figure 1.

There is another problem to be solved: How much should the largest value of N be? Intuitively, the largest value of N, let's denote it as *MAX-N*, should be set to such a value that the *i*-grams with $i \leq MAX\text{-}N$ can cover most of keywords in the documents collection. According to a statistic analysis [3], in Chinese documents, as far as occurrence frequency is concerned, 1-chrarcter words make up the dominating part, and the next is 2-character words, then 3-character and 4-character words. Words with more than 4 characters are quite few and infrequent. So for any document, we can basically represent it with only the 1/2/3/4-character words. In other word, the largest value of N can be 4 because the 1/2/3/4-grams would cover all the 1/2/3/4-character words in Chinese documents.

```
Algorithm 1: N-grams extraction
Input: document collection $D$, $min\text{-}tf$, $min\text{-}df$ and $MAX\text{-}N$.
Output: A set of N-grams $S$ (N ≤ $MAX\text{-}N$) that meet Constraint 1 and 2.
Process (basic steps):
1. Finding the 1-grams set $S_1$: Scanning all documents in $D$ one by one, and
   extracting all 1-grams that meet Constraint 1 and 2.
2. Finding the 2-grams set $S_2$: Carrying out Cartesian product $S_1 \times S_1$ to produce the
   candidate 2-grams set $C_2$ from which the items not conforming to Constraint 1
   and 2 are removed, and the left items make up $S_2$.
3. For $i$=3 to $MAX\text{-}N$ do:
   3.1 Constructing the candidate $i$-grams set $C_i$: for two arbitrary ($i$-1)-gram items
       $t_m$ and $t_n$ in $S_{i-1}$, $t_m(k)$ and $t_n(k)$ ($k$=1～($i$-1)) refer to the $k$-th character in $t_m$ and $t_n$
       respectively. If $t_m(k+1)=t_n(k)$ for $k$=1～($i$-2), then
       $C_i=C_i \cup t_m t_n(i-1)$.
   3.2 Removing the items in $C_i$ that not conforming to Constraint 1 and 2, then
       the left items make up $S_i$.
4. $S=S_1 \cup \ldots \cup S_{MAX\text{-}N}$.
```

Fig. 1. Algorithm for N-grams extraction

2.2 Features Selection

We take the extracted N-grams as documents features, which are also referred to as *terms*. Still, the number of the extracted N-grams is very large, which will affect the quality and efficiency of classification. So a selection process is necessary over the extracted N-grams to get a subset of document features for classification. We use three statistic approaches to accomplish the selection task. The following notation is used: C is the class variable; c denotes a specific class; t is an arbitrary N-gram item.

2.2.1 Information Gain (IG)

Information gain measures the number of bits of information obtained for category prediction by knowing the presence or absence of a term in a document. The information gain of term t is defined to be:

$$IG(t) = \sum_{c \in C} (P(c,t)\log(\frac{P(c,t)}{P(c)P(t)}) + P(c,\bar{t})\log(\frac{P(c,\bar{t})}{P(c)P(\bar{t})})). \quad (1)$$

2.2.2 Mutual Information

Mutual information is a criterion commonly used in statistical language modeling word associations and related applications. The mutual information criterion between *t* and *c* is defined to be

$$MI(t,c) = \log P_r(t \mid c) / P_r(t). \quad (2.1)$$

To measure the goodness of a term in a global feature selection, the category-specific scores of a term are combined into two alternate ways:

$$MI_{avg}(t) = \sum_{c \in C} P(c) MI(t,c); \quad (2.2)$$

$$MI_{\max}(t) = \max_{c \in C}\{MI(t,c)\}. \quad (2.3)$$

2.2.3 The χ^2 Statistic

The χ^2 statistic measures the lack of independence between t and c and can be compared to the χ^2 distribution with one degree of freedom to judge extremeness. The χ^2 statistic term-goodness measure between *t* and *c* is define to be:

$$\chi^2(c,f) = \frac{(P(c,f)P(\bar{c},\bar{f}) - P(c,\bar{f})P(\bar{c},f))}{P(c)P(f)P(\bar{c})P(\bar{f})} \quad (3.1)$$

Similarly, for each category, the χ^2 statistic between each unique term in a training corpus and the category is computed, and then combined the category-specific scores of each term into two scores:

$$\chi^2_{avg}(f) = \sum_{c \in C} P(c)\chi^2(c,f); \quad (3.2)$$

$$\chi^2_{\max}(f) = \max_{c \in C}\{\chi^2(c,f)\}. \quad (3.3)$$

2.2.4 Further Features Reduction

There maybe still some redundant features exist after the above selection process. In Chinese documents, some fixed words or specific names are composed of Chinese characters that occur simultaneously. For example, "巴基斯坦" is a country name, if it is selected as a document feature, then "巴基", "巴基斯" and "基斯坦" will be selected too, which means that three of them are redundant, and the redundant features should be removed. We give a lemma for redundant features removing as follows.

Lemma 2 Given two N-gram items t_i and t_j, if $t_i \supset t_j$ and score(t_i)=score(t_j), then one of them is redundant, and we keep only t_i. Here, score (.) refers to one of the equations from (1) to (3).

2.3 Boosting Algorithm for Hierarchical Classification

Here we adopt the boosting machine learning technique to hierarchically classifying Chinese Web documents. The main idea of boosting is to combine many simple and moderately inaccurate classification rules into a single, highly accurate categorization rule [7]. The simple rules are trained sequentially; conceptually, each rule is trained on the examples that were most difficult to classify by the preceding rules.

Figure 2 is the classification algorithm AdaBoost.MH. Here, X denotes the document space; Y is a finite set of labels or classes. m=|X|, k=|Y|. For a specific training document x_i, y_i is a set of class labels to which x_i belongs, that is to say, x_i is assigned to multiple classes. As described in Figure 2, AdaBoost.MH maintains a set of weights as distribution D_t over examples and labels. Initially, this distribution is uniform. On each round t, the distribution D_t (together with the training sequence) is passed to the weak learner who computes a weak hypothesis h_t. The output of the weak learner is a hypothesis h: X×Y →R. The sign of $h(x,l)$ is interpreted as a prediction as to whether the label l is or is not assigned to x (*i.e.*, a prediction of the value of $y[l]$). The magnitude of the prediction $|h(x, l)|$ is interpreted as a measure of "confidence" in the prediction.

The key problem is how the weak learner generates the weak hypothesis $h(x,l)$. Given a term w, define $h(x, l)$ to be the following form:

$$h(x,l) = \begin{cases} c_{0l} & if \quad w \notin x, \\ c_{1l} & if \quad w \in x. \end{cases} \tag{4}$$

Here the c_{jl}'s are real numbers, and are calculated as follows.

$$c_{il} = \frac{1}{2}\ln\left(\frac{W_{+1}^{jl} + \varepsilon}{W_{-1}^{jl} + \varepsilon}\right) \tag{5}$$

where ε=1/mk and

$$W_b^{jl} = \sum D_t(i,l) \mid x_i \in X_j \wedge y_i[l] = b. \tag{6}$$

In equation (6), $X_0=\{x: w\notin x\}$, $X_1=\{x: w\in x\}$, j∈ {0,1}, b∈ {-1, +1}. From (5) and by setting $\alpha_t=1$, the minimized Z_t can be derived:

$$Z_t = 2\sum_{j\in\{0,1\}}\sum_{l\in Y}\sqrt{W_{+1}^{jl}W_{-1}^{jl}}\,. \tag{7}$$

For each term, values c_{jl} are calculated with (5), and Z_t is also evaluated for the resulting weak hypothesis $h_t(x, l)$. Once all terms in the training corpus have been searched, the weak hypothesis with the lowest Z_t is selected and returned by the weak learner.

Input: $(x_1,y_1),\ldots,(x_m, y_m)$ where $x_i\in X$, $y_i\subseteq Y$, $D_1(i, l) = 1/(mk)$.
Process: for t=1, …, T do
- Pass distribution D_t to weak learner.
- Get weak hypothesis h_t: X×Y→R.
- Choose $\alpha_t\in R$.
- Update:

$$D_{t+1}(i,l)=\frac{D_t(i,l)\exp(-\alpha_t Y_i[l]h_t(x_i,l))}{Z_t}$$

Where Z_t is a normalization factor (chosen so that $D_{t+}1$ will be a distribution).
Output: the final hypothesis:

$$f(x,l)=\sum_{t=1}^{T}\alpha_t h_t(x,l).$$

Fig. 2. The algorithm AdaBoost.MH

AdaBoost.MH is a general method that can handle multi-class multi-label classification problems. Here, we adopt AdaBoost.MH to hierarchically classifying Chinese Web documents. For each training document x_i, we should decide the set y_i. It should include all the class labels on the path from the leaf node corresponding to the training document x_i to the root node of the hierarchy structure. Figure 3 is an example illustrating how to set y_i. In Figure 3, c_i is class-label, while x_1 and x_2 are two training documents. From x_1 and x_2 to root node of the hierarchy structure, the paths are c_{221}-c_{22}-c_2 and c_{222}-c_{22}-c_2 respectively. So we have $y_1=\{c_{221}, c_{22}, c_2\}$ and $y_2=\{c_{222}, c_{22}, c_2\}$.

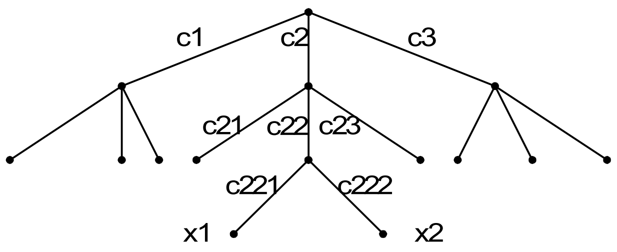


Fig. 3. An example of topic hierarchy structure

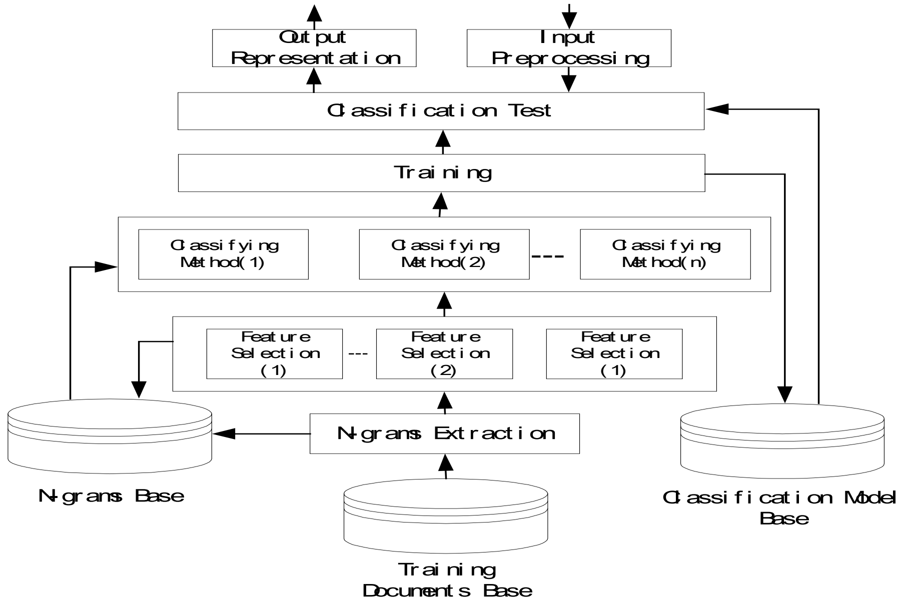


Fig. 4. The open and modularized Classifier architecture

3 An Open Architecture for Documents Classification System

Figure 4 illustrates the scheme of our classifier system. It is an open and modularized architecture that provides general interfaces for different feature selection methods and classification methods, which makes the system to be updated and improved conveniently once the more effective and efficient feature selection methods and classification methods are available. On the other hand, the intermediate results at each different stage during the training period are stored separately. Such a mechanism can promote the training efficiency.

4 Performance Evaluation

There is no standard Chinese Web documents collection available yet for classification experiment. We have to collect training and test documents manually by ourselves. We get the training and test documents from Yahoo China site (http: \\cn.yahoo.com) and the BBS of Fudan University. The topic hierarchy has three levels: the first level has 4 classes; the second level has 8 classes, and the third level includes 16 classes. So the total number of class-labels within the hierarchy structure is 28. The total number of documents is 2000, and every leaf class contains more than 50 documents. For every class, 70% documents are kept for training, and the left 30% for test. Results are average over 5 trials.

First, we examine the effectiveness of the three feature selection methods. The results are given in Figure 5, which shows that χ^2 statistic has the best effectiveness, and IG is the second. This result is similar to that of [4]. We then check the relationship between the number of class labels and classification performance. The experimental results are illustrated in Figure 6. It is obvious that as the number of class labels is increasing, the performance of classifier degrade. Our third experiment is to explore the impact of computation round number over classifier performance. The results shown in Figure 7 conform to our expectation that as the round number growing, classification quality is improved.

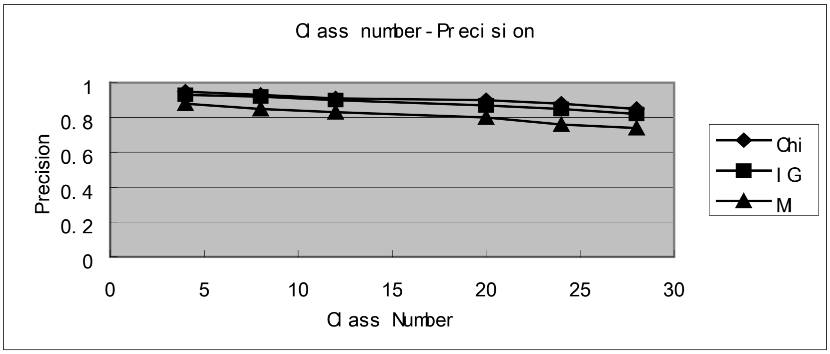
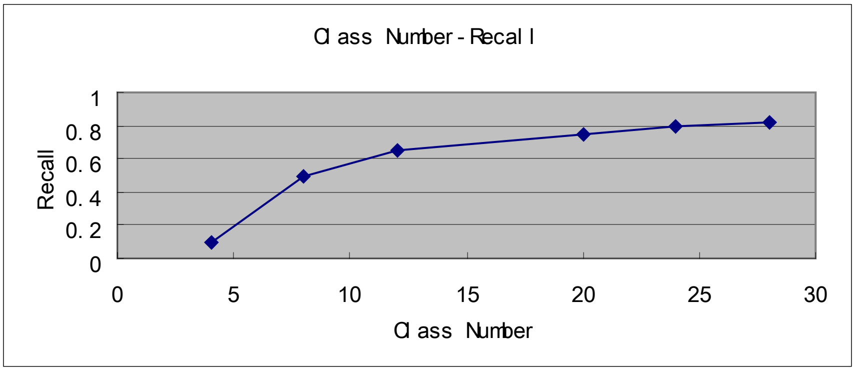


Fig. 5. Performance comparison with different feature selection methods



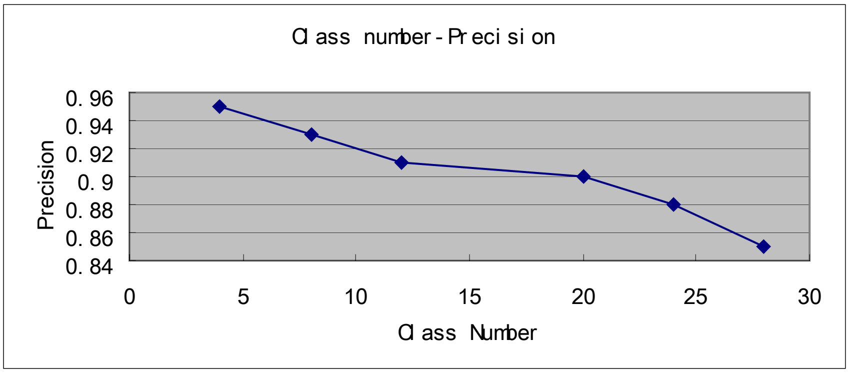


Fig. 6. The relationship between class number and classifier performance

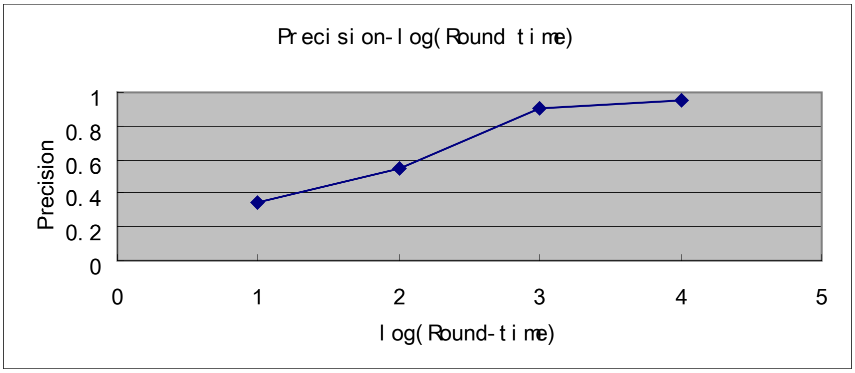


Fig. 7. Relationship between round number and classifier performance

5 Related Work

A variety of techniques for supervised learning algorithms have demonstrated reasonable performance for document classification. A non-exhaustive list includes naive Bayes [5], kNN [2], SVM [6], Boosting [7] and rule learning algorithms [8]. Several papers deal specially with hierarchical classification [9-12], which all divide the original classification task into a set of smaller classification problems corresponding to the splits in the classification hierarchy, and the smaller classifiers are built with Bayesian method, which are different from our method in this paper. There are also some reports on Web page categorization. [13] applies association rule and principal component clustering to feature selection, then hypergraph partitioning is used to categorize web pages. It seems that this method did not achieve good results. [14] discusses how to utilize structure information in Web pages classification. However, the results are not promising either.

6 Conclusions

We have presented an effective and efficient way to classify Chinese Web Documents hierarchically. Due to using N-grams to representing documents, our classifier need no dictionary support and segmentation procedure, which make it more competitive in flexibility and practicability than the conventional Chinese documents classifiers. However, we should notice that not all N-grams are really related to document topics, so more powerful feature selection methods are necessary to eliminate irrelevant and redundant features. The method proposed in [15] may be helpful to enhance feature selection in our system.

References

1. Yahoo! On-line guide for the Internet. http://www.yahoo.com/ (1995)
2. Yang Y. and Liu X. A re-examination of text categorization methods. Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) (1999)
3. Zhao B. and Xu L. Processing Chinese Information with Computer, Vol.2. Space Publisher House (1988)

4. Yang Y. and Pederson J. Feature selection in statistical learning of text categorization. In ICML-97 (1997) 412-420.

5. Lewis D. D. Naive (Bayes) at forty: The independence assumption in information retrieval. In Machine Learning: ECML-98, 10th European Conference on Machine Learning (1998) 4-15

6. Joachims T. Text categorization with support vector machines: learning with many relevant features. In Machine Learning: ECML-98, 10th European Conference on Machine Learning (1998) 137-142

7. Schapire R. E. and Singer Y. Improved boosting algorithms using confidence-rated predictions. In Proceedings of 11th Annual Conference on Computational Learning Theory (1998) 80-91

8. Cohen W. W. and Singer Y. Context-sensitive learning methods for text categorization. In SIGIR'96: Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1996) 307-315

9. Koller D. and Sahami M. Hierarchically classifying documents using very few words.

10. Mladenic D., *et al*. Feature selection in text learning. Proc. Of 10th European Conference on Machine Learning ECML98 (1998)

11. McCallum A., *et al*. Improving text classification by shrinkage in a hierarchy of classes. In ICML-98 (1998) 359-367

12. Chakrabarti S., *et al*. Using taxonomy, discriminants, and signatures for navigating in text databases. Proc. Of the 23rd VLDB Conference Athene, Greece (1997)

13. Moor J. and Han E. H (Sam). Web page categorization and feature selection using association rule and principal component clustering (1998)

14. Quek C. Y. Classification of World Wide Web documents. Senior Honors Thesis, CMU (1997)

15. Koller D. and Sahami M. Toward optimal feature selection. In Lorenza Saita, ed., Machine Learning: Proc. of the 13th International Conference, Morgan Kaufman (1996)

Web Clustering and Association Rule Discovery for Web Broadcast

Shi Wang, Wen Gao, Jintao Li, Tiejun Huang, and Hui Xie

Institute of Computing Technology
Chinese Academy of Sciences, Beijing 100080
{shiwang,wgao,jtli,tjhuang,xiehui}@ict.ac.cn

Abstract. When we broadcast web pages package through broadband broadcast network, what we will broadcast and how we broadcast are two problems. First, we formalize these problems as a clustering problem. Second, we use the association rule to give a better result. In this paper we provide a new web mining method WebClustering to resolve these problems. This method combines the clustering idea and association rule discovery idea. The mining object is the web pages of Cache and Log in WWW Proxy server. By this method we can find a valuable Web broadcast set and create some index HTML pages to direct the users to navigate.

1 Introduction

We can organize web pages that are in the proxy cache into a big package to broadcast. This way brings about a problem: there are thousands of web sites and ten thousands of web pages in this broadcast stream and they always vary every day. Then how can the users browse? A way is to build some index web pages to facilitate the user to browse. When a user browsing, these index pages will help the user to browse the web page package. We need to create these index web pages rationally and automatically.

This paper employs the web usage mining approach, makes use of the hyperlinks between the web pages and the frequency of the web site to find those web sites (centers) that can represent other web sites mostly. Then we create the index web pages with these hyperlinks that link to these centers. So we require an approach to find these representative sites. Clustering [1] can solve this problem naturally. These representative sites can be regarded as the clustering center. We present the WebClustering approach to look for these centers, then employ the association rule discovery algorithm to provide more information to organize the center pages.

Cooley et al [2] firstly give the web usage mining's definition and classification and give a WEBMINER system. Our approach prepares the data in the different way that incorporates the idea of the web structure mining. PageGather [3] algorithm proposed the idea of optimizing the structure of Web sites based on co-occurrence patterns of pages within usage data for the site. Their approach doesn't use the structure information of the web pages (i.e. the links). Our approach not only uses the structure information to cluster but also uses the association relation in the log to provide association rule.

Section 2 gives the definition, notation, and mining objects. Section 3 describes the WebClustering approach. In section 4, we give the experiment.

2 Definitions and Notations

Definition 1 *support*: Within a time interval *T*, the product of the number of times that users access an *IP* address and the number of outer hyperlinks of that *IP* address. For example: *support* (IP_A) = (the number of times that users access the IP_A) × (the number of outer hyperlinks of the IP_A).

Definition 2 Browse transaction: a browse transaction t_i is defined as: $t_i = < IP_i, support(IP_i), \{$ the hyperlinks that point to other web sites $\} >$

Definition 3 *step* (IP_A, IP_B): The least number of the sites that is in the path that someone browses from IP_A to IP_B.

Definition 4 d_{IP_A,IP_B}: the distance between IP_A and IP_B.

When $support(IP_B) < support(IP_A)$:

$$d_{IP_A,IP_B} = \frac{step(IP_A, IP_B)}{support(IP_A)} \tag{1}$$

If the condition isn't satisfied, then the distance is the maximum.

We can get the original mining data from the Cache and the Log in WWW Proxy. The Log of the Proxy conforms the standard of W3C[4]:

Through the definitions, we can build a model that is similar to the real environment. Such as Fig. 1, each node represents an *IP* and the number represents its *support*. Fig. 2. describes the data preparation process:

1) Get all HTML files from the Cache. 2) According to the process of Fig. 2, the browse transactions are created and they are organized into a *WTT* (Web Transaction Table) that includes *n* browse transactions. The transaction in the table is sorted by the *support* in descent order. The *support* must be processed into a unique *support*.3) We create the same table as the original *CCT* (Cluster Center Table), the IP_i in each transaction is the original cluster center. There are *n* original cluster centers.

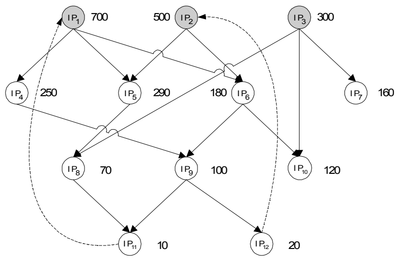


Fig. 1. The site association model

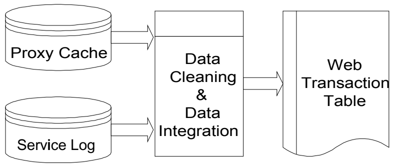


Fig. 2. Get the Web Transaction Table

3 Clustering and Association Rule Discovery

The intention of the algorithm is to reduce the number of the centers in *CCT*. The final intention is to let each user visit each site efficiently.

$$Minimise: P(WTT,CTT,W)=\sum_{j=1}^{n}\sum_{i=1}^{n}w_{i,j},$$
$$\sum_{j=1}^{n}w_{i,j}>0, w_{i,j}\geq 0, i=1,...,n, j=1,...,n \qquad \textbf{(2)}$$

The function P represents the cost of that all users access these sites. Where W is an $n\times n$ partition matrix, it is used to represent which web site is in which cluster. For example: if $w_{i,j}$ isn't the maximum, which represents that IP_i is in the IP_j cluster and the distance of IP_i and IP_j is $w_{i,j}$. Once the algorithm is finished, the *CCT* is the final clustering result.

The clustering search method is a directed graph Broad-First Search.

Algorithm: Cluster_Search
Input: *WTT*, *CCT*, *W*
Begin

1. $IP_i = Max\ (WTT)$; {This step finds the unmarked site with max *support*}
2. *Mark* (*CCT*, IP_i);{This step marks IP_i a cluster center in *CCT*.}
3. *Mark* (*WTT*, IP_i);{This step marks IP_i a visted site in *WTT*.}
4. *BFS_Cluster_Search* (*WTT*, *CCT*, IP_i) ;{This IP_i is the cluster center. The Broad-First Search starts from this IP_i to find its cluster sets. When the search meets an IP_j: 1)*Mark* (*WTT*, IP_j); 2)Adding the distance of IP_j and IP_i to $W_{i,j}$; 3) Adding the IP_j in the IP_i's outer hyperlink table in *CCT*. Repeat these steps until the search ends. The IP_i is the center of the search sets. When the search meets the marked site, there are two different process methods according to two different situations: 1) If the IP_j is marked but it isn't a cluster center, and then the search will process it and continue. 2) If the IP_j is marked and it is a cluster center, and then if the *support*'s difference of the two clusters is in a threshold (such as 5%), then the algorithm will combine the two clusters. The new center is IP_i, but its *support* is the *support* of IP_j.}
5. Return 1, until the algorithm can't find the new center.

End.
Output: *CCT*, *W*

After getting the algorithm, we get the cluster results in *W*:

Table 1. The cluster result in *W* (the example is Fig. 1.)

Site	Cluster	Center
IP_1		IP_1 (1/700)
IP_2		IP_2 (1/500)
IP_3		IP_3 (1/300)
IP_4	IP_1	IP_1 (1/700)
IP_5	IP_2, IP_1	IP_1 (1/700)
IP_6	IP_2, IP_1	IP_1 (1/700)
IP_7	IP_3	IP_3 (1/300)
IP_8	IP_5, IP_3, IP_2, IP_1	IP_1 (2/700)
IP_9	IP_6, IP_4, IP_2, IP_1	IP_1 (2/700)
IP_{10}	IP_6, IP_3, IP_2, IP_1	IP_1 (2/700)
IP_{11}	IP_9, IP_8, IP_6, IP_5, IP_4, IP_3, IP_2, IP_1	IP_1 (3/700)
IP_{12}	IP_9, IP_6, IP_4, IP_2, IP_1	IP_1 (3/700)

Then Cluster_Appoint algorithm scans W to find each site's center:

```
Algorithm:  Cluster_Appoint
Input:      $CCT$, $W$
Begin
  1. $i = 1$;
  2. $IP_j = Find\_Cluster\_Center(W, i)$ ;{This step is to find the center of $IP_i$ in the
ith row in $W$. This step finds the $IP_j$ with the minimal $d$, and the center of $IP_i$ is $IP_j$.}
  3. $Modify\_Cluster(CCT, W, i, IP_j)$ ;{Because $IP_i$ perhaps appears several times
in $CCT$ and it is perhaps in different cluster, this step will modify the $CCT$. Except the
$IP_j$ cluster includes $IP_i$, the algorithm will delete the $IP_i$ in other clusters.}
  4. $i = i + 1$, if $i > n$, stop; otherwise, return 1.
End.
Output: $CCT$
```

The example is Table 1: there are three cluster centers: IP_1 (IP_1 , IP_4 , IP_5 , IP_6 , IP_8, IP_9 , IP_{10} , IP_{11} , IP_{12}), IP_3(IP_3 , IP_7), IP_2(IP_2); but because IP_2(IP_2) can't represent many other sites, so we can consider deleting it.

The intention of association rule discovery is to help users to access these clusters. In our experiment, we find that the centers are portal sites generally, so we care more about the rule that comes from the site with high *support* to the site with low *support*. We can get access transaction according to Cooley's approach [2]. At the same time according to the result of clustering and Agrawal[5], we can get the rules that are from the center of a cluster to the cluster elements. Then we create the index web page: according to the cluster centers, cluster sets, and confidences, we can create a (some) broadcast index web page(s), the web page(s) can help user to access the entire broadcast web space.

4 Experiments

We choose a Proxy in Windows NT4.0 with 80-100 users, and get its cache and log of six months. There are 360M different files in cache; the HTML files have 10267 pages, 78.8M; log has 142M. After pre-processed, the cache has 1086 sites.

Definition 5 the cluster density $D(IP_i)$: IP_i is a cluster center. $D(IP_i)$ is the number of sites in this cluster.

Definition 6 the number of cluster centers C (CCT, *condition*): the number of cluster centers that each of the clusters satisfies the *condition* in CCT.

Definition 7 the total density TD(CCT, *condition*): the sum of all the cluster densities that each of the clusters satisfies the *condition* in CCT.

The definition of the *support* influences the result of the algorithm directly. We compare three kinds of different definitions of the *support*: 1) $support_1$: the *support* is defined in definition 1; 2) $support_2$: the number of the original outer hyperlink; 3) $support_3$: the number of times that all users access a site.

Table 2. The result of three kinds of different *support* definitions

	$support_1$	$support_2$	$support_3$
$C(CCT, D(IP_i)=0)$	526(526)	524(524)	563(563)
$C(CCT, D(IP_i) \geq 1)$	61(560)	56(562)	50(523)
$C(CCT, (D(IP_i) \geq 1\& D(IP_i) \leq 2)$	49(105)	46(100)	37(81)
$C(CCT, D(IP_i) \geq 3)$	12(455)	10(462)	13(442)

The number in the bracket is the total density that satisfies the same condition. $C(CCT, D(IP_i) \geq 3)$: means that the number of cluster centers that each cluster's cluster density ≥ 3. The final result manifests that there are 12 best cluster centers and 49 common centers. We analyze the some Proxy and Caches and find that the $support_1$ is the best definition.

5 Conclusions and Future Work

In this paper we present the clustering approach and association rule discovery approach to resolve the two problems that are how to get broadcast web page sets and how to organize them. We consider these problems as a special clustering problem and give the cluster definition. We employ the association rule discovery approach in order to resolve the problems better. So we present a new clustering approach: WebClustering. Through this approach, we can get a valued broadcast web page package and through the new created index web pages, users can access this package efficiently. The experiment result manifests our approach is successful.

There are some characteristics in our approach. 1) It is a kind of optimizing approach. 2) This approach provides the links between the different page sets, the web pages that aren't adjacent can be linked to be the new created index web page. 3) The result is the good self-organized broadcast web page sets, which is equal to visualizing the mining results.

Because our approach can't deal with the content of the web, the further work is to classify the web pages according to XML or other machine learning approaches. If we have the classification information of the web pages we can use it to improve the index web pages.

Reference

1. Rasmussen, E. Clustering algorithms. In Frakes, W., and Baeza-Yates, R., eds., Information Retrieval. Prentice Hall, Eaglewood Cliffs, N.J. 419-442. 1992.
2. Cooley, R., Mobasher, B., et al. Web Mining: Information and Pattern Discovery on the World Wide Web. In Proceedings of International Conference on Tools with Artificial Intelligence, Newport Beach, IEEE, 1997.
3. Perkowitz, M., and Etzioni, O. Adaptive Web Sites: Automatically Synthesizing Web Pages. in Proceedings of AAAI98. 1998.
4. Luotonen, A. The common log file format. http://www.w3.org/pub/WWW/, 1995.
5. Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In Proc. Of the 20th VLDB Conference, 487-499, Santiago, Chile, 1994.

A Non-Euclidean Model for Web Retrieval

Z.W. Wang, R.B. Maguire, and Y.Y. Yao

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2

Abstract. A non-Euclidean model is proposed for resolving problems in similarity-based retrieval. This model establishes a theoretical basis and implementational guidelines for designing similarity-based Web retrieval systems.

1 Introduction

Similarity-based matching is widely used in the vector space model. However, "the widespread adoptance of similarity-based matching is hampered by disagreements over how similarity measures should be constructed and how large databases should be indexed so the similarity matching is even possible" [10]. Observation shows that the following problems exist in similarity-based matching:

(1) A precise interpretation and definition of similarity/dissimilarity, as well as formal properties of similarity/dissimilarity, are not fully studied. Consequently, some similarity/dissimilarity measures show very strange features.
(2) It is not clear in which situations a particular similarity/dissimilarity measure should be used.
(3) Distance is a simple and natural way to measure the closeness of two vectors. If the ranking result using a similarity/dissimilarity measure is the same as using a distance function, then there is no need for the similarity/dissimilarity measure. If the results are different, one should be able to explain the differences.

The main objective of this paper is to propose a non-Euclidean model intended to overcome these hindrances and establish a theoretical basis and implementational guidelines in using similarity-based matching in Web retrieval systems.

In the vector space model [11, 12], a similarity/dissimilarity measure m is a real valued function of two vectors. Many similarity/dissimilarity measures have been proposed and studied [7, 8]. But the term "measure" is not strictly defined. It is not surprising that some measures have peculiar properties. For example, the pseudo-cosine measure is defined as:

$$scos(\mathbf{q}, \mathbf{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sum_{i=1}^{n} q_i \sum_{i=1}^{n} d_i}.$$

Wang [14] showed that, for pseudo-cosine measures, the term "similarity" may be misleading. Using this measure, a vector may not be the vector that is most similar to itself.

Example 1. Assume that a system uses the pseudo-cosine measure. Let $D = \{\mathbf{d_1}, \mathbf{d_2}, \mathbf{d_3}, \mathbf{d_4}\}$, where $\mathbf{d_1} = (100, 0, 0)$, $\mathbf{d_2} = (0, 100, 0)$, $\mathbf{d_3} = (0, 0, 100)$, and $\mathbf{d_4} = (30, 34, 36)$. Suppose that a user does not want articles too focused on a single topic and provides a query $\mathbf{q} = \{30, 34, 36\}$. It follows: $scos(\mathbf{q}, \mathbf{d_1}) = 0.3$, $scos(\mathbf{q}, \mathbf{d_2}) = 0.34$, $scos(\mathbf{q}, \mathbf{d_3}) = 0.36$, and $scos(\mathbf{q}, \mathbf{d_4}) = 0.3352$. The ranking $\mathbf{d_3} \succ \mathbf{d_2} \succ \mathbf{d_4} \succ \mathbf{d_1}$ does not rank $\mathbf{d_4}$ ahead.

2 A Non-Euclidean Model

In the proposed non-Euclidean model, we assume that dissimilarity between documents can be evaluated by a metric [6], that is, a symmetric nonnegative function $m(x, y)$ satisfying the Triangle Inequality $m(x, y) + m(y, z) \geq m(x, z)$. Since the space is curved, the shape of the space may differ from one place to another. In this model, it is possible that different formulas can be used in different parts of the space. Each similarity/dissimilarity measure is suitable for a subset of documents, yet none is suitable for the whole space. This explains why so many similarity/dissimilarity measures have been proposed.

The term "measure" used in the conventional vector space model does not have a strict common mathematical definition. It is just a function of two variables. Actually, in mathematics, a *measure* is a function of *one* variable, and has very precise technical definitions (usually involving Sigma Algebras). In the non-Euclidean model, we use the non-Euclidean distance to measure the similarity/dissimilarity between two points.

Web space can be viewed as a *topological space* [1] with a non-trivial topology. Let S be the collection of all accessible Web pages. Assume that there is a search engine that can access all these Web pages. Assume further that the search engine uses exact matching and supports queries combined by Boolean operations AND and OR. Let $\mathcal{U}$ be the collection of all the search results. We can verify that $\mathcal{U}$ is a topology on the Web space. Both the empty set and the entire set belong to $\mathcal{U}$, i.e., $\emptyset \in \mathcal{U}$ and $S \in \mathcal{U}$. If U_i is the search result of query $\mathbf{q_i}$, for $i = 1, \ldots, k$. Then $\bigcap_{i=1}^{k} U_i$ is the search result of $\mathbf{q_1}$ AND $\mathbf{q_2}$ AND $\ldots$ AND $\mathbf{q_k}$. This implies $\bigcap_{i=1}^{k} U_i \in \mathcal{U}$. Similarly, $\bigcup_{i=1}^{k} U_i$ is the search result of $\mathbf{q_1}$ OR $\mathbf{q_2}$ OR $\ldots$ OR $\mathbf{q_k}$, and $\bigcup_{i=1}^{k} U_i \in \mathcal{U}$. Therefore, $\mathcal{U}$ is a topology. We call it the *topology induced by queries*. With this topology, the collection of all Web documents is a topological space. We call it the *Web space*.

In the proposed non-Euclidean model, The Web space is a manifold. The dissimilarity measure is defined by a Riemannian metric on the manifold.

Manifold [4] generalizes the notion of *surface*. Roughly speaking, a manifold is a surface without involving the ambient space. Rigorously, an n-dimensional manifold is a topological space [1, 9] M such that any point in M has a neighborhood which is homeomorphic to R^n. The concept of *differentiation* can be applied on *differentiable manifolds* [2, 5, 13]. The following is a formal definition.

Definition 1. *[13] A* differentiable manifold *of dimension n is a pair (M, X) where M is a Hausdorff topological space*[1]*, and X is a collection of homeomorphic mappings $x_\alpha : M \supset U_\alpha \to V_\alpha \subset R^n$ of open sets U_α of M into R^n such that:*

(1) $\bigcup_\alpha U_\alpha = M$.
(2) for any pair α, β, with $U_\alpha \bigcap U_\beta = W \neq \emptyset$, the mapping $x_\beta \circ x_\alpha^{-1}$ is differentiable.
(3) X is maximal*; that is, if $x_\gamma : M \supset U_\gamma \to V_\gamma \subset R^n$ is a mapping such that for each $x_\alpha \in X$ with $U_\alpha \bigcap U_\gamma \neq \emptyset$, $x_\gamma \circ x_\alpha^{-1}$ and $x_\alpha \circ x_\gamma^{-1}$ are differentiable mappings from $x_\alpha(U_\alpha \cap U_\gamma)$ and $x_\gamma(U_\alpha \cap U_\gamma)$ into R^n, then $x_\gamma \in X$.*

The pair (U_α, x_α) (or the mapping x_α with $\mathbf{a} \in U_\alpha$) is called a *parameterization* (or *local coordinate system*) of M at $\mathbf{a}$. U_α is called a *coordinate neighborhood* at $\mathbf{a}$. A family $\{(U_\alpha, x_\alpha)\}$ satisfying (1), (2), and (3) is called a *differentiable structure* on M.

In the conventional vector space model, it is assumed that Web space is embedded in R^n. In the non-Euclidean model, we embed it in a differentiable manifold M. The topology on M has a correspondence to $\mathcal{U}$, the topology induced by queries. If u_α and u_β are two open sets in $\mathcal{U}$, and U_α and U_β are their corresponding sets in M, then $U_\alpha \bigcap U_\beta$ is the corresponding open set of $u_\alpha \bigcap u_\beta$, and $U_\alpha \bigcup U_\beta$ is the corresponding open set of $u_\alpha \bigcup u_\beta$. With this understanding, we also call M the Web space.

To further explain these notions, assume that we add a new ranking function to the Boolean based search engine. In response to a query $\mathbf{a}$, the search engine first uses exact match to locate a set of Web documents u_α. It then indexes the documents. The indexed documents are considered as vectors (points) in R^n. The indexing defines the homeomorphic mappings $x_\alpha : M \supset U_\alpha \to V_\alpha \subset R^n$, where U_α is an open set in M, and V_α is an open set in R^n. The indexing method defines the parameterization x_α of M at $\mathbf{a}$. The local coordinate system is realized in V_α.

Definition 2. *[13] Let (M, X) be a differentiable manifold, and let x be a local coordinate system on U. Let $r_j : R^n \to R$ be the j-th coordinate function on R^n; that is, $r_j(a_1, a_2, \ldots, a_n) = a_j$. The j-th coordinate function of the coordinate system x is the function $x_j : U \to R$ defined by $x_j = r_j \circ x$.*

The n-tuple of function $(x_1, \ldots, x_n)$ is also referred to as a coordinate system.

Intuitively, a manifold is like a surface, and the notion of tangent vector to a manifold is analogous to that to a surface. In an Euclidean space, a tangent vector at a point $\mathbf{a}$ defines the *directional derivative* with respect to the given vector. Using this property, tangent vectors are defined as:

[1] A Hausdorff space is a topological space in which any two points have disjoint neighborhoods.

Definition 3. *[13] Let (M, X) be a differentiable manifold and let $\mathbf{a} \in M$. Let $C^\infty(M, \mathbf{a}, R)$ be the set of all real-valued differentiable functions in a neighborhood of $\mathbf{a}$. A* tangent vector *at $\mathbf{a}$ is a mapping $v : C^\infty(M, \mathbf{a}, R) \to R$ such that, if x is a local coordinate system on a neighborhood of $\mathbf{a}$, then there exists an n-tuple $(a_1, a_2, \ldots, a_n)$ of real numbers with the following property. For each $f \in C^\infty(M, \mathbf{a}, R)$,*

$$v(f) = \sum_{i=1}^{n} a_i \frac{\partial}{\partial r_i}(f \circ x^{-1}) \mid_{x(\mathbf{d})} .$$

The set of all tangent vectors to M at $\mathbf{a}$ is denoted as $T_{\mathbf{a}}M$, and is called the *tangent space* of M at $\mathbf{a}$. Let $x : U \to R^n$ be a parameterization at $\mathbf{a} \in M$. Let $\left(\frac{\partial}{\partial x_i}\right)_0$ be the directional derivative with respect to x_i at $\mathbf{a}$. $\{(\frac{\partial}{\partial x_1})_0, \ldots, (\frac{\partial}{\partial x_n})_0\}$ forms an *associated basis* in $T_{\mathbf{a}}M$.

Definition 4. *A Riemannian metric on a manifold M is a correspondence that associates to each point $\mathbf{a}$ in M an inner product $<,>_{\mathbf{a}}$ on the tangent space $T_{\mathbf{a}}M$.*

This correspondence changes from point to point in the following sense: if $x : U \to R^n$ is a parameterization at $\mathbf{a} \in U$, and $\{(\frac{\partial}{\partial x_1})_0, \ldots, (\frac{\partial}{\partial x_n})_0\}$ is the associated basis in $T_{\mathbf{a}}M$. Then $< (\frac{\partial}{\partial x_i})_0, (\frac{\partial}{\partial x_j})_0 >$, denoted as $g_{i,j}(\mathbf{a})$, is a differentiable function on U. The associated basis $\{(\frac{\partial}{\partial x_1})_0, \ldots, (\frac{\partial}{\partial x_n})_0\}$ generally is not an orthogonal system. The angle between vectors $(\frac{\partial}{\partial x_i})_0$ and $(\frac{\partial}{\partial x_j})_0$ is associated with the inner product of $(\frac{\partial}{\partial x_i})_0$ and $(\frac{\partial}{\partial x_j})_0$.

The function $g_{i,j}(\mathbf{a})$ is called the *local representation* of the Riemannian metric in the coordinate system. A differentiable manifold with a given Riemannian metric is called a *Riemannian manifold.*

Let M be a Riemannian manifold. There is a special form of curve called a *geodesic* [3, 15]. These curves resemble the straight lines in R^n. The length of the geodesic between two points can be used as distance. We call this the geodesic distance. There may be more than one geodesic connecting two points. The shortest one should be chosen.

The main idea of the non-Euclidean space model is as follows. The Web space is viewed as a Riemannian manifold. A retrieval system first locates a neighborhood, and then approximates the neighborhood by its tangent space. The geodesic distance describes the dissimilarity between vectors, but the formula for the geodesic distance is unknown. Therefore, the distance in the tangent space is used to approximate the geodesic distance. When the neighborhood is small enough, the ranking results should be the same.

Suppose a user sends a query by example. This query is a point $\mathbf{a}$ in the Web space M. Based on $\mathbf{a}$, the search engine constructs a query α that can be used for Boolean search. The collection of Web documents returned as relevant to α is the open set u_α. The search engine indexes the documents in u_α according to T, where $T = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$ is a controlled vocabulary consisting of keywords $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$. This indexing maps points in u_α to vectors in V_α, and can be viewed as x_α. The local coordinate system for V_α is the coordinate system defined by T. Let $x_j = r_j \circ x_\alpha$ be the j-th coordinate function. $(\frac{\partial}{\partial x_j})_0$ is a tangent vector at $\mathbf{a}$ along the x_j-axis, where x_j-axis is the image in U_α of the j-th axis in V_α corresponds to term $\mathbf{t}_j$. The set of vectors $\{(\frac{\partial}{\partial x_1})_0, \ldots, (\frac{\partial}{\partial x_n})_0\}$ is the associated basis in $T_\mathbf{a}M$. $\mathbf{a}$ is the origin in this coordinate system. Generally, the associated basis is not an orthogonal basis.

In the tangent space, the distance is calculated as:

$$dist(\mathbf{a}, \mathbf{d}) = \sqrt{\sum_{i,j=1}^{n} g_{i,j}(a_i - d_i)(a_j - d_j)},$$

where $g_{i,j}$ is the *local representation* of the Riemannian metric. When $g_{i,j} = \delta_{i,j}$, the distance is an Euclidean distance, where $\delta_{i,j}$ is the Kronecker Delta defined as:

$$\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}.$$

If we know the correct value of $g_{i,j}$, then this method should give the correct ranking. The problem is therefore transferred to finding the value of $g_{i,j}$, that is, to determining the Riemannian metrics at $\mathbf{a}$.

The quantity $g_{i,j}$ is associated with the angles between coordinate axes. We can estimate these angles by analyzing the relationship between keywords. A preliminary method is proposed as follows. Let $D = \{\mathbf{d}_i\}_{i=1,2,\ldots,m}$ be a subset of the indexed documents in V_α. The coordinate for $\mathbf{d}_i$ is $(w_{i1}, w_{i2}, \ldots, w_{in})$. Let

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & \ldots w_{1,n} \\ w_{2,1} & w_{2,2} & \ldots w_{2,n} \\ \ldots & \ldots & \ldots\ldots \\ w_{m,1} & w_{m,2} & \ldots w_{m,n} \end{pmatrix}.$$

Column j corresponds to keyword $\mathbf{t}_j$. If we consider $w_{i,j}$ as the coefficients of linear combinations of $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m$, then $\mathbf{t}_j$ can be expressed as

$$\mathbf{t}_j = w_{1,j}\mathbf{d}_1 + w_{2,j}\mathbf{d}_2 + \ldots + w_{m,j}\mathbf{d}_m.$$

If $\{\mathbf{d_i}\}_{i=1,2,\ldots,m}$ is an orthonormal basis, then column j can be viewed as the coordinates of $\left(\frac{\partial}{\partial x_j}\right)_0$. The Riemannian metric $g_{i,j}(\mathbf{a})$ can be calculated as

$$g_{i,j}(\mathbf{a}) =< (\frac{\partial}{\partial x_i})_0, (\frac{\partial}{\partial x_j})_0 >= \sum_{k=1}^{m} w_{ik} \cdot w_{jk}.$$

Using the Riemann space model, we have designed and implemented a new type of *personal* Web retrieval system that complements the current subject trees, search engines, and metasearch engines.

3 Conclusion

We showed that the Web space would be modeled better as a curved space. This non-Euclidean model, explains some previously unexplained phenomena and integrates many dissimilarity measures into a single notion of geodesic distance. To some extent, this answers the long existing open problem of under what conditions should a similarity measure be chosen. The non-Euclidean model establishes a theoretical basis and guidelines for designing a new type of similarity-based Web retrieval system.

References

1. Berge, C. *Topological Spaces Including a Treatment of Multi-Valued Functions, Vector Spaces and Convexity.* New York: Dover, 1997.
2. Bredon, G. E. *Topology & Geometry.* New York: Springer-Verlag, p. 69, 1995.
3. Cipra, B. *What's Happening in the Mathematical Sciences, Vol. 1.* Providence, RI: Amer. Math. Soc., p. 27, 1993.
4. Conlon, L. *Differentiable Manifolds: A First Course.* Boston, MA: Birkhuser, 1993.
5. Do Carmo, M. P. *Riemannian Geometry.* Boston, MA: Birkhauser, 1992.
6. Gray, A. "Metrics on Surfaces." Ch. 15 in *Modern Differential Geometry of Curves and Surfaces with Mathematica*, 2nd ed. Boca Raton, FL: CRC Press, 1997.
7. Jones, W.P. and Furnas, G.W. "Pictures of relevance: a geometric analysis of similarity measure". *Journal of the American Society for Information Science*, **38**, 420-446, 1987.
8. McGill, M.J., Koll, M., and Noreault, T. *An evaluation of factors affecting document ranking by information retrieval systems.* School of Information Studies, Syracuse University, Syracuse, New York 13210, 1979.
9. Munkres, J. R. *Topology: A First Course.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
10. Parsaye, K., Chignell, M., Khoshafian, S, and Wong, H. *Intelligent Databases: Object-Oriented, Deductive Hypermedia Technologies* John Wiley & Sons, Inc., 1989.
11. van Rijsbergen, C.J. *Information Retrieval*, London: Butterworth, 1979.
12. Salton, G. and McGill, M.J. *Introduction to Modern Information Retrieval.* New York, McGraw-Hill, 1983.
13. Singer, I.M. and Thorpe, John A. *Lecture Notes on Elementary Topology and Geometry.* Scott, Foresman and Company, p. 97, 1967.
14. Wang, Z.W. *An Analysis on Vector Space Model Based on Computational Geometry.* M.Sc. thesis, Computer Science Department, University of Regina, Regina, Canada, 1993
15. Weinstock, R. *Calculus of Variations, with Applications to Physics and Engineering.* New York: Dover, p. 26-28 and 45-46, 1974.

Advanced Replacement Policies for WWW Caching

Kai Cheng and Yahiko Kambayashi

Graduate School of Informatics, Kyoto University
Sakyo, Kyoto 606-8501, Japan
{chengk,yahiko}@isse.kuis.kyoto-u.ac.jp

Abstract. WWW caching necessitates advanced replacement policies that include sophisticated control logic and efficient contents management. This paper presents a constructive approach for the design and analysis of such advanced policies. Based on this approach, we develop a new caching policy, namely, PSS-W. Trace-driven simulations show PSS-W outperforms most contemporary policies in both hit rates and byte hit rates.

1 Introduction

Employing caches in the World Wide Web has been proven to be an effective approach to alleviating performance bottlenecks of web access [1]. Key to the effectiveness is how to decide and manage a suitable subset of requested data so as to maximize the hit ratio and other performance metrics. Strategies for this purpose, known as *cache replacement policies*, have been a focus of research for decades in paging scenarios [5].

However, due to the protocol restriction, WWW caching (web caching) has to deal with whole documents instead of data blocks. Web documents vary in sizes, costs, file types and sources. To cope with such complications, it is necessary to employ sophisticated control logic and special performance metrics. Furthermore, web caching is featured by large storage space, which, on the one hand, implies documents can be "probated" in cache for some long time, but on the other hand, it also means high overhead in maintaining large number of documents. The management of cache contents plays an important role in replacement policies of web caching.

In this paper, we propose a constructive approach for design and analysis of advanced replacement policies. According to this approach, an advanced replacement policy is constructed from several simple ones. The resulted policy is parameter-less, low overhead, able to deal with comprehensive factors and implement sophisticated control logic. Based on this framework, we develop a new replacement policy, namely Pyramidal Selection Scheme with aWard (PSS-W). Trace-driven simulations show that PSS-W outperforms most other algorithms in both hit rate and byte hit rate.

2 Related Work

Replacement policies for WWW caching have been extensively investigated in recent years. Charu Aggarwal et al in a recent paper [2] have surveyed and suggested classifying current replacement schemes for the Web into three categories: direct extensions of traditional policies, key-based policies and function-based policies.

However, the key-based policies are too simple and the priority between subkeys is not always ideal [8]. On the contrary, the function-based policies are too complicated and often suffer from heavy parameterization and high overhead. In practice, most well-known caching schemes turn out to be some hybrid ones where simple policies are organized to manage a segmented cache space. *Size-Adjusted LRU* is constructed from LRU and SIZE [2]; *Least Relative Value* (LRV) is an integration of SIZE, LFU and FIFO [7]; and the algorithm given by Pitkow and Recker is constructed from SIZE and LRU where the time since last access is rounded to days [6]. However, the constructions of these policies are mostly based on personal experiences. It is hard to tell what are the substantial differences between them.

3 Constructed Replacement Policies

A cache can be characterized as a software agent that consists of a finite storage space (*cache space*), a set of objects (*object space*), a replacement policy and a set of *constraints*. Replacement policy determines the contents and how to manage the contents of the cache. Constraints are additional conditions that a cache has to satisfy. We distinguish three classes of constraints: *admission constraints*, *consistency constraints*, and *miscellaneous constraints* such as comfortable level of cache space.

3.1 Framework of Constructed Replacement Policies

As described previously, the diversity of web documents and large scale of cache space complicate the design of new replacement policies. Thus, we propose a constructive approach to design and analysis of advanced replacement policies. As depicted in Fig. 1, an advanced policy is constructed from several simple policies with (1) a set of *classification rules*; (2) few *unit caches* and (3) a *central cache*

Classification Rules

Classification rules are based to allocate objects and space among all unit caches. An object can belong to only one unit cache at a specific time. The classification rule may be a simple function, or a sophisticated set of logic rules. For example

1. *If object X has a specific size, then cache X in unit* $\lfloor log_2(size) \rfloor$
2. *If object X is from Japan and content is about baseball, and its type is picture and size is bigger than 24KB, then keep X in unit 1*

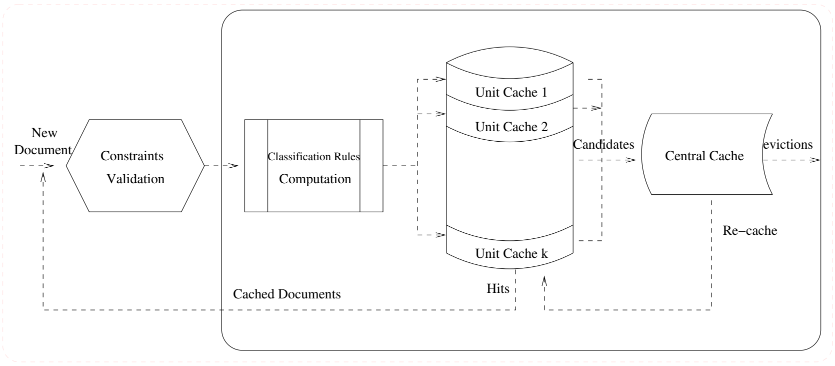


Fig. 1. Framework of constructed replacement policies

Unit Caches

A unit cache is an independent cache that has its own object space, cache space, caching policy and constraints. Which unit an object belongs to is determined by classification rules. Thus, unit cache can use simpler caching policy. For example, if classification rule is rule 1 as in previous example, then objects in the same unit cache have the similar sizes. As a result, the unit cache need not take size into account. For this reason, simple policies such as LRU, FIFO, and SIZE [8] are often used in this case.

Central Cache

A central cache manages the candidate evictions of unit caches and makes final decision. Central cache may be without its own cache space and since there are only a limited number of eviction candidates, the central caching policy can be very elaborate, with many factors in consideration. The caching decisions have several types: *eviction*, *re-cache*, and *probation*.

An eviction object will be purged at once. However, if an object is selected to re-cache or to probate, it will remain in cache. The distinction between re-cache and probation is: an object to be re-cached will be cached at once in a certain *unit cache*, while a probation object will be held by central cache in its own cache space and the final decision is expected to make in the next turn.

3.2 Analysis of Existing Caching Policies

From the constructive point of view, we analyze the caching policies surveyed in Section 2. Results are listed in Table 1, where $nref$ represents the reference frequency of an object, $size$ is the object size and $atime$ is the elapsed time since the object's last access.

Table 1. Comparison of Various Constructive Policies

Algorithm	Rules	U	Unit Caches	Central Cache	Factors
Segmented LRU	$nref$	2	LRU	Re-cache in D_{i-1}	atime, nref*,
Size-Adjusted LRU	$\lfloor log_2(size) \rfloor$	24	LRU	max($size * atime$)	atime, size*
Least Relative Value	$nref$	10	SIZE, FIFO	max(lrv)	atime, nref* size
Pitkow/Recker	$day(atime)$	7	SIZE, LRU	max($day(atime)$)	atime*,size
PSS-W	$\lfloor log_2(\frac{size}{nref}) \rfloor$	24	LRU	max($\frac{size*atime}{nref}$)	atime, nref*, size*

Segmented LRU

Segmented LRU [4] use a classification rule based on reference frequency ($nref$). This is because objects with at least two accesses are for more popular than those with only one access. Cache space is partitioned to two segments: *probationary segment* and *the protected segment*. Objects with at least two accesses are kept in the protected segment, while new objects (with only one access) are first faulted into the probationary segment. When a probationary object gets one more reference, it will change to the protected segment.

Both unit caches are managed as LRU queues. When the whole cache space becomes full, the least recently used object in the probationary segment will first be replaced. The protected segment is finite in size and when it gets full, the overflowed will be *re-cached* in probationary segment.

Size-Adjusted LRU [2]

The Size-Adjusted LRU chooses a victim by sorting all objects in cache in terms of the cost-to-size ratio, $1/(size \cdot atime)$. It then greedily discards those with least cost-to-size ratios from the cache. Size-Adjusted LRU uses a pyramidal selection scheme (PSS) to manage cache space and object space.

The classification rules is based on $\lfloor log_2(size) \rfloor$, that is to say, objects within a same group are similar in sizes. Each group is maintained using a LRU mechanism. Though a hit will make the object move to the most recently used end, but an object can not move to another group. The computation of $1/(size \cdot atime)$ is done only to a limited set of least recently used objects from all nonempty groups and the object with largest $(size \cdot atime)$ will be purged from the cache.

4 Pyramidal Selection Scheme with Award (PSS-W)

Object size has been considered as one of the most important features of web caching [3,**?**]. Reference frequency is a strong indicator to the overtime popularity of web objects [7]. However, Segmented LRU takes advantage of the popularity information but fail to distinguish object sizes ($size$); whereas Size-Adjusted

LRU generalizes LRU to handle variable sizes but fails to utilize reference frequency ($nref$).

We extend the cost-to-size ratio in Size-Adjusted LRU by incorporating frequency ($nref$). Sine each hit will reasonably increase the cost savings, we use ratio of ($nref/atime$) to size and choose the object with minimum value of ($nref/(size \cdot atime)$). Based on this benefit-to-cost ratio, objects with more references are given larger benefit-to-cost ratio and can stay more time before aged out.

Now we can construct this new replacement policy. First, to simplify unit cache, the classification rule is chosen to be $\lfloor \log_2(size/nref) \rfloor$. Since, $nref$ changes with each hit, when $\lfloor \log_2(size/nref) \rfloor$ changed, an object may move to another unit cache. Compared to Size-Adjusted LRU, the $size/nref$ results in a small value in adjusting LRU choice and long stay in cache for an object with larger $nref$. For this reason, we call the new policy *Pyramidal Selection Scheme with aWard* (PSS-W).

Each unit cache is managed using a LRU policy. The ($nref/(size \cdot atime)$) is computed for the eviction candidates from all nonempty groups, purging the object with least ($nref/(size \cdot atime)$). The number of units is 24, because the objects larger than 16MB(2^{24}B) are very rare.

5 Performance Evaluation

Through trace-driven simulations, we evaluate the replacement polices listed in Table 1. The dataset used in our experiments is a one-week top level caching proxy traces publicly available (ftp://ircache.nlanr.net/Traces/). This dataset contains 1,848,319 requests with total 21.0 GB Web data, where unique data is 15.9 GB with a maximum hit rate 0.228 and byte hit rate 0.245.

The candidate replacement policies to be evaluated are LRV (Least Relative Value), SLRU (Segmented LRU), Pitkow(Pitkow/Recker algorithm) together with PSS-W. The results shown in Fig. 2. Plots in the left side depict the hit rates. The hit rate of PSS-W is much better than the rest. Plots in the right size are byte hit rates. PSS-W can achieve quite high byte hit rate.

Furthermore, the time complexity of PSS-W in servicing each request is a small constant in maintaining LRU queues and computing and comparing cost-to-size ratios. Thus, PSS-W is an ideal replacement policy for web caching.

6 Conclusion and Future Work

To cope with the complications in design and analysis of advanced replacement policies for web caching, we have proposed a constructive framework. This framework has been proven helpful in analysis of various current policies, and useful in making the design of PSS-W, a new efficient caching policy for Web caching. However, this framework has several aspects to be completed or improved. One of our future works is to study advanced classification rules since classification rules play a key role in construction of an advanced policies.

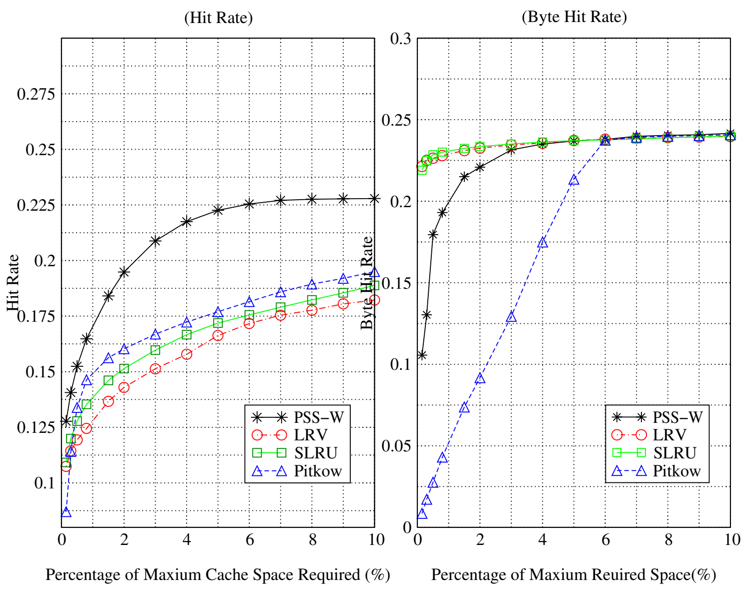


Fig. 2. Hit Rates and Byte Hit rates of various policies

References

1. Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Wililams, and Edward A. Fox. Caching Proxies: Limitations and Potentials. In *Proceedings of the Fourth International WWW Conference*, 1995. 239
2. Charu Aggarwal, Joel L. Wolf, and Philip S. Yu. Caching on the World Wide Web. *IEEE transactions on knowledge and data engineering*, 11(1), 1999. 240, 242
3. Pei Cao and Sandy Irani. Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems*, pages 193–206, December 1997. http://www.cs.wisc.edu/cao/publications.html. 242
4. Ramakrishna Karedla, J. Spencer Love, and Bradley G. Wheery. Caching Strategies to Improve Disk System Performance. *IEEE Computer*, 27(3):38–46, March 1994. 242
5. T. H. Merrett and Yahiko Kambayashi. Join scheduling in a paging environment using the consecutive retrieval property. In *Proceedings of International Conference on Foundations of Data Organization and Algorithms (FODO)*, pages 323–347, 1981. 239
6. James E. Pitkow and Margret M. Recker. A Simple Yet Robust Caching Algorithm Based on Dynamic Access Petterns. Technical Report VU-GIT-94-39, GVU Technical Report, 1994. 240

7. Luigi Rizzo and Lorenzo Vicisano. Replacement Policies for a Proxy Cache. Technical report rn/98/13, University College London, Department of Computer Science, Gower Street, London WC1E 6BT, UK, 1998. http://www.iet.unipi.it/ luigi/caching.ps.gz. 240, 242
8. Stephen Williams, Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, and Edward A. Fox. Removal Policies in Network Caches for World-Wide Web Documents. In *Proceedings of ACM SIGCOMM'97*, pages 293–305, Stanford, CA, August 1997. 240, 241

Extending Rectangle Join Algorithms for Rectilinear Polygons[⋆]

Hongjun Zhu, Jianwen Su, and Oscar H. Ibarra

University of California at Santa Barbara

Abstract. Spatial joins are very important but costly operations in spatial databases. A typical evaluation strategy of spatial joins is to perform the join on approximations of spatial objects and then evaluate the join of the real objects based on the results. The common approximation is the minimum bounding rectangle. Minimum bounding rectangles are coarse approximations of spatial objects and may cause a large number of "false hits". In this paper, we consider a more general form of approximation with rectilinear polygons for spatial objects in the context of spatial join evaluation. A naive approach is to decompose rectilinear polygons into rectangles and use an exisiting rectangle join algorithm. This may require additional cost for sorting, index construction, and decomposition and prohibits the join evaluation to be pipelined. The main contribution of the paper is a technique for extending plane sweeping based rectangle join algorithms to perform a spatial join on rectilinear polygons directly. We show that the join of two sets of rectilinear polygons can be computed in $O(bN \log_b \frac{N}{b} + \ell^2 k)$ IOs directly, where N is the total number of boundary points in each input set, ℓ the maximum number of boundary points of a rectilinear polygon, b the page size, and k the number of rectilinear polygon intersections. When the rectilinear polygons are y-monotone, the IO complexity becomes $O(bN \log_b \frac{N}{b} + \ell k)$.

1 Introduction

Spatial joins are important and useful operations in spatial databases, constraint databases, GIS, etc. [14,12,13]. A spatial join links together tuples that have overlapping spatial values. For example, the query "find all cities with lake" can be a spatial join of a relation with the geographical information of cities and another with that of lakes. Similar to join operations in relational databases, spatial joins are very expensive to evaluate. Furthermore, because spatial objects are very different from and are much larger in size than traditional data, efficient evaluation of spatial joins can be even more difficult. In this paper, we investigate efficient evaluation of spatial joins in spatial or constraint databases.

Because spatial data can be very large, approximations of spatial objects are frequently used in spatial query evaluation to reduce accesses to the spatial

[⋆] Support in part by NSF grants IRI-9700370 and IIS-9817432.
Authors' address: Dept. of Computer Science, University of California, Santa Barbara, CA 93106, USA. {`hongjunz`, `su`, `ibarra`}`@cs.ucsb.edu`

objects. Typically, the evaluation of a spatial join employs two steps [16]: (1) a *filter* step that performs a spatial join on approximations of the objects; and (2) a *refinement* step that checks whether the objects discovered by the filter step actually intersect. A great deal of research has been done to speed up the filter step and to reduce the size of the result it produces. Many algorithms [16,18,17,20,3,7,1,19,22,15,11,2,25] focus on the filter step and use the *minimum bounding rectangle* (*MBR*), the smallest rectangle containing a spatial object, as an approximation of the object. In this paper, we call these algorithms *rectangle join* algorithms.

Rectangle join evaluation has been the focus of the study on spatial join for a long time. The existing algorithms can be classified into two categories. The first category includes algorithms that do not need any (spatial) index structures in addition to the input rectangles. In this category, the algorithms in [16,17,18] use a space-filling curve, called z-ordering; the algorithm in [3] transfers rectangles into points in the 4 dimension space and uses grid-files to compute the join; both algorithms in [19,15] use a partition based method. In addition, [2] uses a distributed-sweeping technique [9] to achieve IO efficiency. The algorithms in the second category requires additional index structures. The algorithm in [20] uses join indices [24] that are actually partially precomputed join result. R^*-trees [4] are used in the algorithms of [7,11]. [1] extended segment trees [5] for external memory to improve IO performance. [22] used a combination of a space-filling curve and filter trees that extends the idea of quad-trees (see [21]). In [25], we developed interval B^+-trees that combine segment trees and B^+-trees and an algorithm that uses interval B^+-trees and the plane sweeping technique.

MBRs are coarse approximations and intersection of MBRs does not guarantee intersection of actual spatial objects. Although the filter step can be sped up by using MBRs, the refinement step may still access a large number of objects that are not part of the final join result. These unnecessary accesses reduce the performance of join evaluation. To overcome this problem, other approximation methods such as minimum bounding circles [7] and raster approximations [27] are used to provide an initial screening at the beginning of the refinement step. In [6], the additional step is to determine intersection of finer approximations (than MBRs) such as convex hulls and minimum bounding n-corner convex polygons for each pair of objects generated by the filter step. In [27] an algorithm was developed to identify actual or possible object intersections from raster approximations. It is desirable to further improve the performance of join evaluation by modifying the filter step to use finer approximations instead of MBRs.

In this paper, we consider rectilinear approximation, which approximates each spatial object by a rectilinear polygon containing it. Since rectilinear approximations are finer than MBRs, the objective of this paper is to develop IO efficient join algorithms for rectilinear polygons. We show that existing rectangle join algorithms can be naturally extended for rectilinear polygons.

A naive approach is to decompose the rectilinear polygons into rectangles. Then, every existing rectangle join algorithm can perform a join on the decomposed rectangles. However, the algorithms of [16,18,17,20,3,7,10,19,15,22,11] have quadratic IO complexity in the worst case. The algorithms in [1,2,25] have better IO complexity and are appropriate to perform the rectangle join. For example, after the decomposition and necessary preprocessing, the join of two sets of rectilinear polygons can be computed in $O(bN \log_b \frac{N}{b} + \ell^2 k)$ IOs using the algorithm in [25], where N is the total number of boundary points in each input set, ℓ the maximum number of boundary points of an input rectilinear polygon, b the page size, and k the number of pairs of rectilinear polygons that intersect.

Unfortunately, there are problems with this decomposition based approach. Decomposition and preprocessing require additional IO cost. If indices and/or sorting are assumed in the rectangle join part, constructing the indices and/or sorting can also add to the cost (IO and space). More importantly, the decomposition and rectangle join steps are not tightly integrated; a consequence is that they cannot be pipelined and are hard to be used in a database environment as most of query processing is done in a pipelined fashion.

A main contribution of this paper is a technique to extend plane sweeping based rectangle join algorithms [1,25] to perform an IO efficient filter step using rectilinear polygons instead of MBRs, i.e., a spatial join on rectilinear polygons directly. We illustrate this technique by extending the interval B$^+$-tree based join algorithm of [25]. We show that the join of two sets of rectilinear polygons can be computed in $O(bN \log_b \frac{N}{b} + \ell^2 k)$ IOs directly, where N, ℓ, b, k are the same as above. A similar statement can be made for the algorithm in [1]. For the case where the rectilinear polygons are y-monotone, we show that the join problem can be solved in $O(bN \log_b \frac{N}{b} + \ell k)$ IOs.

This paper is organized as follows. Section 2 introduces rectilinear approximations for spatial objects. Section 3 discusses the decomposition based approach. Section 4 presents the plane sweeping based approach. Section 5 give a short conclusion. Due to space limitation, detailed proofs are omitted.

2 Rectilinear Approximations

In the two-step join evaluation, the filter step performs join on the approximations and eliminates those objects that can not contribute to the join result. To improve the performance of the spatial join, it is very important to improve the performance of the filter step and to reduce the size of the result generated by the filter step.

Most join algorithms use MBRs in the filter step. MBR approximations can be indexed using index structures such as R-trees and quad-trees. Consequently, they can be processed efficiently and used to speed up the filter step of the spatial join evaluation. But clearly, MBRs are coarse approximations of spatial objects and it is very likely that two objects do not intersect while their MBRs do (a *false hit*). False hits force accesses to spatial objects that are not part of the join result and slow down the refinement step of join evaluation. By reducing the number of false hits, the performance of spatial join evaluation can be enhanced. It becomes interesting to develop suitable approximation methods for spatial joins.

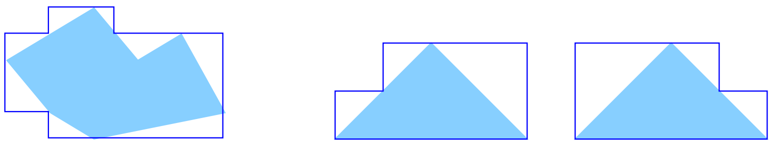

(a) 10-corner rectilinear approximation (b) Two rectilinear approximations

Fig. 1. Rectilinear approximations

The *quality* of an approximation is defined as the ratio of the area of an actual spatial object to that of the approximation. [1] An approximation of quality 1 is identical to the actual object. The closer the quality of an approximation is to 1, the finer it is. The quality of a "conservative" approximation (i.e. containing the object) is ≤ 1. In this case, an approximation of lower quality increases the number of false hits, while an approximation of higher quality can reduce the number.

In [7,6], six conservative approximation approaches are studied: MBR, rotated MBR, minimum bounding circle, minimum bounding ellipse, convex hull, and minimum bounding n-corner, and compared based on the false hits over several data sets in the context of spatial join. As an extreme case, raster approximation of polygons was used in spatial join evaluation [27]. The join strategies used in these studies are to evaluate join on MBRs of spatial objects first, determine the intersections of finer approximations based on the MBR join results, and perform the refinement step for possible intersecting object pairs. The studies show that rotated MBR, minimum bounding 5-corner, and raster approximations improve the overall performance of spatial join evaluation. Clearly, further optimizations can be made if the MBR filtering and refined filtering with such finer approximations can be combined such that the filter step uses these approximations directly without using MBRs.

In this paper, we consider a "rectilinear" approximation scheme, which combines the simplicity of MBR and the high quality of raster approximation. A *rectilinear* (or *orthogonal*) *polygon* is a polygon whose boundaries are parallel to the coordinate axes. A rectilinear polygon is *ℓ-corner* if it has ℓ boundary points (ℓ must be even). Given a 2-dimensional simple polygon S, we define an ℓ-corner rectilinear approximation of S as following:

Definition 2.1. Let S be a simple polygon and $\ell \geq 0$ an even integer. An *ℓ-corner rectilinear approximation of S*, denoted by $\bar{S}^{\ell}$, is an ℓ-corner rectilinear polygon containing S which has the highest quality among all ℓ-corner rectilinear polygons containing S.

Fig. 1(a) shows an object and a 10-corner rectilinear approximation. It is important to note that the rectilinear approximation of a given simple polygon S is not necessarily unique, but the number of the rectilinear approximations of S is always finite. Fig. 1(b) shows two different 6-corner rectilinear approximations of the same polygon. (The problem of how to generate rectilinear approximations

[1] In [7], the approximation quality is defined as the ratio of the sum of the area of the object and the false area of the approximation with respect to inside and outside of the object to the area of the actual spatial object.

is not the focus of this paper.) In the remainder of the paper, we assume that rectilinear approximations of spatial objects are always available and consider the problem of efficient join evaluation of rectilinear polygons.

An ℓ-corner rectilinear polygon is represented by a sequence of its boundary points $p_1, ..., p_\ell$ in counterclockwise order, where p_1 is the leftmost boundary point with the smallest y coordinate. The *size* of a rectilinear polygon is the number of its boundary points and the *size* of a set of rectilinear polygons is the sum of the sizes of all polygons. Let b be the page size in the remainder of the paper.

3 Decomposition Based Join Algorithms

In the following two sections, we extend rectangle join algorithms to evaluate joins of rectilinear polygons. In this section, we introduce a naive solution that decomposes rectilinear polygons into rectangles and then applies a rectangle join algorithm on the rectangles generated.

Let r, s be two sets of n ℓ-corner rectilinear polygons. The join problem of r and s is to report all pairs of polygons in r and s (resp.) that intersect. The join can be evaluated in two steps: to perform a decomposition and a rectangle join.

The first step decomposes every rectilinear polygon in r and s into one or more (possibly overlapping) rectangles, resulting in two sets of rectangles $rect(r)$ and $rect(s)$ (resp.).

Using an algorithm in [8], we can partition an ℓ-corner rectilinear polygon into no more than $(\frac{\ell}{2} - 1)$ (non-overlapping) rectangles. Fig. 2 gives an ℓ-corner rectilinear polygon whose decomposition must contain at lease $(\frac{\ell}{2} - 1)$ rectangles no matter how it is done. In fact, the following holds.

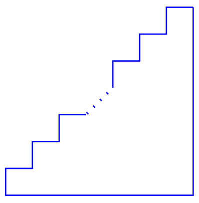


Fig. 2.

Proposition 3.1. Each ℓ-corner rectilinear polygon can be partitioned into at most $(\frac{\ell}{2} - 1)$ rectangles. Furthermore, this bound is tight.

It follows that the decomposition phase will produce no more than $n \times (\frac{\ell}{2} - 1)$ rectangles in each of $rect(r)$ and $rect(s)$, since r, s each contains n polygons.

The second step applies a rectangle join algorithm on $rect(r)$ and $rect(s)$. All existing rectangle join algorithms can be used for this step. Fig. 3 enumerates these algorithms. The first column lists categories of algorithms according to their key techniques. The second gives the worst case IO complexity for each category, where N is the total number of rectangles, b the page size, m the number of pages available in memory, and k the number of rectangle intersections. The third shows whether the algorithms use spatial index structures. the algorithms. And the last column presents the basic requirements on the input data by the algorithms.

In order to evaluate join for rectilinear polygons efficiently, we should choose an IO efficient rectangle join algorithm. The algorithms using z-orderings [16,17,18], grid-files [3], partition based method [19,15], join indices [20], R^*-trees [7,11], and filter trees [22] all have quadratic worst case IO complexity and are not desirable.

Algorithms using	Worst Case IO	Spatial Index	Input Requirement
z-orderings [16,17,18]	$O(N^2)$	no	z-orderings computed from MBRs
grid-files [3]	$O(N^2)$	no	grid-files on the 4-d points corresponding to MBRs
partition [19,15]	$O(N^2)$	no	partitions of MBRs
distributed sweeping [2]	$O(N \log_m \frac{N}{b} + \frac{k}{b})$	no	MBRs sorted on their x and y projections
join indices [20]	$O(N^2)$	yes	grid-file of the join result
R^*-trees [7,11]	$O(N^2)$	yes	none
external segment trees [1]	$O(N \log_m \frac{N}{b} + \frac{k}{b})$	yes	MBRs sorted on their y projections
filter trees [22]	$O(N^2)$	yes	none
IB$^+$-tree [25]	$O(bN \log_b \frac{N}{b} + k)$	yes	MBRs sorted on their y projections

Fig. 3. Summary of Existing Rectangle Join Algorithms

The algorithms using distributed sweeping [2], external segment trees [1], and interval B$^+$-trees (IB$^+$-tree in short) [25] have desirable IO complexity and are candidates for this step. In the following, we use the algorithm rjoin in [25] as an example to analyze the IO complexity of this join evaluation approach. Similar statements can be made for algorithms in [1,2].

Let r and s be two sets of n ℓ-corner rectilinear polygons. Both r and s have the same size $N = n \times \ell$. The decomposition produces two sets of rectangles, each containing $n \times (\frac{\ell}{2} - 1) = O(N)$ rectangles. The algorithm rjoin uses a new index structure *interval B$^+$-tree* and applies the plane sweeping technique [23]. From Fig. 3, we can see that the IO operations needed by the rectangle join step using rjoin is $O(bN \log_b \frac{N}{b} + k')$, where k' is the number of pairs of intersecting rectangles. In the worst case (as shown in Fig. 4), there are $O(\ell^2)$ rectangle intersections among the rectangle decompositions of two ℓ-corner rectilinear polygons. Let k be the total number of rectilinear polygon intersections. It follows that the total number of rectangle intersections is $O(\ell^2 k)$. Thus we have the following theorem.

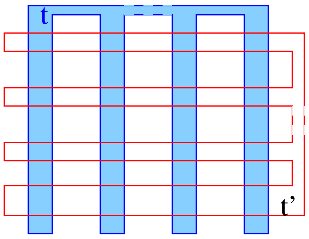


Fig. 4. Two polygons

Theorem 3.2. [25] *Let r, s be two sets of ℓ-corner rectilinear polygons and N the size of r and s. We further assume that decomposed rectangles and IB$^+$-tree indices on them are available, and the rectangles are sorted on their projections on the y axis. The join of r and s can be evaluated within $O(bN \log_b \frac{N}{b} + \ell^2 k)$ IOs using* rjoin*, where k is the number of pairs of rectilinear polygons that intersect.*

A polygon is called *y-monotone* if for any line l parallel to the x axis the intersection of the polygon with l is a line segment, a point, or empty. In Fig. 4, polygon t' is y-monotone while polygon t is not. It is easy to show that there are $O(\ell)$ rectangle intersections among the rectangle decompositions of two ℓ-corner y-monotone rectilinear polygons. Under the same assumption as the general case, we can then show that in the case where all the rectilinear polygons are y-monotone, the join can be evaluated in $O(bN \log_b \frac{N}{b} + \ell k)$ IOs, where k is the number of pairs of rectilinear polygons that intersect.

There are problems with this decomposition based approach. First, the previous IO complexity analysis does not include the cost of the entire join algorithms. Theorem 3.2 does not take into account the cost of decomposition and preprocessing of the rectangles generated by decomposition. No matter which (IO efficient) rectangle join algorithm (the ones in [1,2,25]) is used, the decomposition of rectilinear polygons must be completed first. Even if we precomputed the decomposition and maintain rectangles that constitute the rectilinear polygons in databases, we still need to perform decomposition whenever an update happens. Also, the algorithms in [1,25] require index structures to be constructed based on all rectangles. The input rectangles must be sorted in all three algorithms [1,2,25]. The construction of index structures and sorting requires additional storage space and IO cost.

Second, the two steps of this approach are not tightly integrated; a consequence is that they cannot be pipelined and hard to be used in a database environment as most of query processing is done in a pipelined fashion.

In sum, it is very desirable to have an algorithm that can process the rectilinear polygons directly, which will be discussed in the next section.

4 Plane Sweeping Based Join Algorithms

In the section, we extend plane sweeping based rectangle join algorithms [1,25] for evaluating spatial joins of rectilinear polygons directly. We use the algorithm rjoin [25] as an example to demonstrate the key ideas and techniques. We show that the rectilinear join algorithm extended from rjoin can evaluate a join of two sets of ℓ-corner rectilinear polygons in $O(bN \log_b \frac{N}{b} + \ell^2 k)$ IOs, where N is the size of each input set, and k the number of polygon intersections. A similar statement can be made for the algorithm in [1].

In order to describe the extension, we first give a brief description of the rectangle join algorithm rjoin.

Algorithm rjoin applies plane sweeping technique and uses IB$^+$-trees as index structures. The plane sweeping technique is widely used in the field of computational geometry. The idea is to sweep a line across the plane and perform some actions at appropriate points. The IB$^+$-tree index structure was developed for IO efficient interval management [25].

Let r and s be two sets of rectangles. In rjoin, the sweep line is a horizontal line that lies below the lowest rectangle in r and s at the beginning. During the execution, the sweep line moves up. A rectangle in r or s is called *active* if it intersects the current sweep line. Two sets are used to keep track of the active rectangles in r and s, and are called *active set* of r and of s, resp. Both r and s have IB$^+$-tree indices on the x axis, resp. The two IB$^+$-trees are basically B$^+$-trees on the endpoints of the x projections of rectangles in r and s (resp.)

with auxiliary structures. The auxiliary structures are used to maintain the two active sets. Initially, both active sets are empty.

During the sweeping, when the sweep line encounters the lower boundary of a rectangle t, t becomes active and is inserted into the corresponding active set. Also at this point, two different searches need to be performed to report all active rectangles in the opposite active set intersecting t. First, a *range search* finds all active rectangles in the opposite active set whose projections on the x axis are contained in the x projection of t. Second, a *point search* finds all active rectangles in the opposite active set whose x projections contain at least one endpoint of the x projection of t. These two searches are guaranteed to find all active rectangles in the opposite active set that intersect t. When the sweep line reaches the upper boundary of t, t becomes *inactive* and is deleted from the corresponding active set. The sweep line keeps moving up and appropriate operations are executed at boundaries of rectangles until the sweep line leaves the upper boundary of the highest rectangle in r and s.

Example 4.1. Let r and s be two sets of rectangles. Parts of r and s are shown in Fig. 5, where t_1, t_2, t_3, t_4 are in r and t'_1, t'_2, t'_3 in s. At the beginning, the sweep line is below the lowest rectangle t_2. When the line meets the lower boundary of

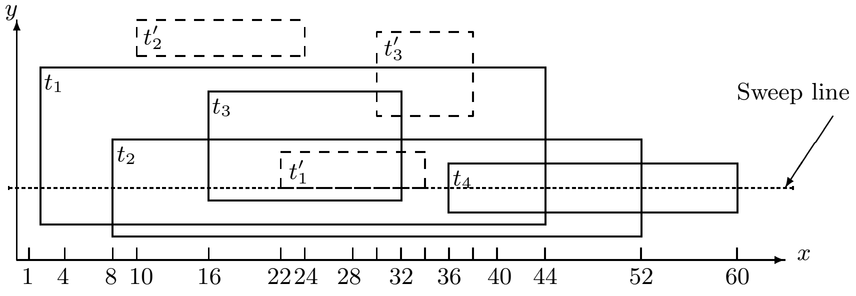


Fig. 5. Rectangles in r and s

t'_1 (Fig. 5), t_1, t_2, t_3 and t_4 are active, and are stored in the active set of r. The range search on the active set of r finds t_3 and the point search reports t_1, t_2 and t_3. Rectangle t'_1 becomes active and is stored in the active set of s. When the sweep line reaches the upper boundary of t'_1, t'_1 is deleted from the active set of s.

A key part of the above algorithm is the active set. During the execution of the algorithm, the intersection of the horizontal sweep line with an active rectangle t is a horizontal line segment whose x projection is always the same as the x projection of t. The active sets in fact store the x projections of the current sweep line with rectangles in r and s (resp.) using IB$^+$-trees.

We now consider rectilinear polygons. Let r and s be two sets of ℓ-corner rectilinear polygons. Suppose we sweep a horizontal line through all polygons in r and s. During the sweeping, a rectilinear polygon either intersects the current sweep line or does not intersect it at all. We call a rectilinear polygon *active* if it intersects the current sweep line. Unlike the rectangle case, the intersection of an active rectilinear polygon t and the sweep line could be a set of line segments, and its x projection may change before the sweep line leaves t. Fig. 6(a) shows two positions of the sweep line and a polygon t. When the sweep line is at the lower position, the intersection of t and the sweep line is one line segment. When the sweep line moves to the upper position, the intersection becomes two line segments. We observe that the x projection of the intersection of the sweep line and an active rectilinear polygon t only changes when the sweep line encounters a horizontal boundary of t. In fact, we can view the rectilinear polygon t as a set of rectangles generated by cutting t with horizontal lines overlapping at least one horizontal boundary of t. Fig. 6(b) shows 4 rectangles t^1, t^2, t^3, and t^4 that compose t. When the sweep line reaches the lowest horizontal boundary of t, it in fact meets rectangle t^1. When the sweep line moves to the position of Fig. 6(b), it leaves t^1 and enters two new rectangles t^2 and t^3 of t.

Based on the above observation, we extend rjoin for joins of rectilinear polygons. The following example illustrates the basic idea of the extension.

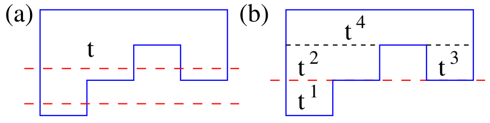


Fig. 6. Active rectilinear polygons

Example 4.2. Let r and s be two sets of 8-corner rectilinear polygons. Parts of r and s are shown in Fig.7, where t_1, t_2, t_3 are in r and t'_1, t'_2 in s. The dotted lines are the sweep line at different positions. At the beginning, the sweep line is below t_3 which has the lowest horizontal boundaries. Two active sets are used to maintain the x projections of the intersections of the current sweep line and active polygons in r and s, resp.

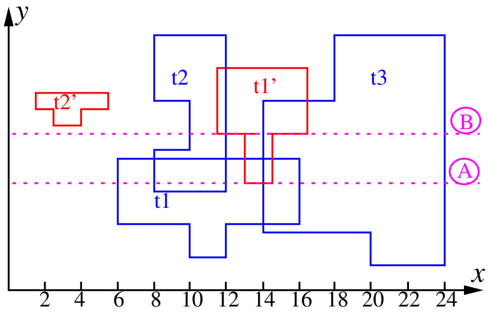


Fig. 7. Rectilinear polygons in r and s

When the sweep line reaches the lowest boundary of t_3, t_3 becomes active. The x projection of the intersection of the sweep line and t_3, $(20, 24)$, is inserted into the active set of r. (Note that all intervals are closed even though we use "(" and ")" in the representation.)

When the sweep line reaches position A, t'_1 becomes active. Polygons t_1, t_2 and t_3 are active, and the intervals stored in the active set of r for them are: $(6, 16)$, $(8, 12)$, and $(14, 24)$, resp. The x projection of the intersection of the sweep line and t'_1, $(13, 15)$, is inserted into the active set of s, and a range search and a point search are executed to find all intervals in the active set of r that intersect $(13, 15)$. As a result, t_1 and t_3 are reported as intersect with t'_1.

When the sweep line moves to position B, the x projection of the intersection of the sweep line and t'_1 changes from $(13, 15)$ to $(11, 17)$. Interval $(13, 15)$ is then removed from the active set of s and $(11, 17)$ is inserted. A range search and a point search is performed for $(11, 17)$. At this moment, only t_2 and t_3 are active, and the intervals corresponding to them in the active set of r are $(10, 12)$ and $(14, 24)$. As a result, t_2 and t_3 are reported as intersect with t'_1.

Theorem 4.1. *Let r, s be two sets of ℓ-corner rectilinear polygons and N the size of r and s. Assume further that there are IB$^+$-tree indices of r and s on the x axis. The algorithm* rjoin *can be extended to compute the join of r and s directly in $O(bN \log_b \frac{N}{b} + \ell^2 k)$ IOs, where k is the number of pairs of rectilinear polygons that intersect.*

We now discuss the key techniques needed in the proof of the theorem.

Let r, s be two sets of n ℓ-corner rectilinear polygons, and k the number of rectilinear polygon intersections. The size of r and s is $N = \ell \times n$. Each rectilinear polygon in r and s is represented as a sequence of its boundary points (counterclockwise). Same as in rjoin, we assume that both r and s have IB$^+$-tree indices T_r and T_s on the x axis. The IB$^+$-trees are essentially B$^+$-trees on the x coordinates of all boundary points of rectilinear polygons in r and s (resp.) with auxiliary structures. Initially, those structures are empty and T_r and T_s are just two B$^+$-trees, each occupies $O(\frac{N}{b})$ pages. The auxiliary structures of IB$^+$-trees are used to maintain the active sets. The insertion and deletion of an interval from an active set are implemented by the "mark" and "unmark" operations on the corresponding IB$^+$-tree, resp. A "report" operation then performs the range search and point search on the IB$^+$-tree. In [25] it was shown that the mark and unmark operations need $O(b \log_b \frac{N}{b})$ IOs, and the report operation takes $O(\log_b \frac{N}{b} + k_i)$ IOs, where k_i is the number of intervals in the IB$^+$-tree that intersect a given interval.

During the join evaluation, we sweep along the y dimension. In order to do so, all boundary points in r and s (resp.) are sorted based on their y coordinates. Sorting can be avoided if we keep two sorted lists L_r, L_s for r, s (resp.). Each entry in L_r has the form (K, ID), where K is a key value, ID is a list of identifiers (ids) of rectilinear polygons $t \in r$ such that there are boundary points of t whose y coordinates are K. Each id of t in ID is associated with a list of boundary points p of t whose y coordinates are K. Furthermore, all the points are stored in the order they appear in the representation of t. Entries in L_s are similar. Usually these lists can be either stored or obtained from an index structure. The sweeping is then accomplished by a sequential scan of all keys in the lists L_r, L_s. It is easy to verify that the sweeping takes no more than $O(N)$ IOs.

When the sweep line reaches position $y{=}K$, the entry (K, ID) in L_r or L_s is processed. The following actions are taken for each tuple t in ID:

- The x projection of the intersection of the current sweep line and t is computed using intervals that are currently marked in the corresponding IB$^+$-tree for t and the horizontal boundaries formed by the boundary points in the list associated with t. The intervals that are currently marked in the corresponding IB$^+$-tree for t are unmarked and the new intervals are marked.
- For each interval I newly marked in the IB$^+$-tree, the report operation is executed to find all intervals marked in the IB$^+$-tree of the opposite set that intersect I. The corresponding polygons are reported as intersecting.

A key step here is to compute the intersection of a rectilinear polygon and the current sweep line. Let t be the rectilinear polygon under consideration and I^* the set of intervals marked in the corresponding IB$^+$-tree for t. The only information we have are I^* and the boundary points that locate on the current sweep line. We note that the x projection of the intersection may overlap intervals in I^*. We also observe that among the boundaries of t that overlap the current sweep line, the ones that bound t from below are part of the new intersection while those that bound t from above are not. Therefore, We can first extract the x projections of the horizontal boundaries formed by the boundary points in the list associated with t and store the projections of the boundaries that bound t from below and above in two sets I^{below} and I^{above}, resp. Then we can compute the x projection of the current intersection using I^{below}, I^{above}, and I^*.

The extraction of I^* can be done in $O(\log_b \frac{N}{b})$ by changing the structure IB$^+$-tree slightly such that the intervals currently marked in the tree for t is stored along with the id of t. The computation of the x projection of the new intersection is performed in memory. And it is not hard to see that for each horizontal boundary line of t, only one mark, unmark, and report operation are executed on the IB$^+$-tree.

From above discussion, we can show that the join of r and s can be computed in $O(bN \log_b \frac{N}{b} + \ell^2 k)$ IOs directly.

In the case where all input rectilinear polygons are y-monotone, we have the following result:

Theorem 4.2. *Let r, s be two sets of ℓ-corner y-monotone rectilinear polygons and N the size of r and s. Assume further that there are IB$^+$-tree indices of r and s on the x axis. The join of r and s can be computed directly in $O(bN \log_b \frac{N}{b} + \ell k)$ IOs, where k is the number of pairs of rectilinear polygons that intersect.*

5 Conclusions

In this paper, IO efficient join algorithms for rectilinear polygons are developed by extending the algorithms for rectangles. In [26], we presented an IO efficient join algorithm for trapezoids and minimum bounding 5-corners. There are many interesting problems left open. For example, there are no efficient join algorithms for approximations such as minimum bounding circles and convex hulls, although they are better approximations than MBRs. It will be very useful to allow these approximation methods in efficient join evaluation. On the other hand, it is also interesting to study the trade-off between these sophisticated approximation techniques and false hits in the overall performance of join evaluation.

References

1. L. Arge. The buffer tree: A new technique for optimal I/O-algorithms. In *Proc. Workshop on Algorithms and Data structures*, 1995.
2. L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, and J. Vitter. Scalable sweeping-based spatial join. In *Proc. Int. Conf. on Very Large Data Bases*, 1998.

3. L. Becker, K. Hinriches, and U. Finke. A new algorithm for computing joins with grid files. In *Proc. Int. Conf. on Data Engineering*, 1993.
4. N. Beckmann, H-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1990.
5. J. L. Bentley. Algorithms for Klee's rectangle problems. Technical report, Carnegie-Mellon University, 1977.
6. T. Brinkhoff, H-P. Kriegel, R. Schneider, and B. Seeger. Multi-step processing of spatial joins. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1994.
7. T. Brinkhoff, H-P. Kriegel, and B. Seeger. Efficient processing of spatial join using R-trees. In *Proc. Int. Conf. on Data Engineering*, 1993.
8. L. Ferarri, P. V. Sankar, and J. Sklansky. Minimal rectangular partitions of digital blobs. In *Proc. 5th Int. Conf. on Pattern Recogonition*, 1980.
9. M. T. Goodrich, J.-J. Tsay, D. E. Vengroff, and J. S. Vitter. External-memory computational geometry. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1993.
10. O. Günther. Efficient computation of spatial joins. In *Proc. Int. Conf. on Data Engineering*, 1993.
11. Y.-W. Huang, N. Jing, and E. A. Rundensteiner. Spatial joins using R-trees: Breadth-first traversal with global optimizations. In *Proc. Int. Conf. on Very Large Data Bases*, 1997.
12. M. Kreveld, editor. *Geographic Information systems*. Utrecht University, 1995.
13. G. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases*. Springer, 1999.
14. R. Laurini and A. D. Thompson, editors. *Fundamentals of Spatial Information Systems*. Acedemic Press, 1992.
15. M-L. Lo and C. V. Ravishankar. Spatial hash-joins. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1996.
16. J. A. Orenstein. Spatial query processing in an object-oriented database system. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1986.
17. J. A. Orenstein. Redundancy in spatial databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1989.
18. J. A. Orenstein and F. A. Manola. PROBE spatial data modeling and query processing in an image datebase application. *IEEE Trans. on Software Engineering*, 14(5):611–629, 1988.
19. J. M. Patel and D. J. DeWitt. Partition based spatial-merge joins. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1996.
20. D. Rotem. Spatial join indices. In *Proc. Int. Conf. on Data Engineering*, 1991.
21. H. Samet. Hierarchical representations of collections of small rectangles. *ACM Computing Surveys*, 20(4):271–309, 1988.
22. K. C. Sevcik and N. Koudas. Filter trees for managing spatial data over a range of size granularities. In *Proc. Int. Conf. on Very Large Data Bases*, 1996.
23. M. I. Shamos and D. Hoey. Geometric intersection problems. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1976.
24. P. Valduriez. Join indices. *ACM Trans. on Database Systems*, 12(2):218–246, 1987.
25. H. Zhu, J. Su, and O. H. Ibarra. An index structure for spatial joins in linear constraint databases. In *Proc. Int. Conf. on Data Engineering*, 1999.
26. H. Zhu, J. Su, and O. H. Ibarra. Toward spatial joins for general polygons. Technical report, Dept. of Computer Science, U C Santa Barbara, 2000.
27. G. Zimbrao and J. M. Souza. A raster approximation for the processing of spatial joins. In *Proc. Int. Conf. on Very Large Data Bases*, 1998.

HMT: Modeling Temporal Aspects in Hypermedia Applications

G□nther Specht [1] and Peter Zoller [2]

[1]Technische Universit□t M□nchen, Institut f□r Informatik
Orleansstra□e 34, D-81667 M□nchen
specht@informatik.tu-muenchen.de
[2]Bavarian Research Center for Knowledge-Based Systems (FORWISS)
Orleansstra□e 34, D-81667 M□nchen
zoller@forwiss.de

Abstract. Since today large hypermedia applications such as huge web sites or extensive CD-presentations cannot be built without a dedicated design phase any more, advanced methodologies for modeling these kinds of applications become more and more important.
In this paper, we present the essence of the Hypermedia Modeling Technique (HMT), which supports the modeling of truly interactive, adaptive and time-based hypermedia applications on arbitrary platforms. HMT offers new design primitives for the specification of temporal dependencies, for modeling search and update interfaces to the underlying data sources, for specifying access restrictions and for defining advanced hyperlinks with a more powerful link concept than their current WWW counterparts. Thus, HMT excels other methods which focus mainly on information presentation issues. The complete HMT design process consists of a sequence of 6 specification steps, two of which will be discussed in-depth within this paper: first, the conceptual hypermedia design of HMT is introduced, because it serves as the basis for all subsequent design steps. Second, temporal design with HMT is described in detail, since audio, video, slide shows, and time-based animations play a more and more important role in today's applications.

1 Introduction

Due to the technical progress in hard- and software development, the availability and acceptance of hypermedia applications has increased significantly within recent years. As a consequence, the interest in hypermedia models, methodologies and CASE tools has grown accordingly. Especially the success of the World Wide Web has lead to increased research efforts in this area, and several data models and methodologies for hypermedia design have been proposed: HDM [6] has been one of the early examples, followed by other approaches like OOHDM [19], RMM [10,11,12], ARANEUS [2,3], or Strudel [5]. Although based on different data models and design processes, these methodologies have one common characteristic: they focus mainly on presentation issues. Although this might be sufficient for certain classes of applications, next generation approaches should also address additional aspects:

First, time is an important aspect for hypermedia applications, which is not supported by most methodologies so far, although the possibility to initialize and synchronize the different elements is a crucial feature when designing hypermedia applications. Second, a more powerful link concept is needed covering advanced features like span-to-span links or link context as described in the Dexter Hypertext Reference Model [9] and the Amsterdam Hypermedia Model [7]. Third, advanced design primitives should be provided, allowing to model both query interfaces and components for the manipulation and maintenance of the underlying data source (which will be some kind of database in most cases). Last but not least, offering the ability to manipulate data on the server inevitably requires a solid authorization concept. By now, aspects of access restriction are usually handled at implementation level instead of being covered by a modeling methodology.

Based on the Relationship Management Methodology (RMM), we have developed an advanced design methodology in order to close these gaps. The *Hypermedia Modeling Technique* (HMT) extends the capabilities of RMM by offering new and enhanced design primitives for querying and manipulating the underlying data source, for specifying access restrictions at different levels, for defining hyperlinks with extended functionality, and for the specification of temporal dependencies. Additionally, we partially changed syntax and graphical representation of the original design primitives in order to provide a more problem adequate, intuitive and easy to use interface.

Since we want to focus on the aspect of temporal hypermedia design, the other HMT design steps will not be discussed exhaustively in this paper. Please refer to [22] for more information on these issues (based on former works described in [23,20]). The remainder of this paper is organized as follows: Section 2 gives an overview of the HMT design process, section 3 presents a detailed description of the HMT conceptual model, and section 4 discusses the temporal design step of HMT. The paper ends with conclusions in section 5.

2 The HMT Design Process

The HMT design process consists of a sequence of 6 steps, where each step corresponds to a different abstraction level of design specification:

Step 1: Requirements Analysis
Requirements analysis covers aspects like the definition of the application domain, identification of intended users, and specification of system functionality and usage.

Step 2: E/R Design
After the application domain has been specified by the requirements analysis, an E/R model [4] has to be built reflecting objects and relationships from the real world application domain. If a hypermedia application has to be built upon an already existing database, the first two steps in the design process are omitted.

Step 3: Conceptual Hypermedia Design
The core hypermedia application design starts with the conceptual hypermedia design of the application, based on the E/R model developed in the previous step. Aspects like information clustering within the documents (which attributes in which document) and navigation (which links in which documents) are dealt with at this design step, which will be discussed in detail in section 3.

Step 4: Authorizational Design
Access restrictions are specified during the authorizational design phase using RBAC-techniques (Role Based Access Control, see for example [14, 17]). Based upon the conceptual hypermedia model built during the previous design step, access to certain parts of the application can be limited by specifying roles required for access.

Step 5: Logical Hypermedia Design
During the logical design phase, additional properties concerning the logical representation of a document's content are defined. This mainly refers to the spatial and temporal order, synchronization, labels or descriptions of the elements.

Step 6: Layout Design
Finally, all aspects regarding the layout of the later presentation are covered by the layout design step. This step heavily depends on the hypermedia system used for building a presentation, although certain aspects are common to most systems: The presentation layout can be described by defining a background color or image, font type and color, standard headers and footers, or default resolution and frame rate.

Throughout the paper, we will use the sample scenario shown in figure 1. It shows a model containing information about research cooperations, their projects, and national research associations acting as umbrella organizations.

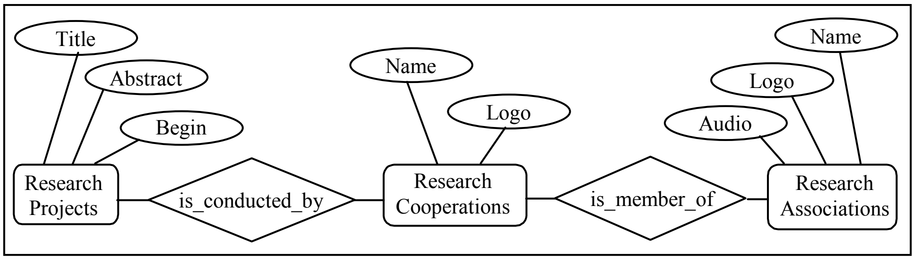


Fig. 1. Sample scenario

3 Conceptual Hypermedia Design

The HMT conceptual design of a hypertext application focuses on two aspects:
Information Clustering is the process of grouping relevant and related portions of information into one place. *Navigational Design* determines the access paths, by the help of which the user can navigate through the application domain by following links to related information. For both tasks, HMT provides a set of primitives described in the following subsections.

3.1 Domain Primitives

In HMT, the unit of presentation is a *document*. A document usually belongs to one E/R-type (called *base E/R-type*) and is identified by a UID (unique identifier, often a meaningful string like "project_overview"). It may contain *attributes* of the base E/R-type, additional information not contained in the E/R model (*adds*), *structural links*, and *access structures* as described later in section 3.2. It may also contain so called *element-groupings*, which are named sets of attributes, adds, structural links, access structures or other element-groupings allowing to build nested structures.

Special cases are documents or element-groupings without a related E/R-type: These can only contain adds, structural links, element-groupings (again without E/R-type), or a restricted set of access structures (which will be described in the following section). They are typically used for the first page (homepage of an application or other documents where no information from the data source is to be displayed.

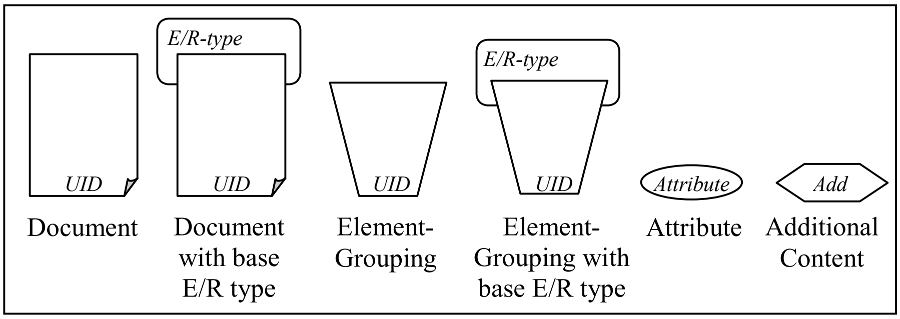


Fig. 2. basic domain primitives

In addition to the basic document primitives described in this paper, HMT also offers a set of *typed documents* like *query documents* or *input documents*, used for building interactive user interfaces for querying or manipulating the underlying data source. These typed documents are labeled with a special symbol in the upper left corner of the corresponding design primitive (for example, query documents are labeled with a question mark). Since temporal modeling with HMT refers mainly to the basic document type, we will not discuss typed documents within this paper.

3.2 Access Primitives

In order to display hyperlinks to or information from other entity types than the base E/R-type, a document or element-grouping can also contain access structures like structural links, TOCs (table of contents), guided tours, or slide shows.

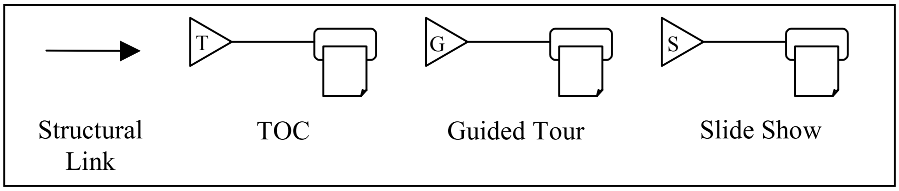


Fig. 3. basic access primitives

A structural *link* originates from an anchor element, which may be an attribute, adds or element-grouping, and points to a target element, which is either a document or an element within a document. This corresponds to span-to-node and span-to-span links as defined in the Dexter Hypertext Reference Model [9]. TOCs, Guided Tours, and Slide Shows reference a (possibly empty) set of entities of another entity type related to the current base E/R-type. These three structures differ in the way they offer access to the information they reference:

TOCs provide a list of descriptions of each referred entity. The contents of this description are defined by the document which is attached to the TOC (see figure 3). A Guided Tour does not display a list of all relevant entities, but shows one single entity of the set together with links pointing to the next element. A Slide Show is like a Guided Tour, but displays the entities automatically one after the other instead of waiting for the user to select the next element. Information about the period of time used to display each entity or the overall time for the Slide Show is specified in the temporal design phase of step 5 (logical design).

The basic link primitive described above causes the current document to be replaced with the target document the link is pointing to. The Amsterdam Hypermedia Model [9] proposes a more powerful approach by introducing the notion of *link context*, describing where the target of a hyperlink is presented and how the source behaves after the link has been selected. HMT deals with this aspect by extending the basic link primitive. Structural links which open a new document are drawn with a small box in the middle of the arrow (see figure 4). If the original document should stop its presentation, the box contains a black square (resembling the stop button of audio or video devices). If the original document should continue its presentation, the box contains a black triangle (resembling the play button of audio and video devices).

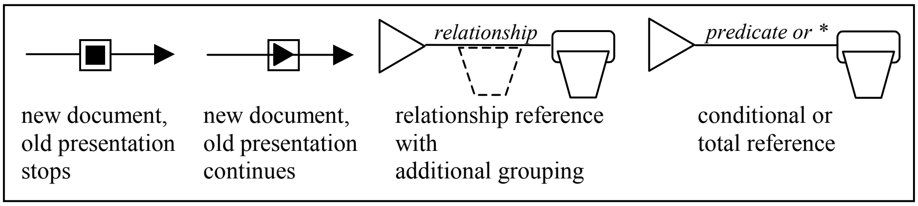

relationship
predicate or *
new document, old presentation stops
new document, old presentation continues
relationship reference with additional grouping
conditional or total reference

Fig. 4. advanced access primitives

Furthermore, TOCs, Guided Tours, and Slide Shows do not always have to be displayed on a new document (*external* TOC/Guided Tour/Slide Show), but can also be included inside the original document (*internal* TOC/Guided Tour/Slide Show). In this case, the document which describes the contents of the access structure has to be replaced by an element-grouping as shown in figure 4. For referencing the set of target entities, access structures can use three different approaches:

- Relationship reference
  By using the name of a relationship, the access structure references all entities of the target entity type which are related to the current entity by this relationship. If the relationship itself contains attributes, an additional *relationship-grouping* with elements of this relationship can be defined as shown in figure 4.
- Total reference
  If the access structure is labeled with "*", all entities of the target entity type are referenced.

- Conditional reference
  The access structure can also be labeled with a predicate specifying a subset of the set of target entities. For example, the predicate "begin > 1998-1-1" for a project TOC references all projects that started 1998 or later. Conditional references can be combined with relationship references.

3.3 Sample Application

In the following example, we model the hypermedia presentation of a research association. The starting document (homepage) includes name and logo, some welcome text and an audio track. At the bottom of the document, a standard footer with links to the homepage and the webmaster is displayed. An external TOC leads to a page with information about associated research cooperations.

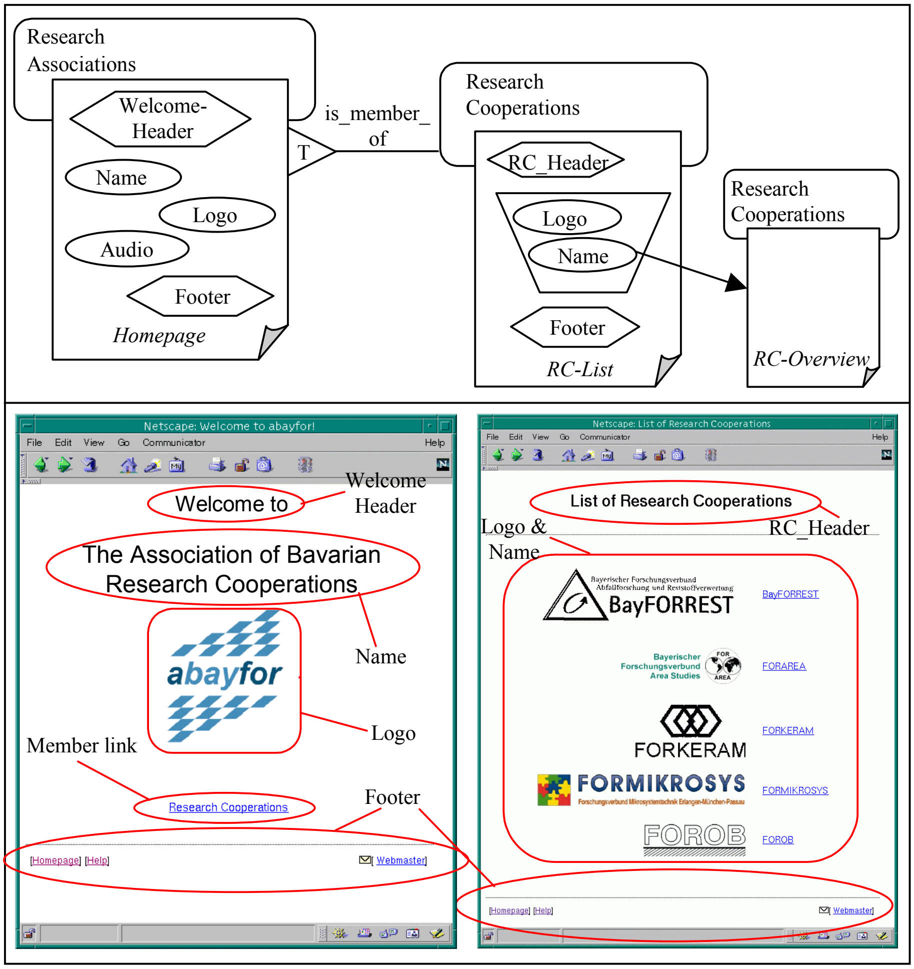


Fig. 5. HMT model and screenshots of the sample application

4 Temporal Design

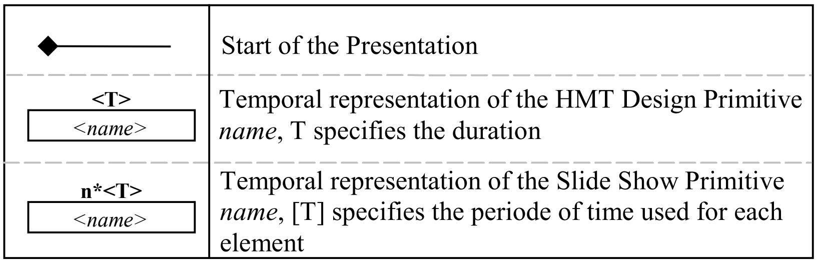
The temporal specification of a hypermedia presentation is an important aspect which has to be supported by hypermedia modeling methodologies. Not only the duration or sequential execution of the presentation's elements, but also various kinds of interdependencies between the different elements have to be specified.

Former approaches were often based on scripting languages, for example Lingo [15]. Although very powerful, these approaches are rather inflexible concerning changes within the presentation, and interdependencies between different elements are often not obvious from the coded specification. Besides, programming skills are required which restricts the number of users for such a methodology. Another classical approach is to use a timeline and to specify absolute positions for each element. This very intuitive technique is rather inflexible regarding changes (all dates "behind" the changed one have to be recalculated), and elements with infinite duration cannot be modeled with this approach, because all dates are absolute points of time. Newer concepts do not rely on exact points of time, but allow to specify temporal relationships between the different elements. Based on the work of Allen [1], several approaches for modeling temporal relations have been proposed [13,8,21,16,18].

The temporal design concept of HMT is inspired by these works and tailored to the special features of its conceptual model. HMT allows to specify the time each design primitive of a document or grouping is active (that means visible). The scope of a temporal specification is always the surrounding HMT grouping or document, which eases the task of synchronizing complex parts of a presentation.

4.1 Temporal Design Primitives

In order to specify aspects of time, the conceptual HMT design primitives are extended by some temporal design primitives. Each document or grouping defines its own local presentation, which is specified within a dashed box attached to the document or grouping. Within this dashed box, the start of a presentation is marked with a special symbol, which is connected to the first element of the presentation.



◆———	Start of the Presentation
<T> *<name>*	Temporal representation of the HMT Design Primitive *name*, T specifies the duration
n*<T> *<name>*	Temporal representation of the Slide Show Primitive *name*, [T] specifies the periode of time used for each element

Fig. 6. HMT temporal design primitives

Each element is represented by a rectangle containing the element's name. The length of the rectangle is primarily determined by the length of its name and is no indicator for the element's absolute duration. However, relations between elements are to some extent symbolized by the relative position of the elements against each other (for example "element a includes element b", see next section). If the absolute duration of an element has to be specified, the corresponding period of time *T* is written above the rectangle. Two cases have to be differentiated:

If timeless elements like text or images are concerned, the effect is trivial. These elements are simply displayed for the specified period of time and then vanish. In contrast to this, time-based components like audio or video elements already have a predefined duration T_P. If they are assigned a certain time of duration T_S, (specified duration) these elements are either cut off if $T_P > T_S$ or restarted if $T_P < T_S$.

A special kind of element are Slide Shows, because for them a temporal specification is mandatory. Three different approaches are possible:

- If an overall duration *T* is specified, the period of time *t* used to present a single element of the Slide Show is calculated as t = T/n, where n is the number of elements of the Slide Show. This ensures that a given presentation doesn't have to be changed if the number of elements of the Slide Show changes.
- Alternatively, it is also possible to specify the period of time *T* used for a single element of the Slide Show by using the expression n**T*. In this case, changes to the number of elements of a Slide Show influence the overall duration and thus have effects on the synchronization with other elements.
- If all elements of the Slide Show are time-based (like video or audio), then no explicit specification of T is necessary, because each element has its own duration.

Since each element of a document or grouping has its own temporal representation, also access structures like structural links or TOCs can be temporarily restricted and synchronized with other elements, for example video clips or audio tracks.

4.2 Temporal Relations

For interval based approaches, Allen [1] has identified 13 characteristic relations, which can be reduced to 7 if the inverse relations are derived by swapping the corresponding elements. HMT adopts and extends these relations:

First, a new relation *synchronizes* is introduced which is especially suited for time-based elements like video or Slide Shows. Second, a simple arrow denotes a delayed sequential execution of two or more elements, whereas an arrow originating from a bullet is used for specifying parallel execution. The element where the arrows originate from are called *synchronizing* elements, the arrows' targets are the *synchronized* elements. Third, the *equals* relation is left out, because it can be replaced by a combination of the *starts* and the *finishes* relation. Figure 7 shows an overview of the temporal relations used in HMT.

For a formal description of the temporal relations in HMT, we use the following syntax: Let *x* be an HMT element, then *T(x)* denotes the overall duration of *x* (either explicitly specified or implicitly defined in case of time-based elements), *start(x)* and *end(x)* define the starting point and the end point of *x*. Please keep in mind that *start(x)* and *end(x)* are abstract measures and are only used to define the relations' semantics, they are not part of the temporal model of HMT.

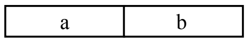
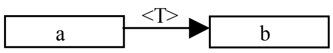
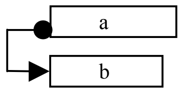
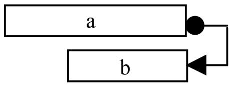
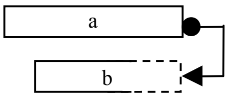
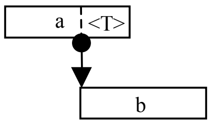
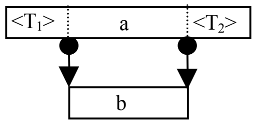
	Immediate sequential execution ("a starts b")
	Deferred sequential execution, T specifies the delay ("a starts b")
	Parallel execution, synchronized beginning ("a starts b")
	Parallel execution, synchronized termination, time-based *bs* are cut off or restarted ("a finishes b")
	Parallel execution, synchronized termination, time-based *bs* are slowed or accelerated ("a synchronizes b")
	Parallel execution, overlapping, T specifies overlapping interval ("a overlaps b")
	Parallel execution, inclusion, T_1 and T_2 specify the delay ("a includes b")

Fig. 7. HMT temporal relations

a meets b: This defines an immediate sequential execution, where *b* is started after *a* has finished. Prerequisite for this relation is that *a* is either a time based element (thus having a predefined duration), or explicitly restricted by specifying *T(a)*.

```
Formal: start(b) := end(a)
```

a before b: Similar to the previous case, *b* is started after *a* has finished. Instead of an immediate execution, the start of *b* is delayed by *T*. Similar to the previous case, element *a* must have a finite duration *T(a)*.

```
Formal: start(b) := end(a) + T
```

a starts b: The elements *a* and *b* are starting simultaneously and are executed in parallel.

```
Formal: start(b) := start(a)
```

a finishes b: This relation synchronizes two elements *a* and *b* in the way that the end of *a* causes *b* to terminate, too. Prerequisite for this case is that *a* has a finite duration (either specified or predefined by time-based elements).

```
Formal: end(b) := end(a)
```

While the synchronization of timeless elements is trivial, time-based elements have to be treated differently. If *b* would normally end before *a* has finished, then *b* has to be restarted in order to bridge the gap until *a* terminates.

```
Formal: T(b)< T(a)-(start(b)-start(a)) ⇒ "restart b"
```

If a should terminate before b, then b has to be stopped.

Formal: $T(b) > T(a)$-(start(b)-start(a)) ⇒ "stop b"

a synchronizes b: For timeless elements, this relation has the same effect as *a finishes b*. But if *b* is a time-based element and has been initiated by some other element *x*, then *b* is slowed or accelerated in order to synchronize with *a*.

Formal: T(b) := *T(a)*-(start(b)-start(a))

a overlaps b: The elements *a* and *b* are executed partially in parallel, where *T* specifies the overlapping interval. Element *a* must have a finite duration *T(a)*.

Formal: start(b) := end(a) - T

a includes b: The elements *a* and *b* are executed in parallel, and *a* starts after and ends before *b*. Element *a* must have a finite duration, element *b* may be infinite unless T_2 is specified.

Formal: start(b) := start(a)+T_1, end(b) := end(a)-T_2

4.3 Sample Application

Modifying our sample scenario from section 3.3, our new homepage shall start with a welcome audio track and an introductory slide show, where name and logo of all connected research cooperations are shown (the order of the elements on the homepage and the order of the Slide Show elements are defined in the logical design step, which is left out due to space limitations). This slide show has to be synchronized with the audio track independent of the number of slides to be presented. Afterwards, name and logo of the research association and the standard footer should be presented together with the external TOC on research cooperations.

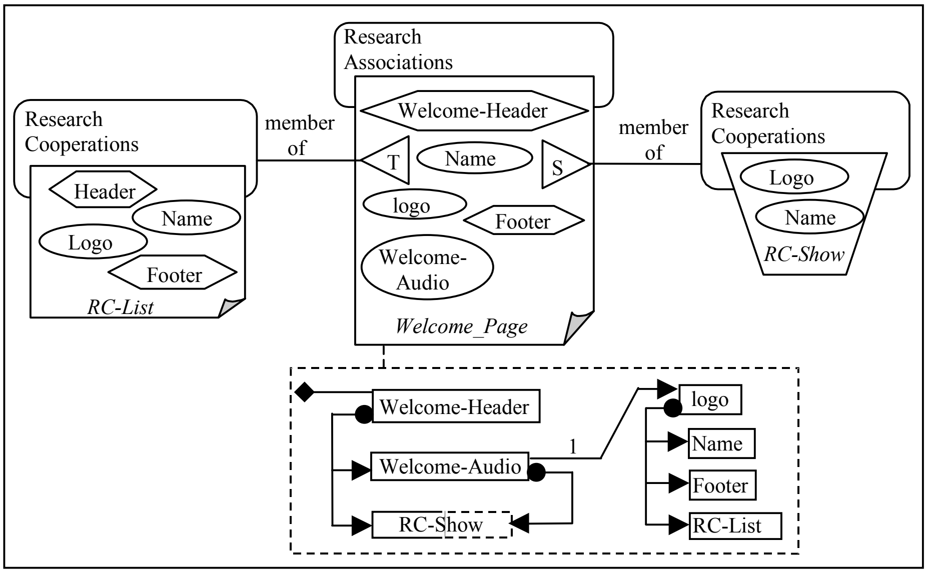


Fig. 8. Temporal specification of the sample scenario

Figure 8 shows the HMT specification, figure 9 shows screenshots of the presentation in different stages.

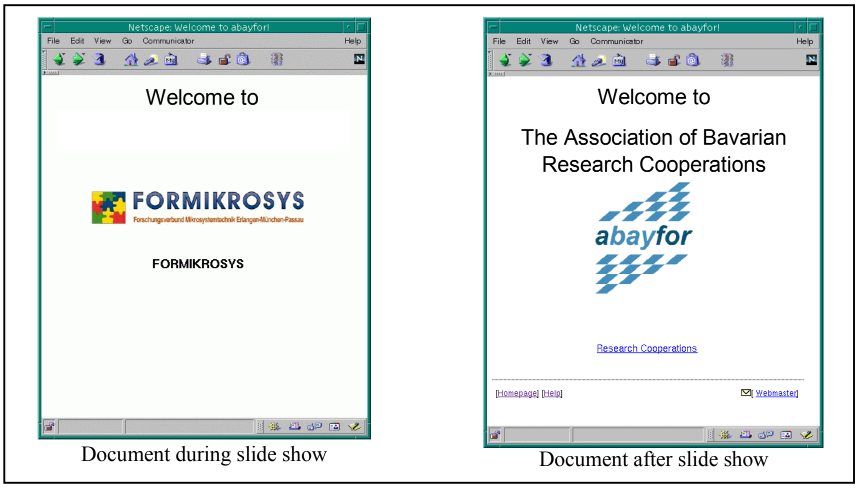

Netscape: Welcome to abayfor!
File Edit View Go Communicator Help
Welcome to
FORMIKROSYS
Forschungsverbund Mikrosystemtechnik Erlangen-München-Passau
FORMIKROSYS
Document during slide show
Netscape: Welcome to abayfor!
File Edit View Go Communicator Help
Welcome to
The Association of Bavarian Research Cooperations
abayfor
Research Cooperations
[Homepage] [Help]
[Webmaster]
Document after slide show

Fig. 9. screenshots of the sample application in different stages

5 Conclusion

In this paper, we have presented the Hypermedia Modeling Technique (HMT), a novel approach to structured hypermedia design. Unlike existing design models and methodologies, HMT covers not only aspects of information presentation, but provides advanced concepts for modeling temporal aspects of hypermedia applications, enhanced hyperlink functionality (link context), interfaces for querying and updating the underlying data source, access restrictions and adaptive documents

In the first part of this paper, we presented an overview of the complete HMT design process, and introduced the HMT conceptual model. Based on an E/R-model of the application domain, the information is clustered within hypermedia documents and groupings. Navigational structures can be defined by using structured links or advanced primitives like TOCs, Guided Tours, or Slide Shows. A sample application has been presented in order to show the usage of the HMT conceptual model.

The second part of our paper discussed in detail temporal hypermedia design in HMT. Each document or grouping is modeled as an independent presentation, and each element (including access structures) gets its individual temporal specification. For synchronization purposes, seven different dependencies between the elements of a hypermedia document have been identified and added to the HMT design primitives. A modified version of the sample scenario has been used to illustrate the usage of the temporal design primitives of HMT.

References

[1] Allen, J.F.: "Maintaining Knowledge about temporal intervals". *Communications of the ACM* 1983/26, Vol. 11, pp. 832-843

[2] Atzeni, P., Mecca, G., Merialdo, P.: "To Weave the Web". *Proc. 23rd International Conference on Very Large Databases (VLDB'97)*, Athens,1997, pp. 206-215

[3] Atzeni, P., Mecca, G., Merialdo, P.: "Design and Maintenance of Data-Intensive Web Sites". *EDBT98*, Valencia, 1998, pp. 436-450

[4] Chen, P.S.: �The Entity-relationship Model � Towards a Unified View of Data� *ACM TODS,* Vol. 1, No. 1, 1976.

[5] Fernandez, M., Fiorescu, D., Kang, J., Levy, A., Suciu, D.: "STRUDEL: A Web-site Management System". *ACM SIGMOD 1997*

[6] Garzotto, F., Paolini, P., Schwabe, D.: �HDM- A Model-based Approach to Hyper-media Design� *ACM Transactions on Information Systems*, 11, 1 (1993), pp. 1-26.

[7] Hardman, L., Bulterman, D.C.A., Rossum, G.: "The Amsterdam Hypermedia Model". *Communications of the ACM*, 1994/2, Vol.37, pp. 50-62

[8] Hoschka, P.: "Synchronized Multimedia Integration Language". http://www.w3.org/TR/WD-smil-971109

[9] Halasz, F., Schwartz, M.: "The Dexter Hypertext Reference Model". *Communications of the ACM*, Feb. 1994, Vol. 37, No. 2, pp. 30-39

[10] Isakowitz, T., Kamis, A., Koufaris, M.: �Extending the capabilities of RMM: Russian Dolls and Hypertext� *Proc.30th HICSS*, 1997.

[11] Isakowitz, T., Kamis, A., Koufaris, M.: �Reconciling Top-Down and Bottom-Up Design Approaches in RMM� *Proc. Workshop on Information Technologies and Systems (WITS97)*, Atlanta, GA, Dec. 1997.

[12] Isakowitz, T., Stohr, E., Balasubramanian, P.: �RMM: A Methodology for the Design of Structured Hypermedia Applications� *Comm. of the ACM*, 38(8), pp. 34-44.

[13] Jourdan, M., et al.: "Authoring Environment for Interactive Multimedia Documents". INRIA Rhone-Alpes, Montbonnot, France, http://opera.inrialpes.fr/OPERA

[14] Lupu, E., Sloman, M.: "Reconciling Role Based Management and Role Based Access Control". *Proc. 2nd ACM RBAC Workshop*, Fairfax, VA, USA, Nov. 1997

[15] Macromedia: Director 4.0 *User's Guide*.

[16] Rossum, G., Jansen, J., Mullender, K.S., Bulterman, D.: "A Presentation for Portable Hypermedia Documents". *Proceedings of the ACM Multimedia '93*, pp.183-188

[17] Sandhu, R., Coyne, E.J., Feinstein, H.L., Youman, C.E.: "Role-Based Access Control Models". *IEEE Computer*, 29(2), Feb. 1996, pp. 38-47

[18] Song, J., Kim, M.Y., Ramalingam, G.: "Interactive Authoring of Multimedia Documents". *Research Report RC 20369*, T.J. Watson Research Center, IBM Research Division Yorktown Heights, NY

[19] Schwabe, D., Rossi, G., Barbosa, S.: �Systematic Hypermedia Design with OOHDM� *Proc. Hypertext 96*, Washington, Mar. 1996.

[20] Sommer, U., Zoller, P.: "WebCon: Design and Modeling of Database Driven Hypertext Applications" *Proc. 32nd HICSS*, 1999.

[21] Yu, J.: "A simple, intuitive hypermedia synchronization model and its realization in browser/java environment". Technical Note 1997-027, Digital Equipment Corporation Systems Research Center, Palo Alto, CA, October 1997

[22] Zoller, P.: "HMT: Modeling interactive, adaptive Hypermedia Applications". To appear in : *Siau, K., Rossi, M. (editors): Modelling Methodologies for the next Millenium*, Idea Group Publishing, spring 2000.

[23] Zoller, P., Sommer, U.: "WebCon: A Toolkit for an Automatic, Data Dictionary Based Connection of Databases to the WWW" *Proc. 1998 ACM Symposium on Applied Computing*, Atlanta, 1998, pp. 706-711

Hana Tree: A Dynamic and Robust Access Method for Spatial Data Handling

Yongwon Kwon and Changsung Jeong*

Department of Electronics Engineering, Korea University
1-5Ka, Anam-dong, Sungbuk-ku, 136-701, Korea
luco@snoopy.korea.ac.kr
csjeong@charlie.korea.ac.kr

Abstract. In this paper, we present a new multidimensional access method, called *hana tree*, which can provide an efficient and robust way for dynamic modifications and interactive queries to spatial database. We designed a new multidimensional binary encoding scheme, *hana code*, for representing points and page regions in hana tree. When a new spatial data is inserted in uncovered data space, the page region of the existing node is expanded to include it instead of creating a new one. These properties allow the decrease in the total number of nodes and the reduction in the number of nodes accessed during search, thus improving the overall performance in storage space utilization and various operations Also, we shall describe our experimental results where hana tree is compared to other point access methods to show the superiority of hana tree.

1 Introduction

Multidimensional access method is an essential part of spatial database system. Therefore, the performance improvement of spatial database system can be accomplished by developing more efficient access method that handles large amount of multidimensional data. The access methods must be dynamic to support arbitrary insertions and deletions of objects without any global modifications and any loss of performance. Also, they should efficiently support a set of queries, such as exact match, partial match and range queries. The access methods can be classified into point access method(PAM) and spatial access method(SAM) [5,11]. The former is designed to handle multidimensional point data, while the latter to handle spatial data such as polygons and rectangles. In this paper we are concerned with the design of new multidimensional PAM.

In multidimensional PAM, a data space is composed of points each of which is stored in a record with its unique multidimensional key, and partitions into page regions such that all the records in a data page are taken from one page region. The multidimensional PAM has been studied extensively in the literature [1,2,3,4,5,6,7,8,9,10,11], and can be classified into several classes according to the three properties of the regions: whether they are rectangular or not, disjoint or not, and complete or not, i.e. partition the complete space or not [5,11].

* This work has been supported by KOSEF and Brain Korea 21 project

Grid file [1], multidimensional extendible hashing [2], k-d-B tree [3] and LSD tree [8] belong to the class where the page regions are disjoint, rectangular and complete. With the disjoint property, they do not allow overlap in the directory nodes and can guarantee that exact match queries are restricted to one path of the directory. However, their performance degenerates for highly correlated data with the complete property. BANG file [4] and hB-tree [6] belong to the class which is similar to the previous one, but not rectangular. They have been proposed to allow more general shapes of regions which are constructed by difference and union of rectangles. Buddy tree [5] and MLGF(multilevel grid file) [7] belong to the class where the page regions are disjoint, rectangular, but not complete. With the incomplete property, they can avoid partitioning the empty data space, and therefore queries, such as partial match queries, where the query region intersects with empty data space, can be performed much faster than the other PAMs partitioning the complete data space. In [11], PAMs is also classified into hashing, tree structured methods, space-filling curves, and hybrid approaches. Grid file and multidimensional extendible hashing belong to the PAM based on hashing, and k-d-B tree, LSD tree, hB tree, BV tree [9] and GIST [10] belong to the tree structured methods. Buddy tree, BANG file, and MLGF are hybrid approaches, that is, tree structured methods with a dynamic hashing scheme. Tree structured methods can be further divided by their internal structure: whether they are height-balanced or not. MLGF and hB-tree are height-balanced, while Buddy tree is not. Space filling curves such as z-ordering [12] and the Hilbert curve [13] preserve spatial proximity to some extent by providing total orders.

In this paper, we present a new multidimensional PAM, called hana tree. It is a height balanced tree which belongs to hybrid approach. We designed a new multidimensional binary encoding scheme, called *hana code* for representing points and page regions in hana tree. We shall show that our access method can provide an efficient and robust way for dynamic modifications and interactive queries to spatial database.

The outline of our paper is as follows: In section 2 and 3, we present a formal description of hana code and a structure of hana tree respectively. In section 4, we describe algorithms. In section 5, we explain an experimental performance comparison. In section 6, we give a conclusion.

2 Hana Code

Hana code is a binary encoding scheme used for representing points and intervals in one dimension. It has a hierarchical structure where hana code in the top level covers the whole interval, and each hana code in one level covers the half interval of its upper level. The hana code in top level 0 is 00..01, and each hana code in level i is obtained from hana code in its upper level $i-1$ by left shifting it once and then attaching 0 or 1 in the lowest significant bit if it covers the left or right half respectively. Hana codes in the lowest level represent points, and others for representing intervals. Figure 1 shows an example of hana code which consists of 4 bits. Points from 000(0) to 111(7) are represented by hana codes from 1000

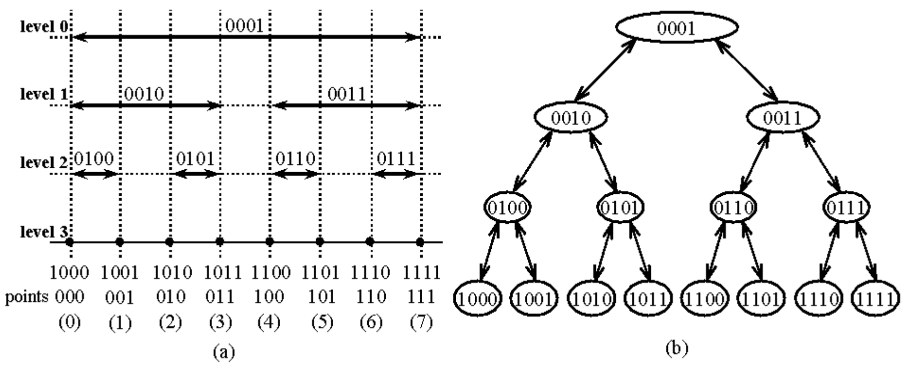


Fig. 1. 4-bit hana code

to 1111 respectively in the lowest level 3 by attaching 1 in the most significant bit. The whole interval which comprises all the numbers is represented by hana code 0001 on top level 0. The left and right half intervals of 0001 are represented by hana codes 0010 and 0011 respectively in level 1, and the left and right half intervals of 0010 represented by 0100 and 0101, and so on. Note that two hana codes lie in the same level if the first '1' appearing to the right of their most significant bit lies in the same bit position. Generally, for hana code with N bits, 2^{N-1} points and $2^{N-1} - 1$ intervals can be represented.

Hana code x is called *left* or *right* code of hana code y if x covers the left or right half interval of y respectively, and y is called a *parent* of x. Note that for hana code y, its left and right hana codes are obtained by $2 * y$ and $2 * y + 1$ respectively, and its parent $\lfloor \frac{y}{2} \rfloor$. Therefore, if we consider hana codes as the indices of heap tree, hana codes for intervals and points can be mapped into internal and leaf nodes respectively in the heap tree. (See Fig. 1b.) We shall often refer to the terminology used in the heap tree interchangeably for the description of hana codes.

We define *h-interval* as the interval which can be represented by hana code. Let E be a set which consists of points and h-intervals each represented by hana code. Then, we define the *mh-interval* of E as the h-interval with the minimum length which spans all the elements in E, and the *mh-interval* is said to *cover* the elements in E. Since each element in E corresponds to a node of heap tree, the mh-interval of E can be obtained by finding the lowest common ancestor(LCA) of all the elements of E in the heap tree. LCA can be computed by repeatedly right shifting the elements in E until all of them become equal, since for each element, its parent can be obtained by right shifting. Therefore, we have the following property.

Property 2.1: Given a point p and h-interval I represented by hana codes respectively, the mh-interval which covers p and I can be obtained by bitwise operations.

Since the inclusion relation between points and h-intervals can be determined by checking their ancestor and descendant relationships, the following property holds.

Property 2.2: Given points and h-intervals represented by hana codes in one dimension, we can determine the inclusion relation between points and h-intervals, or between h-intervals by bitwise operations.

For instance, a point 1001 is included in the h-interval 0010, because 0010 is obtained by right shifting 1001 twice. An h-interval 0101 is not included in h-interval 0011, since its parent 0010 is a sibling of 0011.

Generally, hana code representation for points and intervals in one dimension can be extended to points and page regions in multidimension. Let $(d_1, d_2, d_3, \dots, d_m)$ represent an m dimensional point q, where each d_i is a value in the ith dimension. Then, each point q can be represented by its *hana-code point vector*, *hp_vector(q)*$= (h_1, h_2, h_3, ..., h_m)$, where h_i is a hana code for d_i. That is, each dimension of *hp_vector*(q) is obtained by transforming each dimension of q into its corresponding hana code.

A page region $RG(P)$ for a set P of m dimensional points is represented by *hana-code region vector* *hr_vector*$(P) = (hr_1, hr_2, .., hr_m)$, where each dimension hr_i is a hana code for the mh-interval which covers all the ith dimensional values of the points in P, and $RG(P)$ is said to *cover* the points in P. Therefore, the page region has the shape of the minimal rectangle of P which is obtained by successively halving the h-interval in each dimension. Similarly as in one dimensional case, we have the following property.

Property 2.3: Given a point q and $RG(P)$ for a set P of points in m dimensional space, each represented by hana-code point and region vectors respectively, $RG(E)$ for $E = P \cup \{q\}$ can be obtained by bitwise operations.

Property 2.3 follows from property 2.1, since each ith dimension of $RG(E)$ is obtained by finding the mh-interval which covers hr_i of $RG(P)$ and h_i of q. In property 2.3, $RG(E)$ is called the *expanded page region* of $RG(P)$ to include q. One page region R is said to be *nested* in another page region T, denoted $R \subset T$, if R is completely surrounded by T.

Since the inclusion relation between ith dimensional values of the page regions can be checked by bitwise operations from property 2.2, the following property holds.

Property 2.4: Given points and page regions in m dimension which are represented by hana-code point and region vectors respectively, we can determine the inclusion relation between points and page regions, or between page regions by bitwise operations.

Input data can be broadly classified into two groups: point data and non-point data such as line and polygon. We can represent point data by hana-code point vector by transforming each dimension to hana code. For non-point data, we first approximate arbitrary geometric object to the enclosing rectangle, and represent it by hana-code point vector which stores hana codes for two bounding values along each dimension of the rectangle. Therefore, the dimension of the hana-code point vector for the enclosing rectangle is two times that of the space

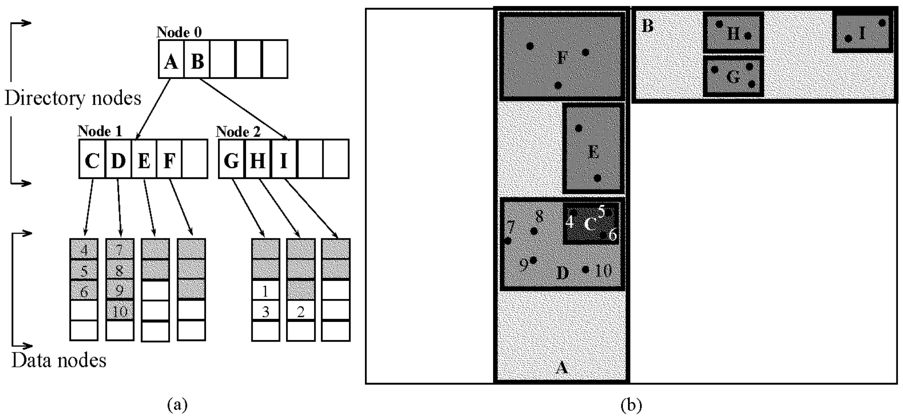


Fig. 2. An example of hana tree

where the rectangle lies. With hana code, we can increase the fan-out of directory, since hana codes for one dimensional point and interval require the same number of bytes as well as hana codes for multidimensional point and page region.

3 Hana Tree

3.1 Structure of Hana Tree

Hana tree is a height-balanced tree where each internal and leaf nodes represent directory and data nodes respectively. Every node consists of several entries $(E_1, E_2, .., E_k)$ each of which consists of two fields. For directory node, each entry E_i consists of two fields (PR_i, pt_i), where PR_i is a page region of its ith child represented by hana-code region vector and pt_i is a pointer to the ith child node. For data node, each entry E_i consists of two fields (P_i, pt_i), where P_i is a input point represented by hana-code point vector and pt_i is a pointer to the input data with P_i as a key.

Each directory node is associated with a page region, and stores information about the page regions of its children. The page region associated with each node covers all the page regions of its children. We assume that a page region in the directory node refers to one of PR_i's of its children. The page regions in the same directory are called *sibling* page regions. Figure 2(a) shows an example of hana tree, and Fig. 2(b) illustrates its page regions and input points respectively. Page region A in the root node covers C, D, E, F in its left child, and B covers G, H, I in its right child.

3.2 Properties of Hana Tree

In this section, we shall explain various properties of hana tree. Specifically, for a directory node N, let CV_i be a set of input points in all of the descendant

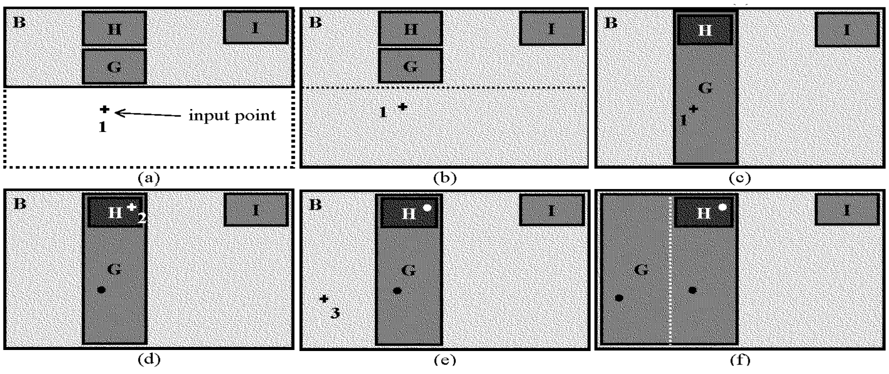


Fig. 3. The examples of the page region expansion

data nodes of its ith child. Then, PR_i in N is a page region $RG(CV_i)$ which covers the points in CV_i, and CV_i is called its *covering point set.* The page region $RG(CV_i)$ is the minimum enclosing rectangle of CV_i each dimension of which is a h-interval. Therefore, the following property holds.

Property 3.1: The page region in hana tree is rectangular, and the union of all the page regions does not span the complete data space.

Thus, the union of page regions avoids partitioning empty data space in hana tree, and queries, such as partial match queries, where the query region intersects with empty data space, can be performed much faster than the conventional structures partitioning the complete data space.

When we insert a point q at a directory node, we consider two cases according to whether it belongs to any page region in the directory or not. If it belongs to only one page region, we simply insert it to that page region. If it belongs to more than one page regions in the directory, we select the smaller page region for insertion. If it does not belong to any page region in the directory, one of the page regions in the directory, say PR_k, is selected, and expanded to include it instead of creating a new one. However, we do not select the page region PR_k to be expanded for the following three cases: 1) PR_k is nested in other page region, 2) the expanded page region of PR_k intersects other page regions partially, and 3) the expanded page region of PR_k nests another page region which in turn nests other page regions. Therefore, the following property holds in hana tree.

Property 3.2: For any two sibling page regions in hana tree, either one is disjoint with another, or one of them is nested in another, and each page region is not nested in more than one sibling page region.

By expanding the existing page region instead of creating a new one as in other access methods like buddy tree and MLGF, we can decrease the number of page regions in the directory node, and hence decrease the total number of nodes.

Figure 3 illustrates an example of inserting three points in hana tree shown in Fig. 2. When a point 1 is inserted in Fig. 3a, it does not belong to any page region, and the page region B at root directory node 0 is expanded to include it in Fig. 3b, and then the page region G is expanded to include it at directory node 2, since it does not belong to any page region at directory node 2. At this point, H is nested by the expanded region of G. When a point 2 is inserted in Fig. 3d, it belongs to two page regions G and H, and the smaller page region H is selected to insert the point. Finally, when a point 3 is inserted in Fig. 3e, it is inserted to the page region B where it belongs at root directory node, and the page region G at directory node 2 is expanded to include it. If the page region I instead of G is expanded to include the point 3, I nest G which in turn nest H, i.e. H is nested in more than one sibling page regions, G and I, violating the property 3.2.

For a page region R in directory node N, let S_R be a set of sibling page regions which are nested by R, and let R_m be the union of all the page regions in S_R. Then, $R - R_m$ is called a logical page region of R, denoted $LG(R)$. That is, the logical page region of the page region R in a directory node is obtained by excluding from its page region all the sibling page regions which are nested in R. Therefore, for two sibling page regions R and T with $R \subset T$, their logical page regions are disjoint. Since we insert a point in the smaller page region if there exist two sibling page regions which cover it, all the points in the covering point set of the page region T lie in its logical page region $LG(T)$, and hence the following property holds.

Property 3.3: For two sibling page regions R and T with $R \subset T$, their covering point sets as well as their logical page regions are disjoint, and hence for exact match query, insertion and deletion, each of their access paths is unique.

Therefore, we can prevent the loss of performance by restricting the insertions, deletions and exact match query to exactly one path of the directory. For the example of Fig. 2(b), the page region D overlaps with C, but their logical page regions are disjoint, since the logical page region of D is $D - C$, and that of C is identical to C. Points 4, 5, 6 are stored in C, since they belong to both of the page region C and D at directory node 1, but C is smaller than D. They constitute a covering point set of C, and lie in the logical region of C. Points 7, 8, 9, 10 are stored in D at directory node 1, since they belong to D but not C. Similarly, they constitute a covering set of D, and lie in the logical region of D. Therefore, the covering point sets of C and D as well as their logical page regions are disjoint, and the access path becomes unique.

The reason that we do not expand the page region which is nested in other page region is that we may lose some data. For instance, in Fig. 4e, if we expand, instead of G, the page region H which is nested by G, we can not find the points in H before expansion, since they now lie in the logical page region of G.

4 Algorithm

4.1 Search

In this section we shall explain about exact match and range queries. Basically, the search algorithm traverses the tree from root to leaf to find the data which satisfies a given query. In exact match query, we select, at each directory node, a page region which covers a given key by comparing hana-code point vector of the key with hana-code region vector of each page region in the directory. We can determine whether or not a page region covers the key from property 2.4. If we find two page regions which cover the key, we select the smaller one. Note that there are at most two page regions which cover the key by property 3.2.

In range query, there are several types like orthogonal range query, circular range query and polygonal range query [8]. In orthogonal range query, we traverse all the directory nodes whose page regions are overlapped with the query region. Other queries are similar to orthogonal range query except the way to evaluate intersection between the query region and page region at each directory node.

4.2 Insertion

The insertion of a single input data consists of two steps. First, we traverse the tree from the root downward and insert it into the proper data node. During the traversal, we find, at each directory node, the page region whose logical page region covers the given input data if it exists as in search algorithm; otherwise expand one of the page regions in the directory node as we described in the previous section. Second, we execute the split step if the data node is full by traversing from the data node toward the root. The detailed algorithm for inserting an input data with a point q as its key is given below.

1. IF root is null THEN
   Create a data node D, and insert an entry (P_1, pt_1) to D, where P_1 stores q represented by hana-code point vector and pt_1 is a pointer to the block containing input data. Then, create a root directory node with an entry (PR_2, pt_2), where PR_2 is a page region of the data node, represented by hana-code region vector, and pt_2 is a pointer to D.
2. ELSE
   Find the deepest node whose page region covers q by using Search Algorithm. If it is a data node, go to step 4; otherwise go to step 3.
3. Expand the page region as follows:
   (a) Select a page region of the child to be expanded to include q as described in the previous section.
   (b) Expand the page region so that it can cover q by updating its hana-code region vector. Repeat this step traversing downward until the selected node is a data node.
4. Insert q to the selected data node S.
5. If S is full, split it into two nodes.

(a) Choose a dimension i with the largest range among the dimensions of R_s, where R_s is the page region which covers the points in S.
(b) Make two hana-code region vectors by replacing hana code in dimension i with its left and right hana codes respectively. One hana-code region vector, denoted by hr_l, covers the lower half of R_s along dimension i. Similarly, the other hana-code region vector, denoted by hr_r, covers the higher half of R_s along dimension i.
(c) Partition the selected node S into two nodes S and N with the page regions represented by hr_l and hr_r respectively.
(d) Replace the entry in the parent node of S with (hr_l, pt_l), and insert a new entry (hr_r, pt_r), where pt_l and pt_r are pointers to S and N respectively.
(e) Repeat step 5 if the parent node of node S is full.

4.3 Deletion

In this section, we briefly explain about the deletion. We apply the search algorithm to find the data node D which stores a given data to be deleted, and then delete it from D. If the number of entries in D is small, we can merge it to the other sibling node. Two methods can be used for merging the nodes. The first is to insert each entry of D into its proper sibling node one at a time. The second one is to find a node that can be expanded to include all the entries of D, and then merge it to the node at once. In our algorithm, the former method is used, since it may generate the smaller page regions than the latter method, even though it takes slightly more time.

5 Performance Evaluation

In this section we shall compare the performance of hana tree with other PAMs: buddy tree and MLGF. In [5], buddy tree is compared with other PAMs such as hB-tree, BANG-file, and grid file, and it is shown that it outperforms other PAMs for the data with the following characteristics: a) densely populated and unpopulated areas vary over the data space, b) sorted data is inserted, and the previous various experimental comparisons show that buddy tree belongs to the best performing access methods[11]. Buddy tree is designed to organize highly correlated data very efficiently, while grid files loose performance for such data. However, grid file is superior to buddy tree for the data with uniform distribution [5]. MLGF is an extension of the grid file, and solves many drawbacks of the grid file caused by its multidimensional array directory. So, we choose these two PAMs for the performance comparison to show the superiority of our hana tree.

We implement all the PAM's with C++ in Pentium II PC. We choose the size of disk page to be 512 bytes and use 32 bits integer type for each dimension of data. For our experiments, we used three different data sets of 2-dimensional points in Fig. 4 and two real data sets in Fig. 5 respectively. Figure 4a is a uniformly generated data. Figure 4b consists of 25 clustered groups, and Fig. 4c

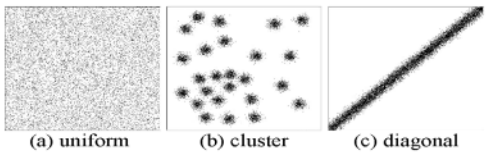


Fig. 4. Point data sets

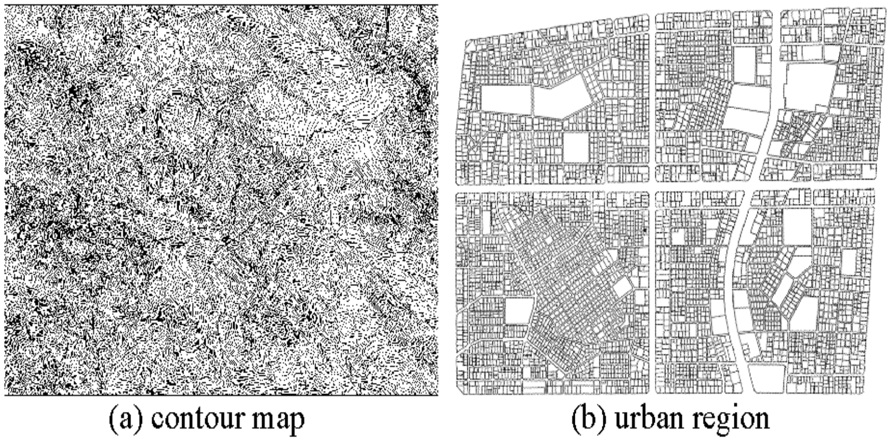


Fig. 5. Real data sets

is a randomly generated data along the diagonal line. Each point data set consists of 25000 points. Figure 5(a) is a contour map for the geographic region, which consists of 6500 polylines. Figure 5(b) shows a number of small sections in the urban area which consists of 15000 lines.

The costs of insertion and range queries are evaluated in terms of the number of page accesses. The number of page accesses is normalized with respect to that of hana tree with its value fixed to 100. In range query, the region of each query is a square which vary in size from 0.1 % to 10% relatively to the data space. The storage utilization is calculated by averaging storage utilization of every nodes. In table 1, we computed the cost of insertion and range queries, the total number of nodes, and storage utilization for three different data sets shown in Fig. 4 respectively. Similarly, we computed the contents of table 2 for data sets in Fig. 5 respectively.

Even though the insertion algorithm of MLGF is simpler than hana tree, the insertion cost of hana tree is slightly better than MLGF for all data sets in table 1 and 2a. For the urban region of table 2b the insertion cost of hana tree is almost similar to that of MLGF. The insertion cost of hana tree outperforms the buddy tree for all the different data sets in table 1 and 2. This is due to that fact that buddy tree needs more disc page I/O to repeatedly test if two page

Table 1. Results of data in Fig. 4.

	a. uniform data			b. diagonal data			c. clustered data		
	Hana	Buddy	MLGF	Hana	Buddy	MLGF	Hana	Buddy	MLGF
Insert	100	353.3	100.9	100	306.4	101.2	100	333.8	101.1
Node	911	1049	907	1087	1094	1087	1117	1177	1136
Storage Util.	72.2	64.2	72.9	60.9	62.1	60.1	59.4	58.8	58.5
Range Query	100	107.3	100.9	100	100.1	100.7	100	104.4	105.6

Table 2. Results of real data in Fig. 5.

	a. contour map			b. urban region		
	Hana	Buddy	MLGF	Hana	Buddy	MLGF
Insert	100	429.1	103.7	100	1004.1	99.3
Node	608	666	610	1819	1834	1762
Storage Util.	58.9	54.2	58.9	46.9	50.6	48.3
Range Query	100	102.2	496.6	100	94.3	152.8

regions can be merged in order to find buddies, while hana tree performs insert operations by expanding the existing page region without the need to merge two page regions.

The number of nodes in hana tree is smaller than that of buddy tree for all data sets. This results from that the fan-out in the directory node is increased by the hana code representation which requires the same amount of memory for both of point and page region and the existing page region is expanded instead of creating a new one. Also, it is almost identical to that of MLGF for uniform, diagonal, contour map, but smaller for clustered data set, while larger for urban data set.

In terms of storage utilization, hana tree is almost identical to MLGF, and better than buddy tree for uniform, clustered data, and contour map. But hana tree is slightly inferior to buddy tree for diagonal and urban region data.

The range query cost of hana tree is better than MLGF for all data sets, and superior to MLGF especially for urban region data and contour map. Also, it is better than buddy tree for all the data sets, with the exception for the urban region data. It shows that hana tree partitions the data space more efficiently than MLGF and buddy tree in the overall cases.

Our experimental result shows that hana tree is a robust multidimensional access method whose performance does not vary to a large extent in relation to the data distribution and data dimension. Especially, for highly correlated clustered data, hana tree is better than buddy tree and MLGF in all aspects.

6 Conclusion

In this paper, we have presented a new multidimensional access method, called hana tree. In order to represent point data as well as the page region in hana tree, we have designed a new multidimensional binary encoding scheme, hana code. By using hana code, the page regions in hana tree can be easily represented as the minimal enclosing rectangle of their covering sets. Therefore, the union of all the page regions does not cover the complete data space, and avoids partitioning empty data space, speeding up queries, such as partial match queries, where the query region intersects with empty data space. Also, hana code representation can increase the fan-out in the directory node by storing point and page region using the same amount of bytes. Moreover, the expansion of the existing page region instead of creating a new one during insertion allows the decrease in the total number of nodes in hana tree and hence the reduction in the number of page accesses during insertion, deletion and query operations, improving the overall performance. The page regions may not be disjoint. However, with the disjoint property of covering sets between sibling page regions as well as their logical page regions in hana tree, we can restrict insertion, deletion and exact query operations to exactly one path of hana tree. Our experimental result shows that hana tree outperforms buddy tree and MLGF in many aspects for different data sets, and it is a robust multidimensional access method whose performance does not vary too much in relation to the data distribution. As a future work, we are going to investigate other spatial operations such as join, neighbor search, etc.

References

1. J.Nievergelt, H.Hinterberger, K. C.Sevcik : The Grid File : An Adaptable, Symmetric Multikey File Structure. ACM Trans. Dtabase System, Vol. 9, No. 1 (1984) 38–71 271, 272
2. E. J. Otoo : A mapping Function for the directory of a multidimensional extendible hashing. In Proc. of the tenth Int. Conf. on Very Large Data Bases (1984) 491–506 271, 272
3. J. T. Robinson : The k-d-B-tree : a search structure for large multidimensional dynamic indexes. In Proc. of ACM SIGMOD Int. Conf. on Management of Data (1981) 10–18 271, 272
4. M. Freeston : The BANG file : a new kind of grid file. In Proc. of the ACM SIGMOD Int. Conf. on Management of Data (1987) 260–269 271, 272
5. B. Seeger and H. P. Kriegel : The Buddy-Tree : An Efficient and Robust Access Method for Spatial Data Base Systems. In Proc. of the 16th Int. Conf. on Very Large Data Bases (1990) 590–601 271, 272, 279
6. D. B. Lomet and Betty Salzberg, : The hB-Tree : A Multiattribute Indexing Method with Good Guaranteed Performance. In Proc. of the fifth IEEE Int. Conf. on Data Engineering (1989) 296–304 271, 272
7. K. Y. Whang and R. Krishmamurthy : The Multilevel Grid File - A Dynamic Hierarchical Multidimensional File Structure. Int. Symposium on Database Systems for Advanced Applications (1991) 449–459 271, 272

8. A. Henrich, H. W. Six, and P. Widmayer : The LSD tree: spatial access to multidimensional point and nonpoint objects. In Proc. of the fifteenth Int. Conf. on Very Large Data Bases (1989) 45–53 271, 272, 278
9. M. Freeston : A general solution of the n-dimensional B-tree problem. In Proc. of the ACM SIGMOD Int. Conf. on Management of Data (1995) 80–91 271, 272
10. J. M. Hellerstein, J. F. Navghton and A. Pfeffer : Generalized search trees for database systems. In Proc. of the 21st Int. Conf. on VeryLarge Data Bases (1995) 562–573 271, 272
11. V. Gaede and O. Gunther : Multidimension Access Methods. ACM Computing Surveys, Vol. 30, No. 2 (1998) 170–230 271, 272, 279
12. J. Orenstein and T. H. Merrett : A class of data structures for associative searching. In Proc. of the third ACM SIGACT-SIGMOD symposium on Principles of Database Systems (1984) 181–190 272
13. C. Faloutsos and S. Roseman : Fractals for secondary key retrieval. In Proc. of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databases Systems (1989) 247–252 272

MADGIS: A New Architecture for Distributed GIS under Internet Environment

Jihong Guan[1], Shuigeng Zhou[2], Fuling Bian[1] and Yunfa Hu[2]

[1]School of Informatics Engineering
Wuhan Technical University of Surveying and Mapping
430079, Wuhan, China
{jhguan, flbian}@wtusm.edu.cn
[2]Computer Science Department
Fudan University, 200433, Shanghai, China
{sgzhou, yfhu}@fudan.edu.cn

Abstract. This paper proposes the architectural concept of Mobile Agent-based Distributed Geographic Information System (MADGIS) for the first time to meet the requirements of GIS applications under Internet environment with limited communication bandwidth and unstable connectivity. First, the architecture of MADGIS is described, while detailed discussions focus on the structure of mobile agent facilitator and the operation process of MADGIS. Then the key techniques for MADGIS implementation are explored. And finally, an application prototype based on the MADGIS architecture is reported.

1 Introduction

Geographic Information System (GIS) is a particular information system that enables the users to query, process and analyze spatial (geometric, topological) data as well as non-spatial data. With the development of network technology and popularity of World Wide Web (WWW), GIS has been changing from the original isolated and centralized information management model to an open and distributed architecture linking through the WWW framework. Such a situation calls for new architectures and computing technologies for distributed GIS under Internet environment. Mobile agent is a recently developed computing paradigm that combines agent technology and distributed computing technology and is well suited for open and dynamically changing environment [1]. Therefore, in this paper we propose a new architecture for distributed GIS based on mobile agent, which we call MADGIS, *i.e.* Mobile Agent based Distributed Geographic Information System.

2 A Mobile Agent-Based Architecture for Distributed GIS

2.1 The Architecture for Mobile Agent-Based Distributed GIS

In our scheme, the MADGIS environment consists of MADGIS servers, agent docks, users or agent users and the network (e.g. Internet), which is shown in Figure 1. The users access the MADGIS servers through network and are served by mobile agents stored at the corresponding servers. MADGIS servers collect and deliver information by starting up its own mobile agents. The agent dock servers are machines that are permanently connected to the network and used as dock for those agents which are originated from the host that are disconnected at the time when they migrate back to it. Docking system is used to solve the congestion of network and make applications suitable for low reliable network.

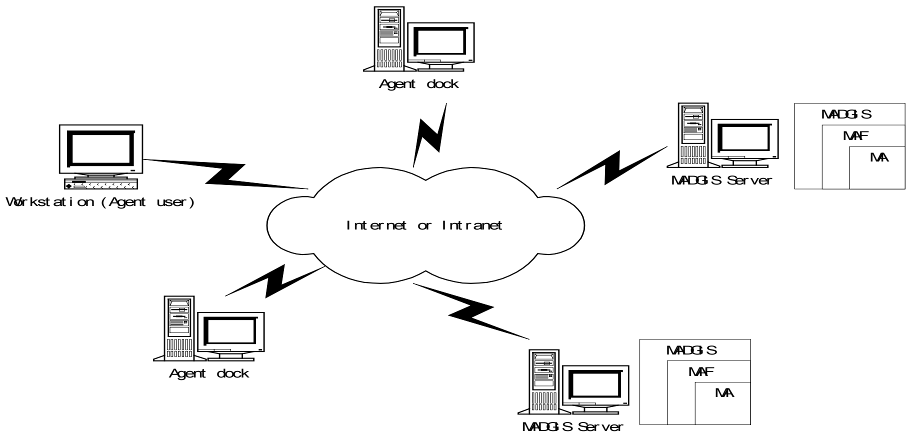


Fig. 1. MADGIS environment

MADGIS server is the core part constructing the MADGIS environment. Basically, it is composed of Graphical User Interface (GUI), Query Manager (QM), Object Manager (OM) and Service Manager (SM).

Figure 2 shows the architecture of MADGIS. User's queries or commands enter the server via the GUI. Transparency is an important feature of distributed GIS so that the main integrated module dispatches the instructions transparently to the corresponding subsystems without exact knowledge of these systems. This is accomplished by creating a heuristic mechanism based on global knowledge that is in charge of finding the appropriate capabilities over the network. The GUI provides access to the system in order to display, query and address commands to specific systems or to particular data and metadata. The GUI interacts directly with the Query Manager.

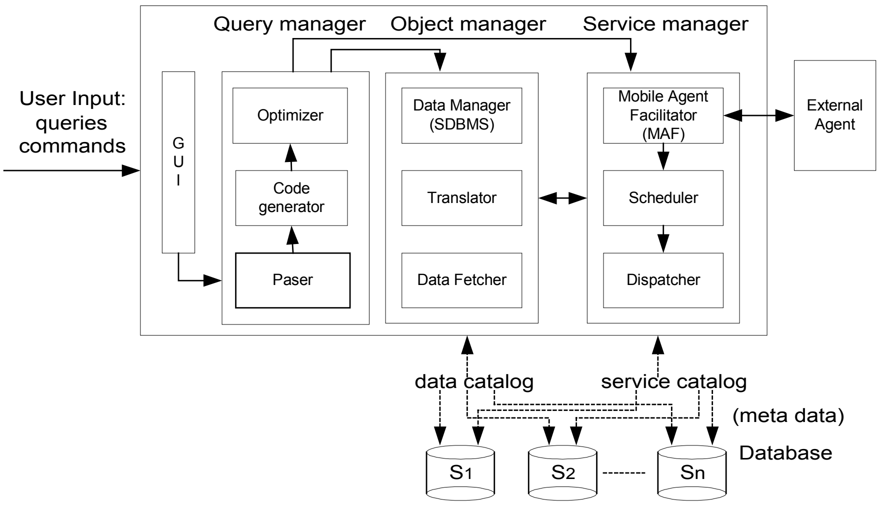


Fig. 2. Architecture of MADGIS

The QM is composed of a Code Generator and an Optimizer. It dispatches the appropriate queries between the Object Manager and the Service Manager. The OM is in charge of fetching relevant objects from repositories and defining, storing them locally. This is accomplished using a virtual database that consists of an integrated view on multiple repositories [2]. Finding objects is concerned with simple selections and distributed spatial data retrieval that is based on a distributed spatial query language. The SM is in charge of all communication services with the aid of a service catalog, which is a resources dictionary. It consists of a scheduler, a dispatcher and a mobile agent facilitator (MAF). The dispatcher is responsible for invoking system calls. The structure of MAF is given in Section 2.2. Alphanumeric data and geo-spatial data, as well as their corresponding metadata, coexist in the MADGIS. The metadata describes the contents of the two kinds of data collections and are stored in a catalog. Concretely, metadata includes data type, location, access rights, owner and creation date. They play a crucial role as they facilitate search, browsing, and data integration. Data related to the services and their locations are stored in a service catalog [2], which is a set of services associated with the system and the description of the system.

2.2 MAF Structure

We design the MAF in MADGIS in line to the Concordia mobile agent system. Concordia features in cross-platform, high-effect, high security and reliability, wide support for distributed applications and cooperative work, which make it advantageous for development of mobile-agent applications [3]. The structure of MAF is illustrated in Figure 3.

Administrator is responsible for the management of whole MADGIS System. A MADGIS node is a host with GIS software and MAF. Multiple MADGIS nodes are connected to form a MADGIS network, and a mobile agent system of MADGIS includes at least a Java Virtual Machine, a MAF and a mobile agent as well as related GIS software. Generally, a Mobile Agent System of MADGIS has several MAFs simultaneously running on different nodes in the network. The MAF is composed of Mobile Agent Manager (MAM), Mobile Agent Transportation (MAT), Mobile Agent Naming Service (MANS), Mobile Agent Communication (MAC), Mobile Agent Security Manager (MASM), and Collaboration Service (CS).

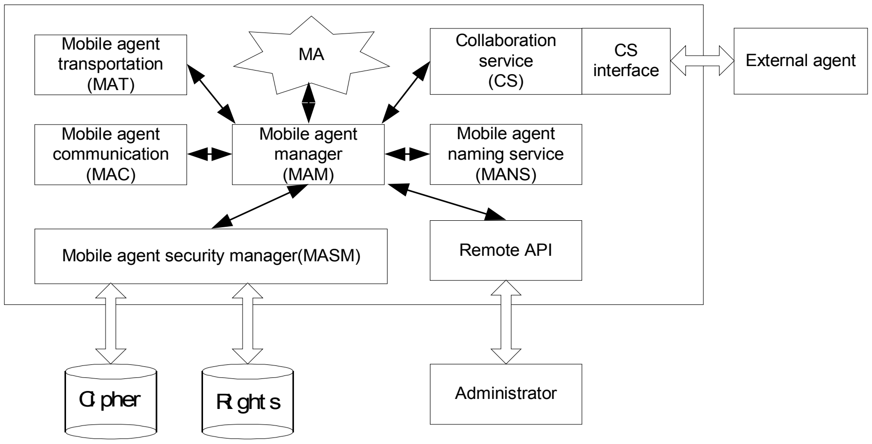

Mobile agent transportation (MAT)
MA
Collaboration service (CS)
CS interface
External agent
Mobile agent communication (MAC)
Mobile agent manager (MAM)
Mobile agent naming service (MANS)
Mobile agent security manager(MASM)
Remote API
Cipher
Rights
Administrator

Fig. 3. Structure of mobile agent facilitator (MAF)

2.3 The Operation Process of MADGIS

The operation process of MADGIS covers the whole life cycle of a mobile agent from its creation, execution, and migration till its end.

When a user inputs queries through the GUI, the queries are analyzed and optimized by Query Manager, then the optimization result is dispatched to Service Manager, and MAF takes over the task. MAM creates an (multiple) agent(s) and starts the agent. If the task requires the agent move, the agent sends request to MAM asking to move. After receiving the request, MAM examines its validness, records related information of the agent to local database, and makes preparation for moving. Then MAM asks MANS to lock the name registration information of the agent, and transfers the agent to MAT when locking requirement is permitted. With certain security methods packed, the agent starts to migrate along the network in light of certain route strategies. After searching on MADGIS server A, the agent can decide the next behavior according to certain route strategies, constraints, network status, server's load, and etc. The agent blocks it itself and asks another MADGIS server B for migration. When it obtains permission, the agent replicates it at server B and resumes its breakpoint status in server B to continue its information-retrieving task. If the generator node is connected to the net and the user's MAF is started, the agent will go back to its user's end directly when the task has been accomplished or it is up to the return time. Otherwise, the agent will stay in the Dock server temporarily. When the agent is averted to Dock server, the server will download the agent partially or wholly to hard disk, and monitor the destination node instead of the agent. Once the transfer condition is met, it will activate the agent and make it transferred to the destination node. When arriving at the destination node, the agent submits its searching results to the user, then stops its execution and ends the task cycle.

3 Key Techniques for MADGIS Implementation

Implementation of MADGIS involves various technologies related to multiple fields, such as GIS, computer network, distributed computing and Artificial Intelligence, and etc. We outline some main technologies here to sketch our solutions.

Data modeling and distributed queries. Information in MADGIS is classified as static and dynamic. With regard to static information, a relational object-oriented database model is used; while to dynamic information, a temporal relational object-oriented database model and relational database techniques are developed. Queries in MADGIS are classified into local and global ones. Local queries are handled efficiently on local computers, while global queries are performed through creating an agent (multiple agents) and dispatching it (them) to different base stations.

Mobility of Mobile Agents. To develop a Mobile Agent system, MADGIS provides a Mobile Agent's base class, in which many essential services requested in the process of agent move and execution are encapsulated. The route of move is decided by itinerary, which can be changed dynamically. MADGIS has three modes to move: push mode, pull mode and push-pull mode.

Collaboration among mobile agents. MADGIS has two kinds of mechanism for communication among agents: asynchronous distributed events and collaboration. There are two kinds of asynchronous distributed events: selective events and group-oriented events [4]. Agents achieve collaboration through group-oriented events. At present, the MAF in MADGIS provides two kinds of Event Group: basic Event Group and persistent Event Group.

Some other related technologies, such as security and reliability, are also very important for implementation. Due to space limited, we give no more details.

4 An Application Prototype and Conclusion

We developed an application prototype based on the MADGIS architecture, which is aimed to set up a distributed GIS to aid drivers find optimal routes in big cities. The distributed GIS covers the whole city. A MADGIS server is installed within each district area to store the roads and streets information (static, spatial data) and traffic information (dynamic, real-time data). All MADGIS servers are connected through the city-area network and the cars exchange information with the servers via wireless network. The server locating in the nearest district will answer the driver's query and direct the car to the destination with the latest optimal route decision. Preliminary test of the prototype system is under way now.

Internet-based distributed GIS systems are rapidly populated as Internet and distributed computing technologies develop. Distributed computing based on Mobile Agent is a new promising computing-mode. The MADGIS architecture can fulfill the requirements of GIS applications under Internet environment. Since distributed GIS is still an emerging research area, there are many problems to be studied.

References

1. Kotz, D., Gray, R. S.: Mobile Agents and the Future of the Internet. ACM Operating Systems Review, Vol.33 (3), (1999) 7-13
2. Jacobsen, H. A., Voisard, A.: CORBA-Based Interoperable Geographic Information Systems. ftp://ftp.icsi.berkeley.edu/pub/techreports/1998/tr-98-011.ps.gz
3. Mitsubishi Electric Information Technology Center America (ITA). CONCORDIA-Redefining Mobile Agent Technology. (1998) http://www.meitca.com/HSL/Projects/Concordia/mobile.htm
4. Wong, D., Paciorek, N., Walsh, T., *et al*: Concordia: An Infrastructure for Collaborating Mobile Agents. Lecture Notes in Computer Science, Vol.1219. Springer-Verlag, Berlin, Germany (1997)

Global View Maintenance by Using Inference Relationship among Views

Haifeng Liu, Wee-Keong Ng, and Chaohui Li

Centre for Advanced Information Systems, School of Applied Science
Nanyang Technological University, Singapore 639798, Singapore
p144330195@ntu.edu.sg

Abstract. Previous work on view maintenance mostly focused on updating a single view at a time. However, to maintain a large warehouse whose source data changes rapidly, an overall maintenance strategy is needed. This paper concentrates on reducing the time taken for updating an entire set of related SPJ views based on the logical inference relationships among the views. Updating a view in response to the changes of the source data needs two types of computations: decision computation and refresh computation. The decision computation determines whether the source change affects the view while the refresh computation installs the source change into the view. By investigating the inference relationships among SPJ queries based on boolean expression relationships, an algorithm is developed to reduce the necessary computation for maintaining a set of views. The more related the SPJ views in the warehouse are, the better the performance of the algorithm is expected to be.

1 Introduction

A *data warehouse* is a repository of integrated information, available for queries and analysis (e.g., decision support, or data mining). *Materialized views* stored in the warehouse are copies of data derived from original data. One of the main problem is to change materialized views when the source data is updated in order to keep the consistency of the warehouse. Previous work has developed various incremental view maintenance algorithms to reduce the time taken for updating the views. Most of them focused on updating a single warehouse view. However, due to the constantly increasing size of warehouses and the rapid update rates of source data, a perfect incremental maintenance algorithm by which each view is maintained in isolation may not work. A simple example illustrates the situation:

Example 1. Consider a source relation R updated at a constant interval four units of time. Five views are affected by the updates of R and it takes one unit of time for the view maintainer (single processor) to incrementally compute the change of each view. Thus, the maintainer totally needs five units of time to refresh all five views in response to an update of R. However, this cannot be satisfied due to the periodical update of R that happens once every four units of time. Only four of views can be maintained timely. ■

Here, the *view maintainer* is the processor to perform the view maintenance. Clearly, two kinds of computations are executed on the view maintainer when the source update is propagated to the maintainer. Firstly, the maintainer needs to determine whether the specific view is affected by the update (i.e., insertion, deletion of tuples). Only when the view is affected by the update does the maintainer execute the second computation, to compute the changes made to the view and install the changes into it. We refer to the two computations as *decision computation* and *refresh computation* respectively. Although the decision computation usually takes less time than the refresh computation, it does make an impact on timely refreshing a large set of views. Our method to save the processor time of the maintainer focuses on saving these two computations. The technique is to reduce the amount of computation on both decision and refresh by observing the relationships among all views, which will be discussed in detail in Section 3.

The following example illustrates our basic method:

Example 2. Based on two relations r and s defined on $R = \{A, B\}$ and $S = \{C, D\}$ respectively, three views are defined as: $v_1 = \Pi_{A,D}\sigma_{(A<4)\wedge(B=C)}(r \bowtie s)$, $v_2 = \Pi_{A,C}\sigma_{(A<2)\wedge(C>2)\wedge(B=C)}(r \bowtie s)$, $v_3 = \Pi_{A,B}\sigma_{A>3}r$.

r	A	B
	2	3
	4	5
	1	2

s	C	D
	3	6
	5	0
	2	4

v_1	A	D
	2	6
	1	4

v_2	A	C

v_3	A	B
	4	5

Consider maintaining the views while **r** *is modified once every six units of time. Assuming that the maintainer takes one unit of time for each decision computation and two time units for each refresh computation. Suppose a tuple $(1, 3)$ is inserted into* **r** *at time instance t_1. Clearly, v_1 and v_2 need to be refreshed in response to this update. But if the maintainer maintains each view separately, it needs to take three units of time to determine whether the views are affected by the insertion and additional four units of time to refresh v_1 and v_2 respectively. So total seven units of time is needed to maintain the views. Unfortunately, this maintenance cannot be completed timely before time $t_1 + 6$ when the next update happens on* **r**.

However, observing the relationships among the views we find that the tuples in v_2 can never appear in v_3 and definitely be contained in v_1. If the maintainer knows these relationships beforehand, when the tuple $(1, 3)$ is inserted into **r** *at time t_1, after the maintainer takes one unit of time to know that v_2 needs to be changed by evaluating boolean expression $(1 < 2)\wedge(C > 2)\wedge(3 = C)$ and checking the content of relation* **s**, *it can infer that v_3 is not affected by the insertion and v_1 needs to be changed instead of executing three decision computations. Another saving happens when the maintainer executes refresh computation for v_1 and v_2. Since the insertion makes the same effect on the states of v_1 and v_2, the maintainer needs to execute only one join computation $\Delta = (1, 3) \bowtie s$ and then simply inserts the results of $\Pi_{A,D}\Delta$ and $\Pi_{A,C}\Delta$ respectively into v_1 and v_2,* *rather than individually executing two refresh computations which both include join operation. Thus, the maintainer totally needs less than five units of time to maintain all views, which then results in timely satisfying the periodic update requirement of* **r**. ∎

Thus, to save the utilization of the maintainer, we must take full advantage of all the information available, namely, not only the definitions of the views, but also the *inference relationships* among the views.

In this paper, we formally define the inference relationships among SPJ (select-project-join) views. Based on that, we develop an algorithm to reduce computation needed to maintain a set of views.

The paper is organized as follows: Section 2 reviews some related work in the area of view maintenance where we differentiate our work with theirs. In Section 3, we formally build the inference relationships between two related SPJ views based on boolean expression relationships and summarize the relationships among a set of views into a *view relationship reference table* (VRRT). In Section 4, with the aid of VRRT a global view maintenance algorithm (GVM) is developed to efficiently reduce the amount of decision computation and refresh computation when maintaining a set of views. We conclude our work in Section 5 and discuss the performance of GVM.

2 Related Work

A survey for traditional maintenance of materialized views has been shown in [1]. The authors motivated and described materialized views, their applications, and the problems and techniques for their maintenance. They classified view maintenance problems based upon the class of views considered, upon the resources used to maintain the view, upon the types of modifications to the base data that are considered during maintenance, and whether the technique works for all instances of databases and modifications.

Many incremental view maintenance algorithms have been developed [3,5,6,10,11]. These algorithms differ somewhat in the view definitions they handle. For example, [4] considers select-project-join (SPJ) views only, while algorithms in [2] handle views defined by any SQL or Datalog expression. Some algorithms depend on key information to deal with duplicate tuples [8], while others use a counting approach [2]. Note that although views in this paper are described by SPJ, our idea can be extended to any types of view definition.

However, most of the previous work in literature concentrate on the maintenance of a single warehouse view. Only until recently does the importance of multiple-view maintenance push researchers into the area. A DAG of views has been concerned by Labio, Yerneni and Garcia-Molina [9]. They developed an efficient algorithm for selecting the optimal strategy to update a set of views. However, they focused on the update of a set of derived views depicted by a *view directed acyclic graph* (VDAG) and an assumption has been made that no decision computation which is discussed in our work needs to be performed.

3 View Relationship Reference Table

3.1 View Relationships

Formally, we define a materialized view as a stored relation in the warehouse resulting from the evaluation of a relational algebra expression against several source relations. In this paper, we consider only relational algebra expressions formed from the combination of selections, projections, and joins, called *SPJ expressions*. Thus, a view can be expressed by the expression $v = \Pi_X(\sigma_{F(Y)}(r_1 \times r_2 \times \ldots \times r_n))$ where $F(Y)$ is a Boolean expression and X and Y are sets of variable denoting the names of (some) attributes for the relation $r_1, r_2, \ldots, r_n$. In addition, we denote the source tables involved with v as $R(v) = \{r_1, r_2, \ldots, r_n\}$. We can define the inference relationships between two views according to the different relationships between their selection conditions:

Definition 1 (View Inference Relationships). *Let $v_i = \Pi_{X_i}(\sigma_{F_i(Y_i)}(r_{i_1} \times r_{i_2} \times \ldots \times r_{i_n}))$ and $v_j = \Pi_{X_j}(\sigma_{F_j(Y_j)}(r_{j_1} \times r_{j_2} \times \ldots \times r_{j_n}))$, if we select v_i as the* **base view***, then six kinds of relationships between the two views are denoted as follows:*

- *$v_i \rightarrow v_j$ if the fact that $F_i(Y_i)$ is satisfiable implies that $F_j(Y_j)$ is satisfiable.*
- *$v_i \rightarrow \neg v_j$ if the fact that $F_i(Y_i)$ is satisfiable implies that $F_j(Y_j)$ is unsatisfiable.*
- *$v_i \not\rightarrow v_j$ if the satisfiability of $F_j(Y_j)$ cannot be inferred from the fact that $F_i(Y_i)$ is satisfiable.*
- *$\neg v_i \rightarrow v_j$ if the fact that $F_i(Y_i)$ is unsatisfiable implies that $F_j(Y_j)$ is satisfiable.*
- *$\neg v_i \rightarrow \neg v_j$ if the fact that $F_i(Y_i)$ is unsatisfiable implies that $F_j(Y_j)$ is not unsatisfiable.*
- *$\neg v_i \not\rightarrow v_j$ if the satisfiability of $F_j(Y_j)$ cannot be inferred from the fact that $F_i(Y_i)$ is unsatisfiable.* ■

Thus, according to the above rules, we obtain the relationships among the three views in Example 2 as: $v_2 \rightarrow v_1$ and $v_2 \rightarrow \neg v_3$ if v_2 is selected as the base view.

The theorem built in [4] as follows provides us a base for our results.

Theorem 1. *Given a view $v = \Pi_X(\sigma_{F(Y)}(r_1 \times r_2 \times \ldots \times r_n))$, when a tuple t is inserted into (or deleted from) r_i defined on the schema R_i for some i, $1 \leqslant i \leqslant n$, let $Y_1 = Y \cap R_i$ and $Y_2 = Y - Y_1$, the update involving tuple t is* irrelevant *to v if and only if $F(t, Y_2)$ is unsatisfiable where $F(t, Y_2)$ is the modified selection condition $F(Y)$ obtained when substituting the value $t(A)$ for each occurrence of the variable $A \in Y_1$ in $F(Y)$. Expression $F(t, Y_2)$ is said to be a* substitution *of t for Y_1 in F.* ■

Thus, for an update on the source r (insert or delete a tuple t), denoted as $\Delta(r) = Insert(t)$ or $\Delta(r) = Delete(t)$, we build the *decision rules* for inferring the relevance of a view v_j with the $\Delta(r)$ from another view v_i where $r \subseteq R(v_i)$ and $r \subseteq R(v_j)$.

Theorem 2. *If an update $\Delta(r)$ is relevant to v_i, then*

- *the update is relevant to v_j if $v_i \rightarrow v_j$, denoted as $v_i \xrightarrow{r} v_j$;*
- *the update is irrelevant to v_j if $v_i \rightarrow \neg v_j$, denoted as $v_i \xrightarrow{r} \neg v_j$;*
- *the effect of the update on v_j cannot be inferred if $v_i \not\rightarrow v_j$, denoted as $v_i \overset{r}{\not\rightarrow} v_j$.*

Otherwise, if $\Delta(r)$ is irrelevant to v_i, then

- *the update is relevant to v_j if $\neg v_i \rightarrow v_j$, denoted as $\neg v_i \xrightarrow{r} v_j$;*
- *the update is irrelevant to v_j if $\neg v_i \rightarrow \neg v_j$, denoted as $\neg v_i \xrightarrow{r} \neg v_j$;*
- *the effect of the update on v_j cannot be inferred if $\neg v_i \not\rightarrow v_j$, denoted as $\neg v_i \overset{r}{\not\rightarrow} v_j$.* ∎

The theorem can be straightforward proved from the definition of the view inference relationships. Therefore, we can save the decision computation when the above rules apply to a set of views with known relationships among them.

Note that the core of deciding view relationships is to decide Boolean expression relationships. Inevitably, we must decide the satisfiability of Boolean expressions, which is in general NP-complete. However, there is a large class of Boolean expressions for which satisfiability can be decided efficiently, as shown by Rosenkrantz and Hunt [7]. This class corresponds to expressions formed from the conjunction of atomic formulae of the form $x\ op\ y$, $x\ op\ c$, and $x\ op\ y + c$, where x and y are variables defined on discrete and infinite domains, c is a positive or negative constant, and $op \in \{=, <, >, \leq, \geq\}$. The sketch of the algorithm to deciding the satisfiability of a conjunctive expression in the class is shown in [4].

3.2 View Relationship Reference Table

In this paper, we use a *view relationship reference table* (VRRT) to store relationships among a set of views involving a same base relation. If m views are involved with the base relation r, then the VRRT for r is a table with m rows and m columns. An element $e_{i,j}$ in VRRT ($i \neq j$) is a tuple $(\mu_{i,j}, \nu_{i,j})$, where $\mu_{i,j}, \nu_{i,j} \in \{-1, 0, 1\}$ are decided by the rules:

$$\mu_{i,j} = \begin{cases} 1 & \text{if } v_i \xrightarrow{r} v_j \\ -1 & \text{if } v_i \xrightarrow{r} \neg v_j \\ 0 & \text{if } v_i \overset{r}{\not\rightarrow} v_j \end{cases} \quad \nu_{i,j} = \begin{cases} 1 & \text{if } \neg v_i \xrightarrow{r} v_j \\ -1 & \text{if } \neg v_i \xrightarrow{r} \neg v_j \\ 0 & \text{if } \neg v_i \overset{r}{\not\rightarrow} v_j \end{cases}$$

Thus, we build a VRRT for relation r in Example 2 as follows:

	v_1	v_2	v_3
v_1		$(0, -1)$	$(0, 0)$
v_2	$(1, 0)$		$(-1, 0)$
v_3	$(0, 0)$	$(-1, 0)$	

4 Global View Maintenance Algorithm

In this section, with the aid of VRRT we propose a global view maintenance (GVM) algorithm to incrementally maintain a set V of views in response to a source update $\Delta(r)$ occurred in the relation r.

Before introducing GVM, we first associate each view v_i $(1 \leqslant i \leqslant m)$ in V with two weights: *decision weight* w_i^d and *refresh weight* w_i^r, which are respectively defined as

$$w_i^d = \sum_{j=1}^{m} |\mu_{i,j}| + \sum_{j=1}^{m} |\nu_{i,j}|$$

and

$$w_i^r = \sum_{j} \mu_{i,j}, j \in \{x | 1 \leqslant x \leqslant m \wedge \mu_{i,x} = 1\}$$

where a pair $(\mu_{i,j}, \nu_{i,j})$ $(1 \leqslant j \leqslant m)$ is an element in the VRRT.

The main idea of GVM is to divide V into two groups V_R and V_U in response to an update $\Delta(r)$, where V_R is the set of views relevant to $\Delta(r)$ whose contents may change while V_U is the set of views irrelevant to $\Delta(r)$ whose contents definitely should not change. Initially, both V_R and V_U are empty. Then, GVM is executed as two steps. Firstly, choose a base view having the maximal decision weight from V and perform the decision computation on it. Then, look up VRRT and add other views inferred from the base view into respective V_R and V_U. This step is repeated until all views in V have been decided to being added into V_R or V_U. Secondly, choose a base view having the maximal refresh weight from V_R and perform the corresponding refresh computation on it. Then, look up VRRT and install the changes into the views inferred from the base view. This step is repeated until all views in V_R have been refreshed. The detailed algorithm is shown in Figure 1.

Applying GVM in the set of views in Example 2, we have $V = \{v_1, v_2, v_3\}$. According to the VRRT on $\mathbf{r}$ shown in the preceding section, the decision weights for three views are $w_1^d = 1$, $w_2^d = 2$ and $w_3^d = 1$ and the refresh weights for three views are $w_1^r = 1$, $w_2^r = 0$ and $w_3^r = 0$. Thus, when a tuple $(1, 3)$ is inserted into $\mathbf{r}$, v_2 is selected as the base view in the decision step and v_1 is selected as the base view in the refresh step. Proceeding with the algorithm, for maintaining the whole view set the maintainer needs to totally performs 2 decision computations on v_2 and v_1 respectively and 1 refresh computation on v_1 only. Compared to the traditional one-by-one maintenance method where 3 decision computations and 2 refresh computations are needed, we have saved 1 decision computation and 1 refresh computation.

5 Conclusion

We have discussed in this paper how to reduce unnecessary decision computation and refresh computation when incrementally maintaining a set of related views. Rather than performing maintenance on each view one by one, our optimization

```
$V_R \leftarrow \varnothing$, $V_U \leftarrow \varnothing$;
do {
    select a base view $v_h$ which has the maximal decision weight among views in
    $V - V_R - V_U$;
    perform decision computation for $v_h$ in response to the update $\Delta(r)$;
    if $v_h$ is relevant to $\Delta(r)$
        add $v_h$ into $V_R$;
        foreach view $v_j \in V - V_R - V_U$, $j \neq h$
            look up the element $(\mu_{h,j}, \nu_{h,j})$ in VRRT;
            if $(\mu_{h,j} = 1)$
                add $v_j$ into $V_R$;
            if $(\mu_{h,j} = -1)$
                add $v_j$ into $V_U$;
    else
        add $v_h$ into $V_U$;
        foreach view $v_j \in V - V_R - V_U$, $j \neq h$
            look up the element $(\mu_{h,j}, \nu_{h,j})$ in VRRT;
            if $(\nu_{h,j} = 1)$
                add $v_j$ into $V_R$;
            if $(\nu_{h,j} = -1)$
                add $v_j$ into $V_U$;
} while $(V \neq V_R \cup V_U)$
do {
    select a base view $v_r$ which has the maximal refresh weight among views in $V_R$;
    perform refresh computation for $v_r$ in response to the update $\Delta(r)$;
    install $\Delta(v_r)$ on $v_r$;
    remove $v_r$ from $V_R$;
    foreach view $v_j \in V_R$
        look up the element $(\mu_{r,j}, \nu_{r,j})$ in VRRT;
        if $(\mu_{r,j} = 1)$
            remove $v_j$ from $V_R$;
            install $\Delta(v_r)$ on $v_j$;
} while $(V_R \neq \varnothing)$
```

Fig. 1. The global view maintenance algorithm.

technique is to utilize the logical implication relationships among views to infer the effects of source updates on views. In response to a source update, after computing the update of the base view, we can save the additional computation on a specific view by inference from the known implication relationship between the base view and this specific view.

We have proposed a global view maintenance algorithm (GVM) which can achieve better performance on a set of more related views. Consider a set of m views, if we separately maintain them one by one in response to a source update, then we may perform m decision computations and m refresh computations if they all need to be changed. However, with GVM, imagine the best situation where we have a base cell from which effects on all other views from a source update can be inferred, thus we only need to perform one decision computation and one refresh computation on the base cell and some other computations with relatively low costs such as looking up VRRT and installing updates to the views. Certainly, the more related the maintained views are, the better the performance of GVM is. In our future work, we will evaluate the performance of GVM for different set of views where the tightness and strongness of relationships among views vary in different degrees.

References

1. Ashish Gupta, Inderpal Singh Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering*, 18(2):3–18, June 1995. 293
2. Ashish Gupta, Inderpal Singh Mumick, V. S. Subrahmanian. Maintaining views incrementally. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 157–166, Washington, D.C., 26–28 May 1993. 293
3. Nam Huyn. Multiple-view self-maintenance in data warehousing environments. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pages 26–35, 1997. 293
4. J. A. Blakeley, P.-A. Larson, F. W. Tompa. Efficiently updating materialized views. In *sigmod*, pages 61–71, Washington, DC, May 1986. acm. 293, 294, 295
5. J. V. Harrison, S. W. Dietrich. Maintenance of materialized views in a deductive database: An update propagation approach. In *Proceedings of the 1992 JICLSP Workshop on Deductive Databases*, pages 56–65, 1992. 293
6. O. Shmueli, A. Itai. Maintenance of views. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 240–255, Boston, Massachusetts, May 1984. 293
7. Rosenkrantz, J. Daniel, H. B. Hunt. Processing conjunctive predicates and queries. In *Proc. of the 6th International Conference on Very Large Data Bases*, pages 64–72, 1980. 295
8. S. Ceri, J. Widom. Deriving production rules for incremental view maintenance. In *Proceedings of the Seventeenth International Conference on Very Large Data Bases*, pages 577–589, Barcelona, Spain, September 1991. 293
9. W. J. Labio, R. Yerneni, H. Garcia-Molina. Shrinking the warehouse update window. In *Proceedings of the 1999 SIGMOD*, 1999. 293
10. X. Qian, G. Wiederhold. Incremental recomputation of active relational expressions. *IEEE Transactions on Knowledge and Data Engineering*, 3:337–341, September 1991. 293
11. Yue Zhuge, H. Garcia-Molina, J. Hammer, J. Widom. View maintenance in a warehousing environment. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 316–327, San Jose, California, 22–25 May 1995. 293

Maintaining Materialized Views for Data Warehouses with Multiple Remote Sources

Weifa Liang[1], Chris Johnson[1], and Jeffrey X. Yu[2]

[1] Department of Computer Science,The Australian National University
Canberra, ACT 0200, Australia
[2] Department Systems Engineering and Engineering Management
Chinese University of Hong Kong, Hong Kong

Abstract. A data warehouse is a data repository which collects and maintains a large amount of data from multiple distributed, autonomous and possibly heterogeneous data sources. Often the data is stored in the form of materialized views in order to provide fast access to the integrated data. However, to maintain the data in the warehouse consistent with the source data is a challenging task in a multiple remote source environment. Transactions containing multiple updates at one or more sources further complicate the consistency issue. In this paper we first consider improving the refresh time of select-project-join (SPJ) type materialized views in a data warehouse by presenting a frequency-partitioned based algorithm, which takes into account the source update frequencies and the total space for auxiliary data. We then propose a solution in the design of data warehouses which can handle a variety of materialized views with different refreshment requirements.

1 Introduction

Data warehouses usually store materialized views in order to provide fast access to the data integrated from several distributed data sources [5], while the data sources are heterogeneous and remote from the warehouse. In the study of data warehousing, one important problem is the view maintenance problem which aims to maintain the content of a materialized view at a certain level of consistency with the source data when updates commit at sources. Consequently, the view maintenance problem with multiple remote sources is totally different from its counterpart in a centralized data warehouse.

Related work: Many incremental maintenance algorithms for materialized views have been introduced for centralized database systems [2,3,5,8,6]. There are also a number of researchers investigating similar issues in distributed environments [4,9,10,20,21,22]. A good overview of the work done in this area can be found in [5]. These previous works have formed a spectrum of solutions ranging from a fully virtual approach at one end where no data is materialized and all user queries are answered by interrogating the source data [10], to a full replication at the other end where the whole databases at the sources are copied to the warehouse so that the view refreshments (updates) can be handled in the warehouse locally [7,15,10]. The two extreme solutions are inefficient in terms of communication and query response time in the former case, and storage space in the latter. An efficient solution is to materialize some relevant subsets of source data in the warehouse (usually the query answers), the warehouse is able to refresh the materialized data incrementally against the updates when any relevant updates at sources are propagated to the warehouse. However, in a distributed source environment, this approach may necessitate a solution in which the warehouse contacts the sources for additional information to ensure the update correctness, in order to maintain the data in the warehouse at a certain level of consistency with the source data [21,20,4,1,9].

In a multiple remote source environment, how to improve the refresh time of a materialized view is very important. In [16] they considered this problem by allocating extra space to the view, i.e., they keep multiple versions of a materialized view in the warehouse. Thus, each tuple in the view has multiple versions with different timestamps. Each subsequent update or query is also with a timestamp. Although this approach allows the view refreshment (view maintenance) and the user query to be processed simultaneously, it still requires that the warehouse contacts the remote sources when performing view refreshment. A similar approach based on timestamps is also given in [12], but this latter algorithm essentially proceeds the periodic updates, not on-line incremental updates that we will consider here. In contrast to keeping multiple versions of a materialized view, there is another approach called *self-maintainable approach* [7,15,13], which keeps all relevant auxiliary information of a materialized view in the warehouse. Thus, the view in the data warehouse can be updated using the source update log and the auxiliary information only when there is any updates at sources, without consulting with the remote sources. This approach requires that the data warehouse run in the two modes (user query and view maintenance) exclusively. As a result, the refresh time of materialized views is reduced significantly, at the expense of lots of space spent for storing the auxiliary information. Similar to the second approach, the third approach also requires that the data warehouse run in user query and view maintenance modes. But no any auxiliary information is stored in the warehouse, the view refreshment is implemented through contacting the remote sources [1,20] during the warehouse maintenance mode. The advantage of this approach is that it does not need any extra space for auxiliary information. On the other hand, the refresh time may take very long, depending on how many sources have to be contacted when performing the view refreshment. Recently a new approach is suggested which just stores partial auxiliary information in the warehouse. At the same time, the warehouse is allowed to have a limited number of accesses to the remote sources [18]. This approach trades-off the two important factors, the amount of available extra space in the warehouse and the refresh time of the views very well. However, they only considered the case where every source has the same update frequency and there is no space limit for auxiliary information.

Our contributions: Following the same spirit in [18], in this paper a complete consistency algorithm for SPJ type of views is proposed. This frequency-partitioned based algorithm groups the sources into several groups such that the sum of source update frequencies of each group is roughly equal and the space for auxiliary views derived from the groups of sources is reasonably small. The algorithm improves the view refresh time at the expense of extra warehouse space to accommodate auxiliary views. Also, a possible solution in the design of data warehouse is suggested, which can handle a variety of materialized views with different refreshment requirements.

2 Preliminaries

2.1 The Data Warehouse Model

A *data warehouse* is a repository of integrated data, which collects and maintains a large amount of data from multiple distributed, autonomous and possibly heterogeneous sources. A typical data warehouse architecture defined in [19] is illustrated in Figure 1. Associated with every source there is a monitor/wrapper

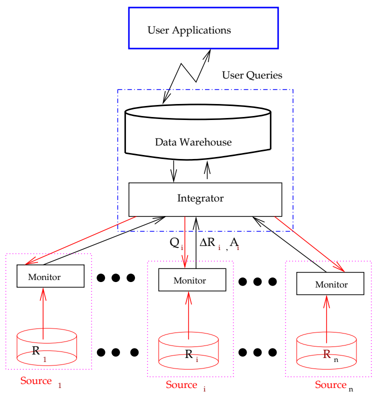


Fig. 1. A data warehouse architecture

which collects the data of interest and sends the data to the warehouse. The monitor/wrapper is responsible for identifying changes and notifying the warehouse. If a source provides relational-style triggers, the monitor may simply pass on information. On the other hand, a monitor for a legacy source may need to compute the difference between successive snapshots of the source data. At the warehouse side, there is an integrator which receives the source data, performs necessary data integration and translation, adds any extra desired information such as the timestamps for historical analysis, and requests the warehouse to store the data. In effect, the warehouse caches a materialized view of the source data. The data is then readily available to user applications for querying and analysis. The communication between a source and the warehouse is assumed to be reliable FIFOs, i.e., messages are not lost on their way and are delivered in the order that they are sent. No communication is imposed between any two sources. In fact, they may be completely independent, and may not be able to communicate with each other.

It is easy to see that the complexity of designing algorithms for view maintenance is closely related to the scope of transactions at the sources. In this model we classify the update transactions into the following three categories: (i) Single update transactions where each update is executed at a single source. (ii) Source local transactions where a sequence of updates are performed as a single transaction. However, all of the updates are directed to a single data source. (iii) Global transactions where the updates involve multiple sources. In this paper we only consider the refreshment problem for SPJ-type materialized views. We further assume that the updates being handled at the warehouse are of types (i) and (ii). The approach described in [21] can be used to extend the proposed algorithms for the updates in type (iii). The source updates can be handled at the warehouse in different ways, depending on how the updates are incorporated into the view at the warehouse. Different levels of consistency of a view have been identified as follows, by Zhuge et al [20] and Hull et al [9]. 1. **Convergence** where the updates are incorporated into the materialized view eventually. 2. **Strong consistency** where the order of the state transformations of the view at the warehouse corresponds to the order of the state transformations at the data sources. 3. **Completeness** where every state of the data sources is reflected as a distinct state at the data warehouse, and each state of views at the warehouse corresponds to a state of data sources. That is, there is a complete order preserving mapping between the states of views and the states of the sources.

As claimed in [21], complete consistency is a nice property since it guarantees that the content of a view is always consistent with its source data in every state. However, this restriction may be too strong in practice. In some cases, the convergence may be sufficient even if there are some invalid intermediate states. In most cases, the strong consistency which corresponds to the batch updates is desirable. In this paper we will focus on the design of maintenance algorithms for complete consistency, which can also be extended to strong consistency easily.

2.2 The Refresh Time of Materialized Views

Let V be a SPJ view derived from n relations $R_1, R_2, \ldots, R_n$, defined as follows.

$$V = \pi_X \sigma_P (R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n) \tag{1}$$

where X is the set of projection attributes, P is the selection condition consisting of the conjunction of disjunctive clauses, and R_i is a database located at a remote source i, $1 \leq i \leq n$. An update to the source data is assumed to be either insert or delete of tuples. A modify is treated as a delete followed by an insert. Also, all the views in the warehouse are based on set semantics.

Given a materialized view V in the warehouse and its two consecutive states ws_i and ws_{i+1}, assume that ws_i is obtained at time t_0 and ws_{i+1} is obtained at time t_1. The refresh time, T_{fresh}, of V from ws_i to ws_{i+1}, is $T_{fresh} = t_1 - t_0$.

To update V incrementally from ws_i to ws_{i+1} due to an update in some source, a number of rounds of communications between the warehouse and the sources is needed. Assume that the warehouse needs to send N queries $Q_{i_1}, Q_{i_2}, \ldots, Q_{i_N}$ to the various sources, and to receive the answers A_{i_1}, $A_{i_2}, \ldots, A_{i_N}$ of the N queries. Then, the refresh time of V from ws_i to ws_{i+1} is $T_{fresh} = \sum_{l=1}^{N}(T_{transfer\_to\_source}(Q_{i_l}) + T_{transfer\_to\_warehouse}(A_{i_l}) + T_{warehouse\_join}(A_{i_l}) + T_{source\_join}(Q_{i_l}))$, where $T_{transfer\_to\_source}(Q_{i_l})$ is the transmission time to send query Q_{i_l} from the warehouse to source i_l; $T_{transfer\_to\_warehouse}(A_{i_l})$ is the transmission time from source i_l to the warehouse by returning the query answer; $T_{warehouse\_join}(A_{i_l})$ is the time for processing the join of A_{i_l} and the concurrent updates received so far, which is evaluated at the warehouse; $T_{source\_join}(Q_{i_l})$ is the time for processing query Q_{i_l} at source i_l, which is evaluated at source i_l. To simplify our later discussion, assume that the transmission time between the warehouse and the sources is symmetric and fixed, denoted by $\bar{t}$. We further assume that the query processing time at either the warehouse or the sources is negligible because the processing time for this part only occurs a small proportion of the entire refresh time, compare to the information transmission time in a wide area network. Also, the number of tuples involved in incremental maintenance of materialized view is relatively small, compared with the size of the materialized view. Therefore, the corresponding query processing time is relatively small. After the simplifications, T_{fresh} can be rewritten as $T_{fresh} = 2N\bar{t}$, which depends on the number of queries sent N by the warehouse.

2.3 An Equal-Partitioned Based Algorithm

In [18] an equal-partitioned based algorithm is proposed, which not only reduces a view refresh time but also maintains the complete consistency between the content of the view and the source data, providing that some extra warehouse space is available. Let V be a SPJ materialized view derived from n relations located at n sources. Assume that p is a given integer.

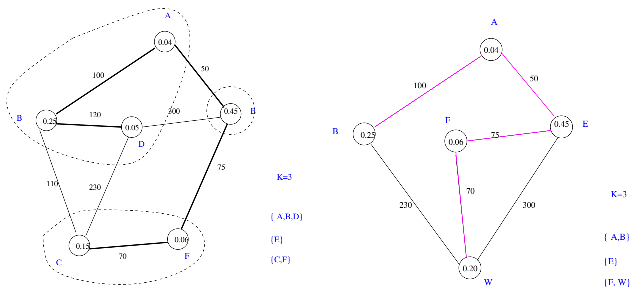
The equal-partitioned algorithm proceeds as follows. First, the n source relations are partitioned into $K = \lceil n/p \rceil$ disjoint groups, and each group consists of p relations except the last group which contains $n - p \times \lfloor n/p \rfloor$ relations. Without loss of generality, assume that the first p relations form group one, the second p relations form group two, and the last $n - p\lfloor n/p \rfloor$ relations form group K. Based on the partition of sources, an auxiliary view V_k for each group k is then derived, $1 \leq k \leq K$, as follows. $V_k = \pi_{X(k)}\sigma_{P(k)}(R_{p(k-1)+1} \bowtie R_{p(k-1)+2} \bowtie \ldots \bowtie R_{pk})$ for all k with $1 \leq k \leq K-1$, where $X(k)$ is an attribute set in which an attribute is either in X or appeared in such a disjunctive clause of P that some its attributes come from the relations which are not in $\{R_{p(k-1)+1}, \ldots, R_{pk}\}$, all the attributes in $X(k) \cup P(k)$ come from relations in $\{R_{p(k-1)+1}, \ldots, R_{pk}\}$ only, and $P(k)$ is a maximal subset of clauses of P in which the attributes of each clause are only from $R_{p(k-1)+1}$ to R_{pk}. The last group K, $V_K = \pi_{X(K)}\sigma_{P(K)}(R_{p(K-1)+1} \bowtie R_{p(K-1)+2} \bowtie \ldots \bowtie R_n)$, where $X(K)$ and $P(K)$ in V_K can be defined similarly as we did for V_k. Thus, V is rewritten in an equivalent form semantically, using the auxiliary views defined, $V = \pi_X \sigma_P (V_1 \bowtie V_2 \bowtie \ldots \bowtie V_{K-1} \bowtie V_K)$.

To refresh V, two assumptions are imposed: (1) V and an auxiliary view V_k are materialized in the warehouse for any k, $1 \leq k \leq K$; (2) the warehouse is capable to decide to which auxiliary view an update belongs and to which auxiliary view a query answer (from sources) belongs. We consider the incremental maintenance of V due to an update ΔR_i at source i, assuming that R_i is in group k. The proposed refreshment algorithm proceeds as follows. First, we perform the incremental maintenance for V_k, using the `SWEEP` algorithm [1]. As a result, an updated view $V_k = V_k \cup \Delta V_k$ is obtained, and so is ΔV_k. We then update V locally without consulting the remote sources. Let V^{new} be the updated result of V, $V^{new} = V \cup \Delta V$ where $\Delta V = \pi_X \sigma_P (V_1 \bowtie V_2 \bowtie \ldots \bowtie V_{i-1} \bowtie \Delta V_i \bowtie \ldots \bowtie V_K)$. Thus, ΔV can be obtained by local evaluation because each V_j is materialized in the warehouse, $1 \leq j \leq K$ with $j \neq k$. To maintain V in complete consistency with the source data, all the other auxiliary views are locked during the incremental maintenance of V_k. Otherwise, the data in the other auxiliary views used later for the maintenance of V may be contaminated during the update. The proposed algorithm has been shown to have complete consistency with the source data. The refresh time of V is $T_{view\_update\_W} = 2(p-1)\bar{t} + t_{WH}$, where t_{WH} is the update time used to update V in the warehouse by joining ΔV_k and all the other V_j with $j \neq k$, which depends on the number of auxiliary views $\lceil n/p \rceil$ participating in the joining and the size of each table involved. From the formula for $T_{view\_update\_W}$, we can see that the choice of the parameter p is very important. If p is chosen too small, although the communication time between the warehouse and the sources can be reduced dramatically, the number of auxiliary views stored in the warehouse $\lceil n/p \rceil$ will increase, which means more space for the auxiliary views is required. Also, the overhead of updating V in the warehouse t_{WH} will increase because more relations need to be joined. Otherwise, there is no any substantial improvement in terms of the view refresh time, despite extra warehouse space spent for auxiliary views.

3 A Frequency Partitioned Based Algorithm

Let V be a SPJ materialized view derived from n sources and f_i be the update frequency of source R_i, $1 \leq i \leq n$ and $\sum_{i=1}^{n} f_i = 1$. To improve the refresh time of V, the relations in the definition of V first are partitioned into K groups (assume that K is given at the beginning) such that the sum of source update frequencies in each group is roughly equal and the total space used for the auxiliary views is limited, where an auxiliary view is derived for each of the groups using the relations in the group and the definition of V. Formally speaking, given a SPJ view V and an integer K, the problem is to find K auxiliary views such that (i) the total space for the auxiliary views is minimized; and (ii) the absolute difference $|\sum_{v \in C_i} f(v) - \sum_{u \in C_j} f(u)|$ is minimized for any two groups C_i and C_j with $i \neq j$, i.e., the sum of source update frequencies in each group is roughly equal, where C_i is the set of source relations. Clearly, this is an optimization problem with two objectives to be met simultaneously. The first objective is to minimize the extra warehouse space to accommodate the auxiliary views. The second objective is to balance the sources' update loads. It is not hard to show that this optimization problem is NP-hard. Instead, we will focus on finding feasible solutions for it by proceeding as follows.

We first construct an undirected weighted graph $G = (N, E, w_1, w_2)$ where each relation in the definition of V is a vertex in N. For every $v \in N$, the associated weight $w_1(v)$ is the update frequency of the corresponding relation. There is an edge $(u, v) \in E$ between $u \in N$ and $v \in N$ if and only if there is a conditional clause in P which contains the attributes only from u and v. The weight $w_2(u, v)$ associated with (u, v) is the size of the resulting table after joining the two tables with the conditional clause, where P is the selection condition in the definition of V. In addition, assume that G is connected. Otherwise, our algorithm will apply to each connected component of G. For example, consider a SPJ view V that consists of six relations. Figure 2(a) is such a graph, where vertex A has weight 0.04 which is the update frequency of the corresponding source. The weight of edge (A, B) is 100, which is the size of the resulting table after joining tables A and B using the conditional clause between them. To



(a) A graph for frequency-based partition

(b) A resulting graph G'.

Fig. 2. An example

avoid the *direct product* joining in generating the auxiliary views, the vertices in a group must be connected in G. Thus, the above optimization problem then becomes as follows. Let $\mathcal{P} = \{C_1, C_2, \ldots, C_K\}$ be a vertex partition of G, i.e., $\cup_{i=1}^{K} C_i = N$ and $C_i \cap C_j = \emptyset$ if $i \neq j$. The problem is to partition the vertices in G into K groups such that (1) the vertices in each group are connected; (2) the weighted sum of vertices in each group is roughly equal, i.e., minimize $|\sum_{v \in C_i} w_1(v) - \sum_{u \in C_j} w_1(u)|$ for any two groups C_i and C_j with $i \neq j$; and (3) the weighted sum of the edges in the forest consisting of K subtrees is minimized. In other words, condition (3) is to minimize the space for the auxiliary views.

3.1 The *K*-subtree Partition Problem

Consider the K-subtree partition problem, defined as follows. Given a tree $T(N, E_1, w_1)$, where w_1 is a weighted function of vertices in T, each vertex $v \in N$ has a weight $w_1(v)$ and $|N| = n$, E_1 is the edge set of T, the problem is to partition T into K subtrees such that the weighted sum of vertices in each subtree is roughly equal. In other words, the problem is to find a K-partition of the tree which maximizes the weight of the lightest weighted subtrees.

There is a naive approach for the problem, described as follows. Each time we first remove $K-1$ tree edges from the tree. As a result, the tree becomes a forest of K subtrees. We then compute the absolute difference of the weighted sums of vertices between any two subtrees and find the maximum one. There are $\binom{n-1}{K-1}$ ways to remove $K-1$ edges from the tree which lead to $\binom{n-1}{K-1}$ different ways of K-subtree partitioning. We finally choose a partition which minimizes the maximum absolute difference in the partition. Obviously, this approach requires $O(n^K)$ time, which grows exponentially when K is not fixed. Another efficient approach to solving the K-subtree partition problem is given in [14], which takes $O(K^2 rd(T) + Kn)$ time, where $rd(T)$ is the number of edges in the radius of T. Note that the K-subtree partition problem was given a different name called *max-min K partition* [14].

3.2 The MST-based Approximation Algorithm

Given V and its source update frequencies, assume that $G(N, E, w_1, w_2)$ defined has been constructed. The MST-based approximation algorithm consists of two stages. In the first stage it optimizes the space for auxiliary views. In the second stage it balances the load of source updates among the auxiliary views. The algorithm for the optimization problem is presented as follows.

```
Appro_Partition($G, N, E, w_1, w_2, K$)
/*$w_1$ is the weight function of vertices and $w_2$ is the weight function of edges*/
1.  Find an MST $T(N, E', w_1)$ from $G$, using the edge weights;
2.  Find a max-min $K$ partition of $T$, using the vertex weights.
3.  The vertices in a subtree form a group and a partition $\mathcal{P}$ of $N$ is obtained.
```

Having the K-vertex partition $\mathcal{P}$, an auxiliary view for each group is then derived using the definition of V. We claim that the space for the auxiliary views generated by the algorithm is reasonably small, which is demonstrated through an example. Consider a view consisting of the joining of three relations R_1, R_2 and R_3. Let s_{12} and s_{23} be the sizes of $R_1 \bowtie R_2$ and $R_2 \bowtie R_3$, respectively. To estimate the size s of $R_1 \bowtie R_2 \bowtie R_3$, a linear cost model is employed, which has also been used in [11] for the similar purpose. Thus, $s = s_{12} + s_{23}$ under this space cost model, the weighted sum of the edges in a minimum spanning tree of a subgraph of G induced by the vertices in a group is equal to the amount of space used for the auxiliary views approximately. Since this space cost model is just an estimate of the real cost. The space for auxiliary views, therefore, is not necessarily optimal, for a given K.

Consider the example given in Figure 2(a) with $K = 3$. A minimum spanning tree in G is first constructed, by applying `Appro_Partition`, which consists of edges $\{(A, B), (A, E), (B, D), (C, F), (E, F)\}$. A 3-vertex partition $\mathcal{P} = \{\{A, B, D\}, \{E\}, \{C, F\}\}$ of the tree is then obtained. In this example, the maximum value of the absolute difference between two groups is 0.24. The space for auxiliary views is $(w_2(A, B) + w_2(B, D)) + S_E + w_2(C, F) = 100 + 70 + 130 + 70 = 370$, assuming that the size S_E of the tuples in table E satisfying the condition of V is 130.

3.3 The Edge Contraction Approximation Algorithm

Here we give another heuristic algorithm for the optimization problem which delivers a solution that better balances the source update load among the auxiliary views.

Let $c > 1$ be constant and $M = \sum_{v \in N} w_1(v)$. The proposed approximation algorithm also consists of two stages. Assume that the current graph is $G(N, E, w_1, w_2)$ initially. In the first stage it proceeds the following iterations until either K vertices are left in the resulting graph or there does exist two neighboring vertices satisfying the *edge-contraction* condition. Each time the algorithm chooses two neighboring vertices u and v such that u and v are the two minimum weighted vertices in the current graph and the sum of $w_1(u) + w_1(v) \leq M/c$, which is referred to the edge-contraction condition. If u and v satisfy the edge contraction condition, they are merged by contracting the edge between them. After the merge, a new vertex w is created. The weight assigned to w is the sum $w_1(u) + w_1(v)$ of the update frequencies of u and v. If either (u, x) or (v, x) but not both of them exists in the current graph, an edge between w and x is then created and assigned weight $w_2(w, x) = w_2(u, x)$, assuming that (u, x) exists. Otherwise, an edge (w, x) is created and is assigned weight $w_2(u, x) + w_2(v, x)$. A resulting graph G' is obtained after the iteration terminated. In the second stage, if G' contains K vertices exactly, then the K vertices form a K-vertex partition of G. The vertices in G merged to a vertex in G' form a group. Otherwise, G' contains more than K vertices, the MST-based approximation algorithm is applied. That is, an MST T' is found for G' first, followed by applying the algorithm [14] for finding a max-min K partition of T'. Clearly, the vertices in each group are connected in G.

Consider the given example in Figure 2(a) with 3-vertex partition ($K = 3$) by applying the edge contraction approximation algorithm. Let $c = 5$. $M = \sum_{v \in N} w_1(v) = 1$. First, two neighboring vertices C and D are chosen to be merged because $w_1(C) + w_1(D) = 0.2 \leq M/c$. The resulting graph G' is generated in Figure 2(b). After that, there is no any neighboring vertices in G' satisfying the edge contraction condition. The algorithm then shifts to the second stage in which an MST T' of G' is obtained, and T' consists of the edges

$\{(A,B),(A,E),(E,F),(F,W)\}$. A 3-vertex partition of T' $\mathcal{P}' = \{\{A,B\},\{E\},\{F,W\}\}$ therefore is obtained. As a result, a corresponding 3-vertex partition of G finally is achieved, which is $\mathcal{P} = \{\{A,B\},\{E\},\{C,D,F\}\}$. In this example, the maximum value of the absolute difference between any two groups is 0.20. The space for auxiliary views is $w_2(A,B)+S_E+(w_2(C,D))+w_2(C,F)) = 100+130+(230+70) = 530$. Compared with `Appro_Partition`, this latter algorithm gives a better solution in terms of load of source updates for each auxiliary view but a worse solution in terms of the space for auxiliary views.

3.4 Optimizing Space for Auxiliary Views

The algorithms above are based on an assumption that the number of auxiliary views K is already given at the very beginning. For a given materialized view, sometimes a smaller or a larger K may lead to small space for auxiliary views. However, when K is too large (almost the linear size of the size of the sources), it is possible to make a duplicate in the warehouse for each source, thus all materialized views in the warehouse are self maintainable. When K is too small, there is no significant improvement in the refresh time despite spending space for auxiliary views. Without loss of generality, we here assume that the number of auxiliary views is between $\lceil \sqrt[3]{n} \rceil$ and $\lfloor \sqrt{n} \rfloor$. Then, we can find a K such that the total space for auxiliary views is minimized by running the following algorithm.

```
Opt_Appro_Partition($G, N, E, w_1, w_2, K, S$)
/*$w_1$is the weight function of vertices and $w_2$ is the weight function of edges,*/
    $K' := \lfloor \sqrt{n} \rfloor$; $Total\_space := S$ /* $S$ is the total space limit. */
    repeat
    1. call Appro_Partition($G, N, E, w_1, w_2, K', S$);
    2. Let $S_{K'}$ be the space required for auxiliary views with the $K'$-partition;
    3. if  $S_{K'} < Total\_space$
    4.    then$Total\_space := S_{K'}$; $K := K'$;
    5. $K' = K' - 1$;
    until    $K' \leq \lceil \sqrt[3]{n} \rceil$.
```

4 Improving Refresh Time of Multiple Views

In this section we deal with how to improve the refresh time for a group of SPJ materialized views instead of a single view. One trivial solution for this problem is to build a set of auxiliary views for each individual materialized view by applying the approaches in the previous section. However, the solution obviously is not the best one because it does not take into account the shared information among the materialized and auxiliary views. In the following we propose an approach which exploits the shared information among the materialized views as well the auxiliary views. Assume that there is a set MV of materialized views with $|MV| = m$. The query frequency of a materialized view $mv_i \in MV$ is q_i and $\sum_{i=1}^{m} q_i = 1$, $1 \leq i \leq m$. Further assume that the materialized views are derived from n remote data sources. Let f_j be the update frequency of source R_j and $\sum_{j=1}^{n} f_j = 1$, $1 \leq j \leq n$. The algorithm is proposed as follows.

```
1. Sort the materialized views by their query frequencies in decreasing order.
     Let $mv_1, mv_2, \ldots, mv_m$ be the sequence of the materialized views after the
     sorting, i.e., $q_1 \geq q_2 \geq \ldots \geq q_m$. Let $AV := \emptyset$ /* the auxiliary view set */
2. for $i := 1$ to $m$ do
2.1 Apply algorithm Appro_Partition to find a $K$-partition of sources for $mv_i$.
     Let $S_{mv_i}$ be the space used by the auxiliary views $AV_{mv_i}$ for $mv_i$.
2.2 Use the views in $U = \{mv_1, mv_2, \ldots mv_{i-1}\} \cup AV$ to rewrite $mv_i$.
     Let $mv_i = ma \cup mb$ where $ma$ is the rewritten part by the existing views
     in $U$ and $mb$ is the original part. Find a set of auxiliary views
     $AVB_{mv_i}$ for $mb$, let $S_b$ be the space requirement for this part.
2.3. If $S_{mv_i} < S_b$ then rewrite $mv_i$ using the auxiliary views in $AV_{mv_i}$;
     $AV = AV \cup \{AV_{mv_i}\}$.
     else rewrite $mv_i$ using the views in $AV = AV \cup \{AVB_{mv_i}\}$.
```

Thus, an auxiliary view set AV for the views in MV has been obtained, which can be used to improve the refreshment of the materialized views in MV.

5 View Maintenance with Different Refreshments

In this section we consider the design of a data warehouse which can handle a variety of materialized views with different refreshment requirements by giving a possible solution. As we know, the refresh time of a materialized view mainly depends on how many remote sources need to be communicated with and how long the communication delay takes. Therefore, to improve the refreshment of a materialized view is to reduce its contact with the remote sources. To this end, we classify the materialized views in a warehouse into three categories, depending on their refreshment requirements. (i) Type one materialized views: the views are the most accessed views and need to be refreshed as quickly as possible. (ii) Type two materialized views: the views are refreshed periodically and must be refreshed within the given refresh time window. (iii) Type three materialized views: the views are only refreshed if no other "urgent" views to be refreshed. The views are refreshed by the user request. The following policy is proposed to handle the three type views.

Let MA_{urgent} be the set of type one materialized views. Apply algorithms in [15,13,17] to find a set of auxiliary views $\mathcal{A}$ such that all views in $\mathcal{A} \cup MA_{urgent}$ are self-maintainable. Thus, once there are any changes in the sources, the views in MA_{urgent} will be refreshed using the source update log without consulting the remote sources. Let MA_{need} be the set of type two materialized views. For each of type two materialized views, rewrite it using the views in $\mathcal{A} \cup MA_{urgent}$ if possible. If it can be rewritten completely, rewrite it using the views. Otherwise, apply the algorithm in Section 4 to find a set of auxiliary views AV for them and rewrite it using the auxiliary views. The refreshment of this kind of view may have a limited number of contacts to the remote sources. For each of type three materialized views, if it can be rewritten using the views in $\mathcal{A} \cup MA_{urgent}$, then rewrite it. Otherwise, rewrite it using the views in $\mathcal{A} \cup MA_{urgent} \cup MA_{need} \cup AV$ if it can be rewritten using the views in $MA_{urgent} \cup \mathcal{A} \cup MA_{need} \cup AV$. In most cases, to implement its refreshment, the view needs to communicate with the remote sources, using the refreshment algorithm like `SWEEP` [1].

Acknowledgement

The work by Weifa Liang and Jeffrey X. Yu was partially supported by a small grant (F00025) from Australian Research Council.

References

1. D. Agrawal, A. El Abbadi, A. Singh, and T. Yurek. Efficient view maintenance at data warehouses. *Proc. of ACM-SIGMOD Conf.*, 1997, 417–427. 300, 304, 310
2. J. A. Blakeley, P. A. Larson, and F. W. Tompa. Efficiently updating materialized views. *Proc. of ACM-SIGMOD Conf.*, 1986, 61–71. 299
3. S. Ceri and J. Widom. Deriving production rules for incremental view maintenance. *Proc. of the 17th VLDB Conf.*, 1991, 577–589. 299
4. L. Colby, T. Griffin, L. Libkin, I. Mumick, and H. Trickey. Algorithms for deferred view maintenance. *Proc. of the 1996 ACM-SIGMOD Conf.*, 1996, 469–480. 299, 300
5. *IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing*. 18(2), June, 1995. 299
6. T. Griffin and L. Libkin. Incremental maintenance of views with duplicates. *Proc. of ACM-SIGMOD Conf.*, 1995, 328–339. 299
7. A. Gupta, H. Jagadish, and I. Mumick. Data integration using self-maintainable views. *Proc. 4th Int'l Conf. on Extending Database Technology*, 1996, 140–146. 300
8. A. Gupta, I. Mumick, and V. S. Subrahmanian. Maintaining views incrementally. *Proc. of ACM-SIGMOD Conf.*, 1993, 157–166. 299
9. R. Hull and G. Zhou. A framework for supporting data integration using the materialized and virtual approaches. *Proc. of ACM-SIGMOD Conf.*, 1996. 299, 300, 302
10. R. Hull and G. Zhou. Towards the study of performance trade-offs between materialized and virtual integrated views. *Proc. of Workshop on Materialized Views: Techniques and Applications*, Montreal, Canada, 1996, 91–102. 299, 300
11. W. J. Labio, R. Yerneni, H. Garcia-Molina. Shrinking the warehouse update window. *Proc. of ACM-SIGMOD Conf.*, 1999, 383–394. 306
12. A. Labrinidis and N. Roussopoulos. Reduction of materialized view staleness using online updates. TR3878, DCS, Univ. of Maryland at College Park, 1998. 300
13. W. Liang, H. Li, H. Wang and M. Orlowska. Making multiple views self-maintainable in a data warehouse. *DKE*, 30(2), 1999, 121–134. 300, 309
14. Y. Perl and S. R. Schach. Max-min tree partitioning. *J. ACM*, 28(1), 1981, 5–15. 306, 307

15. D. Quass, A. Gupta, I. S. Mumick, and J. Widom. Making views self-maintainable for data warehousing. *Proc. of PDIS'96*, FL, 1996, 158–169. 300, 309
16. D. Quass and J. Widom. On-line warehouse view maintenance. *Proc. of ACM-SIGMOD Conf.*, Tucson, Arizona, 1997, 393–404. 300
17. D. Theodoratos, S. Ligoudistianos, and T. Sellis. Designing the global data warehouse with SPJ views. *Proc. of the 11th Int'l Conf. on Advanced Information Systems Engineering*, 1999, 180–194. 309
18. H. Wang, M. Orlowska, and W. Liang. Efficient refreshment of materialized views with multiple sources. *Proc. of the 8th ACM-CIKM*, 1999, 375–382. 300, 303
19. J. Wiener, H. Gupta, W. Labio, Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom. A system prototype for warehouse view maintenance. *Proc. of Workshop on Materialized Views*, Montreal, Canada, 1996, 26–33. 301
20. Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom. View maintenance in a warehousing environment. *Proc. of ACM-SIGMOD Conf.*, 1995, 316–327. 299, 300, 302
21. Y. Zhuge, H. Garcia-Molina, and J. L. Wiener. The strobe algorithms for multi-source warehouse consistency. *Proc. of PDIS'96*, FL, 1996, 146–157. 299, 300, 302
22. Y. Zhuge, H. Garcia-Molina, and J. L. Wiener. Multiple view consistency for data warehousing. *IEEE ICDE'97*, Birmingham, UK, 1997, 289–300. 299

Using Loglinear Models to Compress Datacubes[⋆]

Daniel Barbará and Xintao Wu

George Mason University, ISE Dept.
MSN 4A4 Fairfax VA 22030, USA
dbarbara,xwu@gmu.edu

Abstract. A data cube is a popular organization for summary data. A cube is simply a multidimensional structure that contains in each cell an aggregate value, i.e., the result of applying an aggregate function to an underlying relation. In practical situations, cubes can require a large amount of storage, so, compressing them is of practical importance. In this paper, we propose an approximation technique that reduces the storage cost of the cube at the price of getting approximate answers for the queries posed against the cube. The idea is to characterize regions of the cube by using statistical models whose description take less space than the data itself. Then, the model parameters can be used to estimate the cube cells with a certain level of accuracy. To increase the accuracy, and to guarantee the level of error in the query answers, some of the "outliers" (i.e., cells that incur in the largest errors when estimated), are retained. The storage taken by the model parameters and the retained cells, of course, should take a fraction of the space of the full cube and the estimation procedure should be faster than computing the data from the underlying relations. We use loglinear models to model the cube regions. Experiments show that the errors introduced in typical queries are small even when the description is substantially smaller than the full cube. The models also offer information about the underlying structure of the data modeled by them. Moreover, these models are relatively easy to update dynamically as data is added to the warehouse.

1 Introduction

A *data cube* is a popular organization for summary data [12]. A cube is simply a multidimensional structure that contains at each point an aggregate value, i.e., the result of applying an aggregate function to an underlying relation. For instance, a cube can summarize sales data for a corporation, with dimensions "time of sale," "location of sale" and "product type".

A lot of work on building datacubes efficiently has been done in the recent past [16,2,21]. However, precomputation of the entire cube can take a lot of space. Consider for example a retail sales dataset with dimensions *day*, *store* and *product*. If we assume 1,000 stores, 10 years (3,650 days) and 20,000 products, materializing just the core cuboid requires storage for 73 billion aggregate values.

⋆ This work has been supported by NSF grant IIS-9732113

(The core cuboid is the finest level of aggregation in the datacube, with cells defined for each combination of all the dimensions in the data: in our example, each day, store and product.) While in practice, only a fraction of the cells will be populated, the core cuboid can still be large (e.g., even 2 % of 73 billion amounts to 1.46 billion cells). Moreover, a datacube is defined as a lattice of aggregation levels (see [12,13]), with an exponential number (in the number of dimensions) of cuboids. For instance, in our example, a cuboid can be formed with aggregations of the dimensions days and stores, where every cell contains sales per day and per store, for all the products; notice that this cuboid can be computed from the core cuboid. In Relational On-Line Analytical Processing (ROLAP), the computation of the cube cells is deferred until users examine them. (In real products some of the cuboids are pre-computed and stored in relational tables.) When the entries are needed, the system queries the underlying database to compute them. Sometimes, a hybrid strategy is used in which part of the cube is materialized (e.g., the base data) and the rest is computed on demand. These techniques however, can impose long delays in answering queries.

These limitations have prompted researchers to look for techniques to compress the datacube in such a way that only a fraction of the space is needed [8,7,1,18,20]. Since the compression techniques are lossy, one can only provide approximate answers to the queries posed to the datacube. On the other hand, the queries can be answered without incurring into much disk I/O, so the response time is considerably smaller than the one experienced in uncompressed datacubes.

In this paper we present a technique to compress datacubes based on loglinear models [4]. Loglinear models are a form of statistical parametric models, i.e., models that attempt to estimate the data points using a function composed by a series of parameters. (We have preliminary explored a simpler parametric technique based on linear regression in [8,7].) The technique uses loglinear models to characterize dense chunks of the core cuboid in the datacube. These models, can be used to estimate cells, with a certain degree of accuracy. We keep the errors caused by the estimation process under control by storing, along with the model parameters, those cells whose estimated values differ from the real values by more than a pre-established threshold. By doing this, we maintain a fixed bound for queries over the core cuboid, providing guarantees for the approximated answers. Moreover, the idea can be easily extended to any other cuboid (without having to recompute models for any of these cuboids): by maintaining a small number of cells in each level of aggregation, or cuboid, we can have guaranteed bounds for approximate answers over the entire datacube. This, is an alternative to techniques that decide which parts of the cube should be materialized to obtain a good tradeoff between space and query performance, such as the heuristics presented in [13]. In our case, with a very small investment (a few of the cuboids' cells) we can provide approximate answers for queries posed over all the datacube. Moreover, with coarser aggregations (cuboids with less dimensions, such as $day, product$), we can guarantee tighter bounds for the answers, simply because the errors committed by the estimation of individual cells tend to cancel each other.

We call the structure that results from storing the models and the retained cells a *Quasi-Cube*. It is important to emphasize that, in a Quasi-Cube, the error bound can be kept at a desired level, **independently** of the distribution of the data. When answering queries, the system can use the models and the retained cells to give an answer with a guaranteed maximum error level attached to it.

Parametric methods to compress datacubes have an advantage over other techniques (such as the ones described in [1,20]): the parameters computed describe the data accurately and can serve as a good basis to mine important conclusions about the underlying distribution of data. The structure of the model describes the patterns of interaction [4]. Moreover, one can immediately know which dimension (or combinations of dimensions) exert the biggest influence in the data by looking at the relative size of the model parameters [4]. By comparison, it is difficult to draw information from a histogram (as used in [1]) or a wavelet decomposition (as used in [20]). Although the approach used in [18] offers similar advantages to our technique by obtaining a kernel that explains the distribution of the data, it is well-known that kernel estimations are very inefficient as the number of dimensions grow [19]. This is true because truncating the tails of the distributions can have an enormous effect on the errors obtained. (In other words, in moderate- to high-dimensional cases, regions of relative low density can still be very important parts of the distribution.) Moreover, in [18], the authors decide to retain only the "outliers" that fit in a memory buffer, making the accuracy of the method depend on the data, rather than on the method itself.

It is also important to stress that the providing compression of the whole datacube, while guaranteeing error bounds for **every query**, regardless of the aggregation is not an easy task for the compression schemes proposed in [20,1,18], since they aim to compress a part of the datacube, i.e., the core cuboid. Their answers will, indeed have tighter bounds as the aggregations grow coarser, but their methods can only tell us that the errors are less than the ones committed in the queries over the core cuboid, without being able to establish any tighter bound. This is due to the fact that none of the other techniques takes the approach of retaining outliers to control the error bounds.

There are other disadvantages to published methods. In the wavelet decomposition method advocated by [20] it is not clear how resilient the method is to missing cells, a common occurrence in datacubes. In the histogram-based method of [1], a set of precomputed samples of a small set of distinguished joins, or join synopses, is used to estimate the answers to the queries. A criticism to this technique is that these samples must be frequently recomputed to avoid the bias introduced by answering all queries using the same set of restricted data. Again, in the way that both methods, wavelet-decomposition and join synopses, are implemented, the accuracy of the method depends on the data distribution but not on the method itself.

This paper is organized as follows. In Section 2, we present our technique in detail, including algorithms to divide the cube in regions (chunks) and to model the data in them. Section 3 presents the experimental results over two datasets: a real dataset and a synthetic one (the later used to demonstrate scalability). Finally Section 4 presents the conclusions and future work.

2 Our Method

In this section we describe in detail how we compress the datacube by means of constructing loglinear models for dense regions of the cube. We model regions of the core cuboid (the core cuboid is the set of cells of finest granularity; any other cuboid in the datacube can be computed from the core cuboid by aggregating on one or more of the dimensions) and employ these models to estimate the values of the individual cells. The reason to focus in the core cuboid is simple: the error guarantees for queries to the core cuboid hold for any other cuboid in the lattice. (In reality, as we shall see, the errors when we aggregate cells of the core cuboid decrease dramatically.)

As we stated in the introduction, to avoid incurring in large errors by the estimation, we retain all the cell values whose estimations are farther away from the real value by more than a pre-established threshold. This threshold becomes the guarantee of the approximate answer has. (As we will show later, many answers are, in reality closer to the real answer than what the threshold predicts.) We store the model parameters (for each modeled region of the cuboid) along with the retained cells to process the queries.

In the rest of this section, we describe in detail how we solved each of the issues involved in compressing the cube and processing queries.

2.1 Dividing the Core Cuboid

In order to select the chunks in which we divide the core cuboid we use a density based approach called hierarchical clustering [14] to get high density portions of the cube. This approach has been previously utilized to identify regions of high density in multidimensional data ([3]). (The aim in [3] is to have a density approach to clustering, where a cluster is defined as a region with higher density of points than its surrounding regions.) We assume that the core cuboid has been computed (an algorithm such as the one presented in [16] is well suited for the task).

As we pointed out in Section 2, we need to find “chunks” of the core cuboid that are dense enough to be modeled properly: if a region is too sparse, the models will not fit well. Moreover, if we try to model a chunk that is too sparse, the “holes” in the chunk, i.e., the cells that do not have any values, will be too many. Since we need to list these holes, it is very likely that the compression gains achieved by the model would be offset by the size of the hole list. So, our procedure is to hierarchically explore levels of chunks until we find regions that are dense enough for modeling. Notice that doing this also makes our technique scale well with large datacubes: regardless of the size of the core cuboid, we aim to achieve small descriptions of its dense regions, thereby compressing the data in those regions by describing it via a few model parameters and a list of outliers and holes. For big cuboids, we will have more dense regions, but since each one of them will be compressed, the overall compression rate will be good. Moreover, we never have to deal with having to fit a model for too large a region (which would increase the computation time for the fitting processs), since the chunks are smaller than the entire cuboid.

There are many ways to partition the core cuboid. We take a simple approach described in what follows. Initially, we partition the space of the cuboid into non-overlapping rectangular 1st-level chunks which have the same chunk size. At any point during the process of partitioning we may decide to further divide a 1-st level chunk into several 2nd level chunks, and successively an j-th level chunk into $j+1$-th level chunks (with the maximum level being a predefined constant $MAXLEVEL$).

There are three parameters used to drive the partitioning process: the minimum acceptable density for chunks (number of cells divided by chunk size), the maximum error tolerated in the estimation process (β), and the maximum percentage of outlier cells allowed in the chunk. At every step of the partitioning process we maintain a description list in memory which contains information about every chunk. Chunks in this description can be of three types: null chunks contain no data, sparse chunks which do not have enough cells to be compressed and are kept as a list of the cells they contain, and modeled chunks which are dense and have been modeled.

The actual details of the algorithm for partitioning the cube in chunks are not given in this paper, for reasons of space. They can be found in [9]. Special mentioning should be made of the treatment of holes for modeled chunks. Since it is assumed that the number of holes in this type of chunk is small, we can simply treat them as outliers and keep them in the outliers list. Otherwise, we need a separate index to indicate which cells are non-zero. This has repercussions in our method: the smaller the chunk descriptions are, the more of them we will be able to keep in memory, thereby avoiding the need for fetching them from the disk.

2.2 Modeling the Data Chunks

Notice that many choices of models are possible (a survey of methods can be found in [6]), since the general technique is quite independent of the model chosen for the chunks. Of course, there will be some models that are better suited for specific classes of data and therefore will produce smaller estimation errors. (We have studied simple linear regression models and their effect in the errors in [7].) In this paper we choose to study compression using loglinear models [4,5]. Loglinear models are known to provide a good fit for multinomial distributions.

The loglinear model for an n-dimensional table is:

$$\hat{l}_{i1i2\cdots in} = \log \hat{y}_{i1i2\cdots in} = \sum_{G \subset d_1, d_2, \cdots d_n} \gamma^G_{(i_r | d_r \in G)} \tag{1}$$

A *saturated* model is one that contains all the possible k-factor effects,for k ranging from the number of dimensions to 1. The saturated loglinear model for 4 dimensions is given by Equation 2

$$\begin{aligned}\log \hat{y}_{ijkl} = \gamma + \gamma^A_i + \gamma^B_j + \gamma^C_k + \gamma^D_l + \gamma^{AB}_{ij} + \gamma^{AC}_{ik} + \gamma^{AD}_{il} + \gamma^{BC}_{jk} + \gamma^{BD}_{jl} + \\ \gamma^{CD}_{kl} + \gamma^{ABC}_{ijk} + \gamma^{ABD}_{ijl} + \gamma^{ACD}_{ikl} + \gamma^{BCD}_{jkl} + \gamma^{ABCD}_{ijkl}\end{aligned} \tag{2}$$

There are two approaches to estimate the model coefficients. One is the iterative proportional fitting method, based on solving the corresponding likelihood equations. This method can always get the precise solutions, but it needs many iterative steps over the data. The other method is to compute the coefficients from the values directly. The coefficients corresponding to any group-by G are obtained by subtracting from the average l value at group-by G all the coefficients from higher level group-by-s. The authors of [17] present fast computation techniques that make this approach feasible for large sets, and although this method does not give precise solutions, its faster running time makes it a better choice for compressing data chunks. For our case, we applied a modified version of their *UpDown* method, whose details can be found in [9].

It is obvious that a large number of models can be used to fit a given set of data points. For an n-dimensional loglinear model, there are a total of 2^{2^n} possible models (determined by which parameters of the saturated model are set to zero). Fingleton in [11] presents some possible strategies of model selection. A suitable criterion for choosing a model is to minimize the chi-squared or the associated likelihood-ratio statistic values. However, just minimizing these statistics can lead us to choose the saturated model in view of its perfect fit. For data reduction, we are more interested in concise models: i.e., in choosing the simplest model that is not inconsistent with the dataset. Such a model is easier to interpret, identifies the essential relations among variables and more to our point, gives us a good compression ratio. Using brute-force to compare among models can easily get out of hand; therefore, we must resort to a good strategy that gets an acceptable model using as few as possible model comparisons. In selecting a good model, we have two goals: first, achieve a good level of data reduction (this translates to using as few parameters having as few some outliers as possible to approximate the cube) and to have a meaningful model from which knowledge about this portion of the datacube can be extracted (this implies a good fit, but also a model simple enough to understand; it is for instance, hard to interpret combination effects of more than three attributes).

We have chosen to select our starting models from a set of three types. First, we consider the model without any dimensional effects (only the factor representing the average is present). Secondly, we consider the complete independence model which only contains main-effect terms (not terms that correlate several dimensions). Thirdly, we consider a model in which all the possible k-factor effects are present with $2 \leq k \leq$ MAXLVL, with MAXLVL is a pre-established value less than or equal to n. From all these models we select a start model which has the best compression ratio: that is, the fewest parameters and outliers. This selection is by enumeration: we compute the parameters and the outliers for each model and select the best among them. We then try to improve on this model by using a perturbation heuristic, which utilizes backward elimination and forward selection of factors and whose details can be found in [9]. According to our experience, although we may not get the optimal model following this heuristic, we do get a very good approximation. The advantage of this heuristic is that we only need to perform a number of passes over the data we are modeling equal to MAXLVL plus the number of forward tries, plus the number of backward tries. It is important to notice that we are modeling the subset of data contained in a chunk, which is much smaller than the entire core cuboid and usually fits in memory. Notice that if this is the case for every chunk, then we only need to bring each cell in the cuboid once to memory.

2.3 Querying the Approximate Cube

A range query over the cuboid can be decomposed as the union of several disjoint queries, each spanning a chunk in the cuboid. That being the case, for each one of the disjoint sub-queries there are two possibilities:

– The sub-query completely includes its respective chunk. In this case, the answer of the sub-query can be immediately obtained from the chunk description, which contains the aggregated value of all the cells in the chunk. Notice that this value is free of error and that the answer to this sub-query does not require retrieving any of the values in the retained list of cells for the chunk or estimating any of the cells values.
– The sub-query covers the respective chunk only partially. In this case, we need to estimate or retrieve (from the list of retained cells) the individual cell values and aggregate them.

In larger queries, it is likely that many sub-queries will span complete chunks and therefore, the running time will be sped up considerably.

In closing this section, it is important to remark that updating the models incrementally, as new data arrives to the warehouse is an easy task. The details of this issue are presented in [9].

3 Experimental Results

In this section we show the results of experimenting with one dataset. Additional experiments are reported in [9]. The experiments were conducted in a SUN Ultra 2, with two processors, and 500 Mbytes of RAM.

The dataset used is taken from the U.S Census Bureau data [10]. The dataset contains population broken by country, year, age group and sex. We use C, Y, A

and S respectively, to denote these dimensions. These dimensions have domains of size 227,23,17, and 2 respectively. There is a total of 169,694 cells and 7,820 missing cells in the space of the core cuboid.

Figure 1 shows the data compression achieved under different error levels (β). The table shows the number of cells in the core cuboid, cells that are retained as belonging to sparse chunks (Single Cells), number of parameters used by the models (Parameters), and number of outliers (Outliers). The compression ratio indicates the ratio between the space needed to store the parameters, single cells, outliers and data structures used to describe the chunks, and the space needed to store the cells in the original cube. Notice that we start achieving high compression rates (33 % and under) only when the individual cells error rate is left to be higher than 15 %. For an individual cell error of 40 % we achieve a value of 14 % compression rate. Although, at first glance this may not seem very impressive, one has to remember that very rarely in OLAP the analysts care about single cell values. Rather, what is usually needed is higher level aggregations, such as range queries or other cuboids in the lattice. To show our performance in those aggregations, Figure 2 shows the average error in the answers obtained by range queries as a function of the selectivity of the query (shown as a fraction of the number of cells in the core cuboid). The range queries were run over the approximate core cuboid, compressed with a $\beta = 0.4$. These range queries compute the aggregation of all the cells in their ranges. As it can be seen from the results, none of the answers exhibits an error that goes beyond 3 %, in spite of the 40 % error that the individual cells in the cuboid may reach.

β	Cells	Single cells	Parameters	Outliers	Compression ratio
0.05	169694	29716	15574	87011	0.781
0.10	169694	0	20035	64348	0.499
0.15	169694	0	20055	36365	0.334
0.20	169694	0	20075	20616	0.241
0.25	169694	0	19995	12354	0.192
0.30	169694	0	19921	8033	0.166
0.35	169694	0	19791	5521	0.150
0.40	169694	0	19323	4257	0.140

Figure 1: Compression obtained using the census dataset with different β. The column “Cells” indicates the number of cells in the core cuboid; “Single cells” shows the number of cells retained in sparse chunks; “Parameters” the number of parameters used by all the models; and “Outliers” the number of cells retained in dense chunks.

Figure 3 shows the results of running queries to compute each one of the cuboids in the lattice of the datacube (with the exception, of course, of the core cuboid, which we have in the compressed form). The queries are run over the

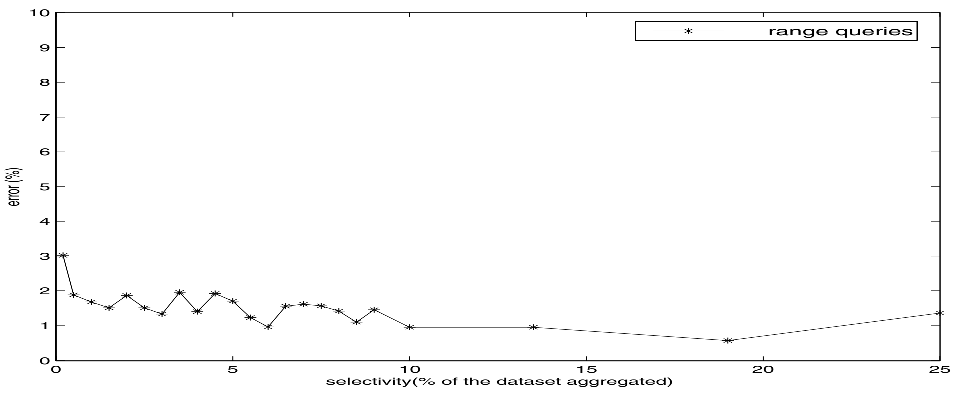


Figure 2: Error of range queries over the census dataset, computed from the approximate base cuboid compressed with $\beta = 0.4$. (The errors reported are the average over range queries of the same selectivity.)

compact representation of the core cuboid. In the table results are given for the minimum error over all the cells (minerr), the maximum error (maxerr), the average error (avgerr), the size of the cuboid in cells (size), and a histogram that shows the number of cells that fall between 0 and 10% of error (0-0.1), more than 10% but less than or equal to 20% of error (0.1 - 0.2), more than 20% but less than or equal to 30% (0.2 -0.3) and more than 30% but less than or equal to 40% of error (0.3-0.4). As shown, as we aggregate more and more cells to compute the cuboids that are in the lower part of the lattice, most of the errors fall in the first bin of the histogram and both the average and maximum errors decrease rapidly. (Keep in mind that the individual cells of the core cuboid were left to have estimation errors up to 40% of the real value.)

Figure 3 proves the point we made in the introduction: if one wants to keep errors in *any of the cells of the cuboids in the lattice* (with the exception of the core cuboid) under 10 %, it would be enough to retain an extra 29,119 values (the aggregated cells that exhibit errors bigger than 10% in all the other cuboids). Notice that this 29,119 are a fraction (24.6 %) of the 117,973 cells contained in all the cuboids other than the core one. In fact, the compression rate for the overall cube would be 18.3 %, still a very good ratio. By doing this, we can guarantee a fixed level of error (10 %) for queries in the other cuboids of the lattice. Queries over the core cuboid would get a guaranteed error of 40 %, but actual errors would likely be much less than that for most queries. If one would insist in making the 10 % guarantee across all cuboids, including the core, we would need to store 20,035 parameters, and 64,348 outliers (as shown in Figure 1), along with the 29,119 cells for the other cuboids. This would bring the overall compression ratio to 40 %. By contrast, using the algorithm described in [13] to decide which cuboids would be the most efficient choices to materialize, assuming we want to materialize up to 3 cuboids, one would need to store 178,194 cells (those of the core cuboid, and the cuboids YAS and CAS), or 62 % of the datacube. Yet, doing this, queries over any other cuboid would require the aggregation of cells in one of the YACS (core), YAS or CAS cuboids (which would involve also additional I/O and CPU time).

cuboid	minerr	maxerr	avgerr	size	0 - 0.1	0.1 - 0.2	0.2 - 0.3	0.3 - 0.4
CYA	0	0.398	0.080	88756	57228	21448	5234	937
CYS	0	0.312	0.034	10442	9415	519	47	1
CAS	0	0.288	0.038	7718	6808	523	47	0
YAS	0	0.118	0.021	782	778	4	0	0
CY	0	0.289	0.029	5221	4767	201	23	0
CA	0	0.219	0.024	3859	3560	127	2	0
CS	0	0.110	0.021	454	431	3	0	0
YA	0	0.107	0.017	391	389	2	0	0
YS	0	0.029	0.012	46	46	0	0	0
AS	0	0.053	0.017	34	34	0	0	0
C	0	0.102	0.013	227	216	1	0	0
Y	0.007	0.019	0.010	23	23	0	0	0
A	0	0.042	0.014	17	17	0	0	0
S	0.001	0.021	0.011	2	2	0	0	0
ALL	0.010	0.010	0.010	1	1	0	0	0

Figure 3: Errors for the cells of the lattice cuboids using the census dataset. The maximum error in the base cuboid is 40%. The columns 0-0.1, 0.1-0.2, 0.2-0.3 and 0.3-0.4 contain the number of cells in each of the cuboids whose estimated errors are within the indicated bounds.

Finally, we illustrate how the chunk models are useful in themselves. Examining the chunk which involves 7 countries (Pakistan, Russia, Great Britain, Indonesia, U.S., India and China), all the years, all the age groups , and both sexes, and examining the standard variances of the factors of the corresponding model, we discover that the most important attribute effect is C (country), the second is A (age group). For a given attribute, for instance country, we know that the biggest parameter value (γ^C) is 1.384, corresponding to China. (Without looking at the data, we know that China is the country with highest population in this chunk.) We can also discover the ranking of population by looking at the ranking of parameters γ^C. Also, taking an effect such as γ_{CY}, we know that for a given country, the sum of these effects is equal to zero (due to the constraints that govern the parameters). That implies that we will have a biggest positive and a biggest negative effect γ_{CY} for each country. These defects tell us which years have the biggest influence (positive or negative) in population for that country. In sum, there is a wealth of information captured by the chunk model.

4 Conclusions

In this paper, we have presented an effective technique to compress datacubes, trading space for accuracy of the query answers. We have shown that in real datasets we can achieve a good compression ratio while still obtaining small errors in the query answers. It is important to remark that as the selectivity of the queries increases, the error in the answer decreases drastically.

Using some extra space, one can retain the aggregate values for cells in the less detailed cuboids on the lattice (coarser aggregations), guaranteeing maximum levels of error for queries on these cuboids (this are hard guarantees, e.g., the error in the query will be less than or equal than 10 %). (This was shown, as an example, in the results presented in Figure 3.) Doing this, one can effectively compete with techniques that choose which cuboids in the lattice are the most effective to materialize, based on space restrictions (such as the ones described in [13]). Those techniques would still have to incur in CPU and I/O time in trying to answer queries from other cuboids that are not materialized, while our technique would provide much faster response time at the expense of some errors in the answers (whose levels are guaranteed).

There are some aspects of this work that merit further research. Among them, we are trying to evaluate other heuristics for model selection, and other ways of pre-clustering the cells in the core cuboid before we proceed to divide it into chunks. We are currently experimenting with heuristics to reshufle the "rows" of each dimensions in order to obtain denser regions whose models provide better fitting (thereby decreasing the number of outliers to be retained). These heuristics, which are based on sampling, bring the added benefit of not having to pre-compute the core cuboid before we chunk it. In other words, the computation of the chunks cells and the chunk modeling can be done in the same step. It is worth to point out that since the heuristics depend on sampling, the reshufling process is linear on the number of datacube dimensions (d), and thus very practical.

Another aspect that merits further research is that of compressing cubes with many measures. (This is becoming a standard occurrence in industry; e.g., for a sales datacube, one might be interested in keeping measures of total sales in dollars, units sold, profit, and so on.) In the worst case, every measure would require its own set of models parameters and outliers. (Notice that the compression ratio would not suffer from this, since not compressing the cube would require to store values for each of the measures in non-null cells, thus increasing the total size of the datacube.) However, it is possible that some of the measures can be estimated or approximated from the approximations of others, and we are currently researching this issue.

Finally, we like to point out that our compressed versions of the core cuboid can serve as concise datamarts for remote clients of the warehouse. (Building smaller datamarts that can be exported to remote machines outside of the warehouse is becoming a standard practice in industry [15].) They are ideal to be used in less powerful machines because of their smaller size, and yet they are powerful enough to not only give approximate answers to queries in the client side, but also they contain a wealth of mining information in the chunk models.

References

1. S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join Synopses for Approximate Query Answering. In *Proceedings of the 1999 ACM-SIGMOD International Conference on Management of Data, Philadelphia, PA*, June 1999. 312, 313
2. S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the Computation of Multidimensional Aggregates. In *Proceedings of the 22nd International Conference on Very Large Data Bases, Bombay, India*, pages 506–521, September 1996. 311
3. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle*, June 1998. 314
4. A. Agresti. *An Introduction to Categorical Data Analysis*. John Wiley, New York, 1996. 312, 313, 315
5. E. B. Andersen. *Introduction to the Statistical Analysis of Categorical Data*. Springer Verlag, New York, 1997. 315
6. D. Barbará, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. G. Sevcik. The New Jersey Data Reduction Report. *Bulletin of the Technial Committee on Data Engineering*, 20(4):3–45, December 1997. 315
7. D. Barbará and M. Sullivan. Quasi-Cubes: A space-efficient way to support approximate multidimensional databases. Technical Report, Department of Information and Software Systems Engineering, George Mason University, 1997. 312, 315
8. D. Barbará and M. Sullivan. Quasi-cubes: Exploiting approximations in multidimensional databases. *SIGMOD Record*, 26(3), September 1997. 312
9. D. Barbará and X. Wu. Using loglinear models to compress datacubes. Technical Report, Department of Information and Software Systems Engineering, George Mason University, 1999. 315, 316, 317
10. U. S. Census Bureau. Population data. http://www.census.gov/main/www/access.html. 317
11. B. Fingleton. *Models of Category Counts*. Cambridge University Press, 1984. 316
12. J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. In *Proceedings of the International Conference on Data Engineering, New Orleans*, 1996. 311, 312
13. V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing Data Cubes Efficiently. In *Proceedings of the ACM-SIGMOD Conference, Montreal, Canada*, 1996. 312, 319, 321
14. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988. 314
15. MicroStrategy. DSS Server[tm] Features. http://www.microstrategy.com/products/ server/features.htm. 321
16. K. A. Ross and D. Srivastava. Fast Computation of Sparse Datacubes. In *Proceedings of the 23rd VLDB Conference, Athens, Greece*, 1997. 311, 314

17. S. Sarawagi, R. Agrawal, and N. Meggido. Discovery-driven Exploration of OLAP Data Cubes. In *Proceedings of the International Conference on Extending Data Base Technology*, pages 168–182, 1998. 316
18. J. Shanmugasundaram, U. Fayyad, and P. S. Bradley. Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. In *Proceedings of the ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA*, August 1999. 312, 313
19. B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, UK, 1994. 313
20. J. S. Vitter and M. Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *Proceedings of the 1999 ACM-SIGMOD International Conference on Management of Data, Philadelphia, PA*, June 1999. 312, 313
21. Y. Zhao, P. M. Deshpande, and J. F. Naughton. An Array-Based Algorithm for Simultaneous Multidimensional Aggregates. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data, Tucson, Arizona*, pages 159–170, May 1997. 311

Materialized View Selection in a Data Warehouse

Feng Yu   Du Xiaoyong   Wang Shan

School of Information, Renmin University of China, 100872
{fengyu, duyong, swang}@mail.ruc.edu.cn

Abstract. Selecting views to be materialized is one of the most important decisions in designing a warehouse. In this paper we present algorithms for how to select a set of views to be materialized so as to achieve both good query performance and low view maintain cost under a storage space constraint. We generated the candidate views based on Query Frames which represent the essential characters of queries. We give an algorithm to select an optimal set of views to be materialized first and then give a feasible algorithm which generates a reasonable solution.

1 Introduction

Data warehouse provides a base for decision support system. Decision-support queries are usually very complex, so they may take very long time to complete. How to improve the query performance in the data warehouse is an important topic in the fields of database research recently.

Since queries may share some intermediate results, Materialized view, which is to precompute frequently-asked queries and store them for future use, is a useful technique for query answering in a data warehouse. But materialized views occupy the disk space and we must maintain the consistence of them, so we should select the views to materialize effectively [1, 2, 3, 4, 5, 6, 7, 8].

The rest of the paper is organized as follows: Section 2 provides an import concept—Query Frame. In section 3, we describe our algorithm of selecting the appropriate views to materialize. Related work is introduced in section 4. We conclude the features of our solution in section 5 .

2 Query Frame(QF)

In this paper, we focus on unnesting Join-Aggregate(JA) type of SQL queries since many other type of queries (including EXISTS, NOT EXISTS, ALL, ANY, etc) can be reduced to JA type queries and then transformed to one or more unnesting JA queries[9,10]. Hence join and aggregation are the essential operations, We use a concept -- Query Frame (QF) to represent them.

Definition 1 (Query Frame): Query Frame of a query Q is a triple QF (JT, JP, GA), where JT is a collection of tables involved in the query, JP is a collection of the join patterns in the query, GA is a collection of attributes which are in the GROUP BY clause of the query.

3 Algorithms for Selection of Materialized View

Given a set of queries Q and a quantity S (available storage space), the view-selection problem is to select a set of views M to materialize that minimize total query response time and the total maintenance cost under the constraint that the total space occupied by M is less than S. The frequency of executing every query in Q can be different. However, without the loss of generality, we suppose all queries have the same frequency for simplicity.

The process of selecting views can be described as follows:

1. Transform every query into unnesting JA type and generate Query Frame.
2. Find the candidates of materialized view.
3. Put the selection and project condition into candidate views.
4. Find the optimal set of views under the space constraint.

In this paper, we focus on the algorithms of step 2 and 4.

3.1 Generation of Candidate Views

Definition 2 (Candidate View): Let Q be a set of queries, QF be the Query Frames of Q ,a candidate view for Q is a query which can benefit the execution of a join or aggregation in QF.

Because we only focus on the join and aggregation in Query Frame, there are two kinds of views that can be used for candidates: Join View(JV) and Aggregation View(GV).

Definition 3 (Join View): Join View is a query which includes a join of some tables without Group-By clause. Given a set of queries, Join View(JV) can be represented as a triple(JT, JP, COUNT), where JT is the tables attending join operation, JP is the join patterns, COUNT is the number of queries which can share this JV.

A multi-table join can improve its performance by using the result of a join which involves the less tables and has the related join patterns.

Definition 4 (Candidate Join View): Let QF(JT, JP, GA) be a Query Frame, JV(JT', JP', COUNT) be a Join View, JV is a Candidate Join View of the QF if

1. JT' $\subseteq$ JT
2. JP' $\propto$ JP ,which means that JP' is a subset of JP and the elements in JP' are the relative join patterns with tables in JT'.

We can aggregate against the result of multi-table join or a base table. There is no difference between them for the aggregation. For simplicity in notation, we think the aggregation is done against a table in this paper.

Definition 5 (Aggregation View): Aggregation View is a query which includes a Group-By clause. Given a set of queries, Aggregation View(GV) can be represented as a triple (T, GA, COUNT), where T is a table on which aggregation was done, GA is a set of Group-By attributes, COUNT is the number of queries which can share GV.

Execution of an aggregation is not the same as that of a multi-table join, which can be divided into many suboperations. But the queries which aggregate on the same table can share the Aggregation Views.

We divide the set of queries into several groups. The queries in one group aggregate on the same table. We can generate candidate aggregate views for every group of queries respectively.

Given a set of Query Frame QF_1, QF_2, ... ,QF_n, the Query Frames in one group must have the same JT and JP.

Definition 6 (Candidate Aggregation View): Given a group of Query Frame QF(QF_1, QF_2, ... ,QF_n), where QF_i is a triple (JT, JP, GA_i), i=1...n. let GV(T, GA, COUNT) be an Aggregation View, GV is a Candidate Aggregation View with respect to QF, if:

1. T is the result of JV(JT, JP, COUNT)
2. $\exists\ GA_i$, $GA_i \subseteq GA$
3. $GA \subseteq \cup GA_i$, where $\cup GA_i$ is a collection of all the attributes in GA_i, i=1,...,n.

3.2 Algorithm for Generating Candidate Views

Algorithm1 and Algorithm2 generate candidate Join Views and candidate Aggregation Views respectively for a set of Query Frames. They count the number of Query Frames which can share it for every candidate view at the same time.

Algorithm 1:

1. VS(VS_1,..VS_i,...VS_k) denotes candidate join views sets, where VS_i contains the set of candidate join views which involve i tables. They are all initialized $\emptyset$
2. Generating a join view JV_i for every Query Frame QF_i, where JV_i.JT= QF_i.JT , JV_i.JP= QF_i.JP, JV_i.COUNT=1, i=1,2,..., n.
3. $\|JV_i\|$ denotes the number of the tables that involved in JV_i. For every JV_i:
   ①S= $\|JV_i\|$; Put it into VS_S;
   ②Generate all the candidate views V_1,V_2,......V_m for the join view JV_i according to the definition of Candidate Join View.
   ③Put V_1,V_2,......V_m into the appropriate VS_1, ... VS_{S-1}。
   /*If V_i involves K tables, then put it into VS_K, and if V_i has already existed in VS_K, then V_i.COUNT++, else put V_i into VS_k and set V_i COUNT=1;*/
4. Output the candidate join views in VS_1,VS_2,......VS_i,......VS_k

Algorithm 2:

箪 GS(GS_1,...GS_i,...GS_L) denotes candidate aggregation views, where GS_i contains the set of candidate aggregate views in a group. They are all initialized $\emptyset$硌

2. Generating an aggregation view GV_i for every Query Frame QF_i whose GA is not null, where GV_i.GA= QF_i.GA , GV_i.T= JV(QF_i.JT, QF_i.JP 0), GV_i.FLAG=1, i=1,2,..., n.

3. Divide the aggregation views GV_i into several groups. The aggregation views in one group aggregate on the same table. Suppose there are L groups.
4. For(every group GS_i)
   ①Generate candidate views according to the definition of Candidate Aggregation View. Assume they are $V_1,V_2,......V_m$.
   /*For example, if there are four aggregation views in GS_i and their Group-By attributes are (A), (A,B), (A,C), (A,D) respectively, then the attributes of the generated candidate views are G1(A), G2(A,B), G3(A,C), G4(A,D), G5(A,B,C) , G6(A,B,D), G7(A,B,C,D)*/
   ②Put V_1、V_2、......V_m into GS_i and set V_i.COUNT to the correct value;
   /*G1.COUNT=1;G2.COUNT=2;G3.COUNT=2;G4.COUNT=2;G5.COUNT =3;G6.COUNT=3;G7.COUNT=4 */
5. Output the candidate views in $GS_1,GS_2,......GS_i,......GS_L$.

3.3 Cost Analysis

To find the optimal set of views $M(V_1,V_2,...,V_n)$ to materialize from the candidate views, we give two cost functions: cost for queries and benefit of a materialized view.

Cost for queries is composed of two parts: the query cost and the maintain cost. The total cost for queries with respect to M is given as follows:

$$C(Q,M) = C_Q(Q,M) + C_M(M) = \sum_{qi \in Q} C_{qi}(M) + \sum_{Vi \in M} C_u(V_i)$$

where $C_{qi}(M)$ is the cost to compute q_i with the set of materialized views M, and $C_u(V_i)$ is the cost to maintain the view $V_i(V_i \in M)$.

The benefit of a view consists of two parts: the profit of queries obtained from the view and the cost of maintaining this view.

We suppose that the profit of queries comes from that queries can use V by scanning the disk directly without calculating it as before, then the benefit of a query from the view $B(V_i,M)$ with respect to M can be given as:

$B(V_i,M)= C_Q(V_i,M) - C_R(V_i) - C_U(V_i)$

Where $C_Q(V_i,M)$ is the cost for calculating the view V_i, $C_R(V_i)$ is the cost for reading the view V_i and $C_U(V_i)$ is the cost for maintaining the view V_i.

If the number of queries which can share the view V_i is $N(V_i)$,then the total benefit of a view $BA(V_i,M)$ with respect to M is: $BA(V_i,M)= B(V_i)* N(V_i)$

3.4 Algorithms for Selection of Materialized Views

Now we need select a set of views M from a given set of candidate views $V(V_1,V_2,.....V_n)$ under the constrains of the size of available space S.

Some views in the set of candidate views are mutually exclusive, they shouldn't be selected at the same time. For the space limitations, the details can be found in [11].

3.4.1 A Naive Algorithm

Let $a_i = S(V_i)$, where $S(V_i)$ is the size of view V_i, then the problem of materialized views selection from n candidate views can be represented as the following mathematical model:

$$\min \ f = C(Q, M) = \sum_{qi \in Q} C_{qi}(M) + \sum_{Vi \in M} C_u(V_i)$$

$$\sum_{i=1}^{n} aiXi \le S, \quad Xi = 0 \square \ Xi = 1, \quad i = 1, 2 ..., n$$

Where M is the set of views which is selected to materialize.

The solution to the above 0-1 integer programming formulation gives an optimal set of views which will be materialized.

There are 2^n solutions for the selection of views from n candidate views. We can generate an optimal execution plan for every query q_i toward every solution M which meets the space constraint, then calculate the total cost for queries C(Q,M) , thus we can find an optimal set of materialized views.

This algorithm can obtain an optimal set of materialized views , but its complexity is near $O(2^n)$. It is unacceptable if there are many candidate views .Therefore, if we just need a reasonable solution, we can use the following algorithm.

3.4.2 A Feasible Solution

The main idea of the following algorithm is: the view which benefit the queries most should be materialized first under the constraint that the total space occupied by the views in M is less than available space.

Algorithm 3:

```
1. M denotes the set of selected views and S(M) denotes the size of all the views
   in M. M is inited Φ and S(M)=0.
2. While(S(M)<S )
   ①Calculate the benefit $BV_i$ of every view $V_i$ in V per unit space using the
     formula :$BV_i = BA(V_i,M)/ S(V_i)$
   ②Find the $V_i$ which has the maximum value of $BV_i$ .
   ③If($S(M)+S(V_i)< S$) Put $V_i$ into M and delete $V_i$ from V;
   ④Move out the views which are mutually exclusive in M.
3. Output M.
```

The complexity of this algorithm is $O(kn^2)$, where n is the number of candidate views and k is the number of stages used by the algorithm.

4 Related Work

The view-select problem has attracted substantial attention in the research community[1, 2, 3, 4, 5, 6, 7, 8]. Some of them discuss a specific model of a data warehouse—data cube[5, 6, 7, 8].

[5] provides a theoretical framework for view-selection problem in terms of the relational model and doesn't describe the feasibility and efficiency of algorithm.[6] considers materializing some views will reduce the total maintain cost.[7]regards a set of queries as AND-OR graph and presents several heuristic algorithm for selecting views, but they don't provide an evaluation of the algorithm in terms of the quality of solutions and it is difficult to provide an intuition as to why the heuristics work. The solution of view-selection problem in this paper is closest to the work in [8].

[8]uses MVPP(Multiple View Processing Plan) for a global execution plan for a set of queries in the data warehouse and select views based on MVPP. It provides two

algorithms to generate MVPP: heuristic algorithm and 0-1 integer programming algorithm. But some questions exist in the solution of this paper. First, it doesn't consider the constraint of storage space at the time of selecting views, second, the solution generates the optimal execution plan(MVPP) at first, then select the views, which doesn't consider that the existence of materialized views can cause the MVPP generated first is not optimal execution plan.

In this paper we use Query Frame(QF) to represent the essential feature of a query, find the candidate views based on Query Frame and then select views efficiently to materialize. The solution needn't generate an optimal global execution plan.

5 Conclude

In this paper we describe algorithms for how to select a set of views to be materialized so that the sum cost of processing a set of queries and maintaining the views is minimized under a storage space constraint. Our algorithms have the following advantages:

1. Our algorithms master the essential characters of queries
2. Our algorithms needn't generate a global optimal execution plan of multi-queries and consider the effect of materialized views on the queries directly.
3. Our algorithms are easily integrated into the existing RDBMS and take good use of the sophisticated optimization technique, so it is very practical.

Reference:

1. V. Harinarayan, A. Rajaraman , and J. D. Ullman. Implementing data cubes efficiently . In ACM SIGMOD,Canada, June 1996
2. E. Baralis, S.Paraboschi, E.Teniente. Materialized View Selection in a Multidimensional Database .In Proc. VLDB, 1997
3. A.Shukla, P.Deshpande, J. F.Naughton. Materialized View Selection for Multidimensional Datasets. In Proc. VLDB, 1998
4. H.Gupta, V.Harinarayan, A. Rajaraman , and J. D. Ullman. Index Selection for OLAP. In Proc. ICDE ,1997
5. D.Theodoratos, T. Sellis.Data Warehouse Configuration. In Proc. VLDB, 1997
6. K.A.Ross, D.Srivastava and S.Sudarshan Materialized view maintenance and integrity constraint checking: trading space for time. In ACM SIGMOD , 1996
7. H.Gupta. Selection of Views In a DataWarehouse. In Proc. ICDT ,1997
8. J.Yang, K.Karlapalem, Q.Li.Algorithms for Materialized View Design in Data Warehousing Environment. In Proc. VLDB, 1997
9. U.Dayal, Of Nests and Trees:A Unified Approach to Processing Queries That Contain Nested Subqueries, Aggregates, and Quantifiers. In Proc.VLDB,1987
10. M.Muralikrishna , Improved Unnesting Algorithms for Join Aggregate SQL Queries. In Proc. VLDB ,1992
11. Y.Feng, X.Y.Du, S.Wang, Materialized View Selection in a Data Warehouse., Renmin University of China, Research Report, 1999

ExSight: Highly Accurate Object Based Image Retrieval System Enhanced by Redundant Object Extraction

Kazuhiko Kushima[1], Hiroki Akama[1], Seiichi Kon'ya[1], and Masashi Yamamuro[1]

[1] NTT Cyber Space Systems Laboratories
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
{kushima, akama, konya, masashi}@dq.isl.ntt.co.jp

Abstract. This paper describes ExSight, a prototype system for content-based image retrieval that will provide image retrieval facilities based on the indexing of component objects. We present a database centric approach to image retrieval and other techniques necessary for successfully implementing ExSight. The essential point of this approach is automatic image data analysis, emphasizing automatic object extraction that implies redundancy. The database module of ExSight coordinates multiple space indices in order to obtain an overall ranking based on several different features. The experimental results reveal that object-based contents retrieval achieves a higher level of retrieval correctness than color-region based retrieval and the implemented multidimensional data access engine achieves real-time response.

1 Introduction

With the digitization of existing photographic resources and the spread of input devices such as digital cameras, hundreds of thousands of digital images are now being created and distributed via the Internet. It will be necessary to find a way to manage and retrieve desired images easily from such a large amount of low-cost image data. We developed a method that automatically extracts objects from regular photographs while allowing redundancy. Image features such as the color and shape are determined for extracted objects and stored in a database that contains multidimensional indices. Based on the developed method, we created ExSight, a prototype system for object-based highly accurate image retrieval, and evaluated the number of significant objects that can be extracted, the retrieval accuracy, and retrieval time.

2 Current Content Based Retrieval Techniques and their Problems

The Content Based Retrieval (CBR) technique is attracting attention [1] as an alternative to the conventional method, which involves manually attaching a text tag to each image in order to retrieve images using keywords. In CBR, the features of original images such as the color and texture are extracted and images with features similar to those of the key image are retrieved. Examples of image retrieval systems using CBR include Illustra&VIR [2], [3], which enable retrieval based on the overall color and pattern of an image, VisualSEEk [4], which focuses on the relative positions of different color regions in an image, and QBIC [5], [6] and NETRA [7], which perform retrieval based on objects in an image. Conventional methods, however, have the following problems:

– Illustra&VIR, which are based on the overall features of an image, are effective in retrieving landscapes and patterns, but they cannot perform more precise retrieval based on an object such as “a picture with a person in blue.”
– VisualSEEk, NETRA, and QBIC are object-oriented retrieval systems. Therefore, various techniques have been proposed to determine objects accurately without involving manual operation. To achieve automatic retrieval, a method is commonly used that divides an image into several regions primarily based on the color data. This enables accurate extraction of objects in an image with clear color distinctions such as in clip art. However, an extraction method for regular photographs has not yet been established. For example, QBIC allows users to assist extraction by defining contours and other elements of an object.

3 Goals of Content Based Retrieval Systems

As mentioned in [3], in the case of image data, one of the most important types of content-based similarity retrieval makes use of the component subimages, or more simply objects, within the image.

One typical approach to object-based retrieval is 'on the fly' template matching[8], where each image in the database is scanned repetitively in an attempt to find a matching part to the reference. However, such methods face a significant time performance problem when applied to very large databases.

Our approach, on the other hand, is a database-centric approach, which includes a data preprocessing phase similar to the case of keyword retrieval. Object extraction and the calculation of associated feature vectors are carried out in advance and the information obtained is stored in the database. This information is later used in query processing.

Normally, object identification is extremely human labor intensive in the same manner as assigning keywords to each image. Although several research projects have attempted to solve this problem[6],[7], all proposed methods still require some human interaction, or have problems in retrieval accuracy in case of applying to regular photographs such as snap shots.

The goals of the system can be summed up in the following:

– The system must be able to perform content-based retrieval for any image including snap shots taken by people.
– It must be able to retrieve objects (human faces, clothes, etc.) in an image.

– It must be able to manage a large number of digital images in a simple manner, and determine and attach indices to the objects in an image completely automatically.
– It must be able to perform real-time high-speed image retrieval.

4 Object-based Image Retrieval System

4.1 Process Flow

An essential part of our approach is to be able to carry out the necessary preprocessing completely automatically, saving significant human effort. The main difficulty for automatic object extraction has been that object recognition was not sufficiently advanced to be able to extract meaningful objects from complex images such as regular photographs. The major element of our approach to object-base retrieval is to extract as many object segments as possible from an image, including all potential meaningful objects as well as many meaningless redundant objects. All extracted objects are stored in the database[9]. Fig. 1 shows the processing flow.

The image features such as the color and shape of the extracted objects are represented as multidimensional vectors and stored in a database. In the proposed method, larger numbers of significant objects are extracted. Though it improves the retrieval accuracy, the retrieval time becomes longer. Instead of calculating distances for all objects, the proposed method enables high-speed selection of images using multidimensional indices where only the top *k* items are necessary as retrieval results.

4.2 Object Extraction by Varying Edge Magnitude Threshold

The conventional methods that determine non-overlapped color regions based on the color histogram cannot extract significant objects from regular photographs due to the following problems:

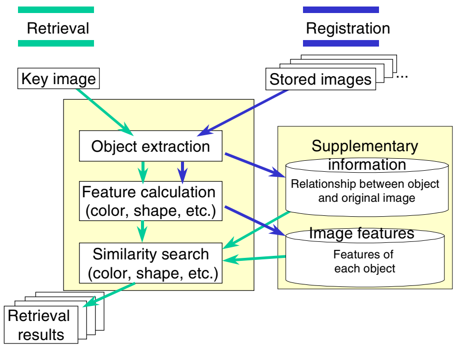

Retrieval
Registration
Key image
Stored images
Object extraction
Feature calculation (color, shape, etc.)
Similarity search (color, shape, etc.)
Supplementary information
Relationship between object and original image
Image features
Features of each object
Retrieval results

Fig. 1. Flow of Object Based Retrieval

- Excessive division of a single region occurs - An object is extracted as several different objects when multiple colors appear due to the level of light.
- Excessive integration of multiple regions - More than one object may be extracted as one object when there are similar objects next to one another.

To solve these problems, one way is to customize the edge threshold, so that meaningful object can be separated properly. However, this is not a generic approach because no general threshold exists for various kinds of images.

We developed an object extraction method that allows the extraction of different levels of objects by varying the edge magnitude threshold (hereinafter called the varying threshold method). The idea is to extract as many objects as possible from an image, including all potential meaningful objects as well as many meaningless redundant objects. The procedure is detailed hereafter (see Fig. 2).

Step 1) The differences in the HSI scale (Hue, Saturation and Intensity) between each pixel and eight adjacent pixels are calculated, and the results are considered as the multi-level edge magnitude of the corresponding pixel. This enables the creation of a shading edge image with the edge magnitude at each point as a gradation (see Fig. 2).
Step 2) With the effective edge threshold set to the gradation of 1, edge regions with the gradation of 0 are considered as potential objects. The following procedures are repeated until the edge threshold reaches the maximum gradation level.

1. The edge threshold is increased by one and the weakest edge regions are deleted.
2. Note the deleted edge regions. If a deleted edge region is adjacent to two or more potential objects (i.e., more than one potential object is integrated due to the deletion), potential objects before the integration are output as objects.
3. The deleted edge region is integrated into the adjacent potential objects and the integrated region is considered as a new potential object.

As shown in Fig. 2, when weaker edges are subsequently deleted from the stage at which the edges of all magnitudes are valid, divided regions become gradually

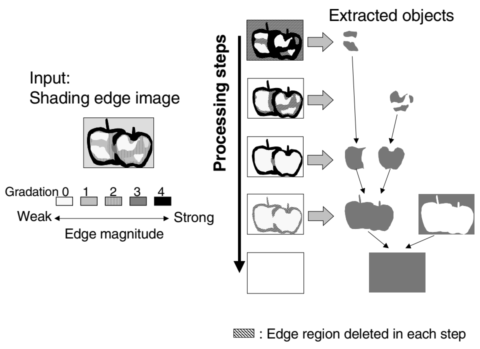


Fig. 2. Object Extraction by Varying Edge Magnitude Threshold

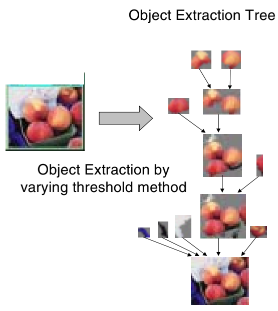


Fig. 3. Example of Object Extraction Tree

integrated. If two or more regions are integrated due to the deletion of an edge, individual regions before the integration are extracted as objects. This approach allows extraction of many potential objects as well as redundant ones. Fig. 3 shows an example of the object extraction tree that is created using the above method.

4.3 Defining object features

Features of an object are represented in the following feature vectors.

4.3.1 Color data

The HSI scale is used as color data. By adding the number of pixels in each of 256 gradations for H, S, and I, respectively, three 256-dimensional histograms with the number of pixels as the height can be obtained. If these 256 dimensions are used for a similarity search based on color, similar colors may be considered as different one in many cases. Therefore, each scale is reduced to 16 dimensions.

4.3.2 Shape data

In object-based retrieval, shape is an important factor in determining similarity. Shapes can be represented by the difference in distances from predefined representative figures. We adopted a method that uses a circumscribed circle as the only representative figure and represents the shape data as the distance from the circumscribed circle to the contour of each object. As shown in Fig. 4, we drew a circumscribed circle with the median point of the object as the center and calculated the distance from a point on the circumference to the point it first touches the object (called the circumscriptive distance) on a line toward the center. To be able to support rotation, the starting point was set at a point where the circumscriptive distance is the shortest and normalized to n-dimensions (24 dimensions in the prototype). Using this method, we were able to determine the shape data of extracted objects automatically.

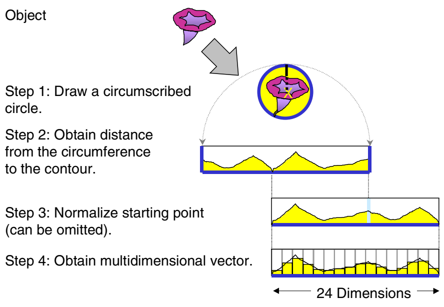

Object
Step 1: Draw a circumscribed circle.
Step 2: Obtain distance from the circumference to the contour.
Step 3: Normalize starting point (can be omitted).
Step 4: Obtain multidimensional vector.
24 Dimensions

Fig. 4. Representation of Contour as the Distance from the Outermost Circle

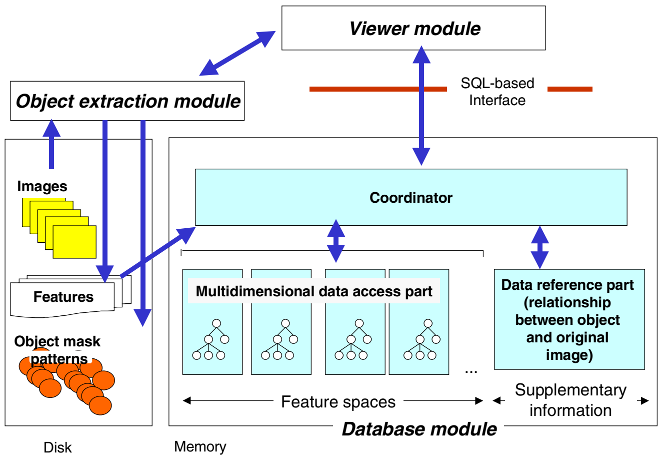


Fig. 5. System Configuration

4.3.3 Size and Position of Objects

The ability to retrieve objects of different sizes and positions is an advantage over a retrieval method based on color regions. The size of an object is represented in a unidimensional vector as the size relative to the original image and the position in a two-dimensional vector with the aspect ratio of the original image normalized.

All of these feature vectors are obtained automatically during the image data preprocessing phase. Individual feature, which represents H, S, I, shape, position and size respectively, is treated independently during the similarity calculation.

5 ExSight System Structure

We created a prototype image retrieval system, ExSight, according to the processing flow shown in Fig. 1. Fig. 5 shows the system configuration. ExSight is a client-server system comprising an object extraction module, a database module implemented on a server, and a viewer module on a client.

5.1 Object Extraction Module

From the input image data, objects are extracted by varying the edge magnitude threshold. Image features such as the color and shape are calculated for each object, and the relationship between the image and objects are stored in a database. These processes are conducted completely automatically.

5.2 Database Module

5.2.1 Data model and retrieval

The main entity types in our data model are image (parent) and object (child). The model includes links to represent the relation between each image and its children objects. An object's attribute includes a set of feature vectors. An image's attributes include title and the name of the author, etc.

The first step in object-based similarity retrieval is to obtain a set of similar objects for a given reference. This result set is found in terms of overall similarity. The overall similarity of an object is determined as a function of its similarity distance for each feature. Typically this function is a linear combination of the individual feature distances with each distance assigned a weight to reflect its importance in the overall distance calculation. Tuning the set of weights is important in order to reflect adequately the query intention of a user. In the second step of the retrieval, parent images are identified by tracking the links from the child object to the parent image.

5.2.2 High performance data access techniques

The database module comprises three parts. Those are the multidimensional data access part, which performs a similarity search based on features; the data reference part, which searches for metadata such as the relationship between an object and the original image; and the coordinator, which coordinates the former two parts (see Fig. 5). The multidimensional data access part holds six feature spaces in our implementation corresponding to six features described in Section 4.3, i.e. H, S, I, shape, position and size.

The reason to hold multiple feature spaces instead of merging them into single feature space is that it enables to reflect weight for each feature's similarity calculation according to user's intention. For example, if a user want to find objects with similar hue to the reference, then the H-feature space should be focused more strongly compared to another feature spaces.

The features of the key object and those of pre-stored objects are compared and the most similar *k* items are returned as the results. Retrieval of the top *k* items corresponds to obtaining *k*-nearest neighbors of the vector point that corresponds to the key in a multidimensional vector space. To avoid an exhaustive search, which calculates distances between a key object and all stored objects, multidimensional space indices have been proposed [10],[11],[12],[13]. Multidimensional data access part may contain any space indices. In the implementation, the VAMSplit R-tree [11] which achieves one of the fastest retrieval speed is selected.

The coordinator is responsible for executing searches on individual feature space and integrating individual results to form one result set ranked by overall similarity. The result returned by each individual feature space is a list of objects sorted by the distances from the reference.

Difficulties exist in determining how many results should be requested to each feature space. The result set returned by each individual feature space will be different and the ranking of images in the individual feature space is not the same as its ranking using overall distance measurement. For example, an image may be ranked below the top twenty in all individual vector spaces but be ranked inside the top twenty for the total distance. It is obvious that simply retrieving and combining the top k ranked objects from individual spaces to obtain the top k of overall similarity is not enough. Let k' be a number of results to be returned from a feature space to obtain the final top k. We want to choose k' such that it is larger than k, but as small as possible to achieve better performance. We addressed this problem in [14]. Based on the experimental result, *k*6* is adopted as k' that ensures approximately 90% of retrieval correctness.

Finally, the data reference part determines the original images to which the objects belong and returns them in an order of similarity.

5.3 Viewer Module

The viewer module is a GUI with functions such as designating key images, adjusting the weighting of features, and indicating retrieval results. It is implemented using a JAVA-applet.

6 Evaluation

6.1 Comparison with Color Clustering Method

QBIC, VisualSEEk, and NETRA divide an image into non-redundant color regions for image retrieval. VisualSEEk determines color regions using a back projection

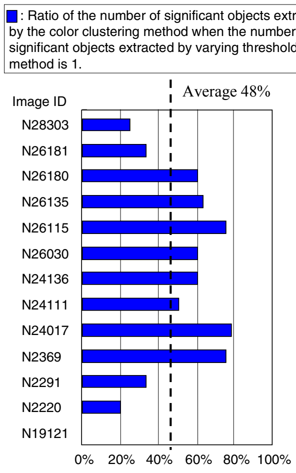


Fig. 6. Comparison of Extracted Object

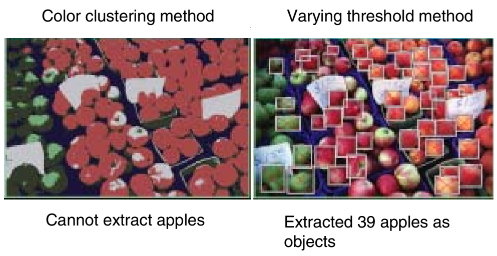


Fig. 7. Example of Extracted Object

based on predefined color sets [4]. QBIC determines color regions by clustering pixels according to the color and position [6]. NETRA proposes a boundary detection method called “edge flow”. It integrates the direction of change in color and texture, and forms the edge flow that indicates the closest image boundaries, i.e., regions of similar feature values. Finally, non-overlapped regions with opposite directions of edge flows are detected as objects [7]. However, how to extract meaningful objects more precisely is still under consideration. For evaluation purposes, color regions were determined in the following way, which is similar to the QBIC method (called the color clustering method).

1. All pixels in the original image are mapped to a five-dimensional space consisting of HSI values (three dimensions) and X and Y coordinates (two dimensions) to create clusters.
2. The original image is labeled based on the clustered pixels to determine color regions.
3. The features described in Section 4.3 are calculated for each color region.

6.2 Evaluation of Number of Extracted Objects

Using 13 different types of photographs from a PhotoDisc [15], the numbers of significant objects extracted by the two methods were evaluated. Fig. 6 shows a comparison of the two methods with the number of significant objects extracted by the varying threshold method as 1. Whether or not an extracted object is significant was determined by human observers.

The varying threshold method extracted twice as many significant objects as color

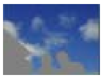
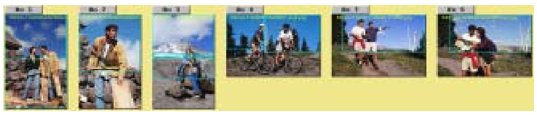
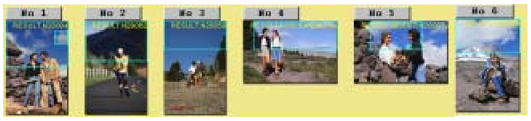
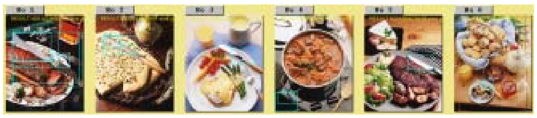
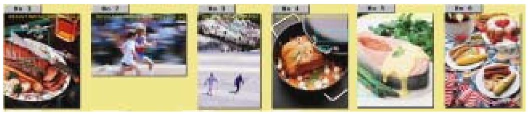
Retrieval key image	Retrieval results (top 6 items only)	
 Blue sky	Varying threshold method	
	Color clustering method	
Knife	Varying threshold method	
	Color clustering method	

Fig. 8. Example of Image Retrieval

clustering method on average. Fig. 7 shows extraction examples from an image with many similar colors in which the conventional color clustering method could not extract any objects. While excessive integration occurred in the color clustering method using a single threshold, the varying threshold method extracted many significant objects and proved to be effective for regular photographs.

6.3 Retrieval Accuracy

For 1,000 photographs randomly selected from the PhotoDisc [15], the retrieval accuracy was evaluated with the following five key images: a) blue sky and b) green lawn as large color regions and c) a red apple, d) a white plate, and e) a knife as arbitrary objects. Three examiners selected pictures that contain objects similar to the key images, and pictures selected by two or more people were regarded as a correct set.

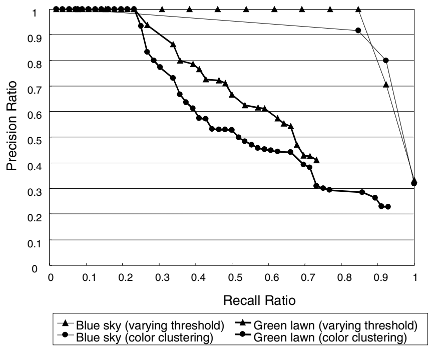


Fig. 9. Retrieval Correctness based on Big Color Region

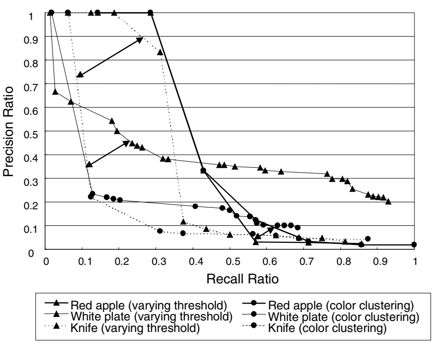


Fig. 10. Retrieval Correctness based on Arbitrary Object

Table 1. Retrieval Time Performance

Compared items \ Object extraction method		Color clustering method	Varying threshold method
Number of objects	Total number of Extracted objects	46,080	96,481
	Average per photograph	46	96
	Ratio when the number of objects extracted by color clustering method is 1	1 :	2.1
Retrieval time	Retrieval time (seconds)	0.69	0.82
	Ratio when retrieval time in color clustering method is 1	1 :	1.19

Measurements were performed using a Sun Ultra 30 in a search for the top 700 items from a database containing features with 75 dimensions. Display time is not included.

Two databases have been created separately using the color clustering method and the varying threshold method, respectively. Both databases adopt the same image features. Then retrieval was performed using 5 key images. The top *k* results were compared with the correct set and the precision and recall ratios were calculated. This procedure was repeated until the recall ratio gradually increased to 1.0, and the precision and recall ratios were plotted [16]. The weighting of features in each retrieval session was adjusted separately so that desirable images were returned in the top items.

Fig. 8 shows an example of the image retrieval results and Figs. 9 and 10 show the evaluation results. The retrieval accuracy was similar in both methods for large color regions such as “blue sky” and “green lawn” as shown in Fig. 9. On the other hand, for the retrieval of arbitrary objects, shown in Fig.10, the varying threshold method had an equal or better precision ratio and recall ratio. Especially for “knife” and “white plate,” the retrieval accuracy was low using the color clustering method because it could not divide objects from a similarly-colored background such as a tablecloth. The varying threshold method could extract both significant objects and those blended in with the background, and thereby showed a better level of retrieval accuracy.

6.4 Retrieval Time Performance

An increase in the number of objects to be retrieved results in a longer retrieval time. How the deterioration of the retrieval time performance can be limited using a multidimensional space index was measured. Table 1 shows the results. On average, the varying threshold and color clustering methods extracted 96 and 46 objects from one PhotoDisc photograph, respectively. Although the total number of objects was doubled in the varying threshold method, the retrieval time performance deteriorated by only about 20%. In a measurement using a Sun Ultra30 workstation, the retrieval time for 1,000 photographs (96,481 objects) was less than one second, which strongly supports the real-time retrieval capability.

7 Conclusion

We developed a highly accurate object-based image retrieval method based on arbitrary objects that allows redundant extraction of objects from regular photographs and stores the results in a database. We created a prototype and evaluated its retrieval accuracy using the subjective correct data. The results definitively show that the proposed varying threshold method can extract twice as many objects as the conventional color clustering method. The proposed method exhibited better retrieval accuracy when there are many objects that blend in with the background, which is often the case in regular photographs. The number of objects was doubled in the proposed extraction method because it allows redundancy, but resulting in a retrieval time performance deterioration of only 20%. With the proposed method, retrieval from about 100,000 objects was possible in less than a second. We will try to improve further the accuracy in object extraction and enhance the engine for handling tens of thousands of images.

Acknowledgment

The authors would like to express their gratitude to Dr. Masaji Kurakake of NTT DoCoMo who contributed many useful ideas to this study.

References

1. M. De Marsicoi et al.: Indexing pictorial documents by their content: a survey of current techniques, Image and Vision Computing, Vol. 15, pp. 119-141(1997).
2. A. Gupta: Visual Information Retrieval Technology A Virage Perspective, info@virage.com (1995).
3. A. Gupta and R. Jain: Visual Information Retrieval, Communications of the ACM Vol. 40, No. 5 (1997).
4. J.R. Smith and S. F. Chang: VisualSEEk: A Fully Automated Content-Based Image Query System," Proc. ACM International Conference on Multimedia, pp. 87-93 (1996).
5. M. Flickner et al.: Query by Image and Video Content: The QBIC System, IEEE Computer, Vol. 28, No. 9 (1995).
6. J. Ashley et al.: Automatic and Semi-automatic Methods for Image Annotation and Retrieval in QBIC, Proc. Storage and Retrieval for Image and Video Databases III, Vol. 2420, SPIE (1995).
7. W.Y. Ma and B.S. Manjunath: NETRA: A toolbox for navigating large image databases, IEEE International Conference on Image Processing, ICIP’97(1997).

8. V.V. Vinod, H. Murase, C. Hashizume: Focussed Color Intersection with Efficient Searching for the Object Detection and Image Retrieval, IEEE International Conf. on Multimedia Computing and Systems, ICMCS 96,pp.229-233 (1996).
9. M. Yamamuro, et al.: ExSight -Multimedia Information Retrieval System. 20th Annual Pacific Telecom. Conf., pp. 734-739 (1998).
10.A. Guttman: R-tree: A Dynamic Index Structure for Spatial Searching, Proc. of the ACM SIGMOD, pp. 44-57, Boston (1984).
11.D.A. White and R. Jain: Similarity Indexing: Algorithms and Performance, Proc. SPIE IV, Vol. 2670, pp. 62-75, San Jose (1994).
12.K. Curtis, N, Taniguchi, J. Nakagawa and M. Yamamuro: Similarity Indexing in High Dimensional Image Space, Proc. Of IPSJ SIGDPS, Japan, 82-18(1997).
13.N. Taniguchi and M. Yamamuro: Multiple Inverted Array Structure for Similar Image Retrieval, IEEE Multimedia System'98, pp.160-169 (1998).
14.K. Curtis, N. Taniguchi, J. Nakagawa, and M. Yamamuro: A Comprehensive Image Similarity Retrieval System that utilizes Multiple Feature Vectors in High Dimensional Space, International Conf. on Information, Communications and Signal processing, pp.180-184, ICICS'97(1997).
15.http://www.photodisc.com
16.D. K. Harman: Overview of the Third Text Retrieval Conference (TREC-3), Gaithersburg, MD 20899-0001, National Institute of Standard and Technology, Special Publication 500-255 (1995).

Applying Anaphora Resolution to Question Answering and Information Retrieval Systems

José L. Vicedo and Antonio Ferrández
{vicedo,antonio}@dlsi.ua.es

Grupo de investigación en Procesamiento del Lenguaje y Sistemas de Información
Departamento de Lenguajes y Sistemas Informáticos
Tel. +34 96-590-37-72. Fax +34 96-590-93-26
Universidad de Alicante
Campus de San Vicente del Raspeig
Apartado 99. 03080 Alicante, Spain

Abstract. Information Retrieval (IR) and Question Answering (QA) systems currently ignore information referred anaphorically in documents. Nevertheless, these references hide important information whose analysis can contribute to improve both IR and QA systems performance. The main aim of this paper is to analyze benefits of solving pronominal anaphora for IR and QA tasks. A complete system has been developed for evaluating the effects of applying this technique. Evaluation demonstrates that performance improvements are obtained in both tasks when pronominal references are solved.

1 Introduction

Information Retrieval systems (IR) have become an essential mean for accessing the huge amount of electronic information that is available today. But they usually accomplish only part of the whole task. When an IR engine retrieves a ranked list of documents, the process of locating the exact information required in the query is not concluded. Once the documents have been retrieved, users have to read completely several of these texts to locate exactly the information they need, which is often time-consuming.

As first approach to manage with this problem, application of Natural Language Processing techniques (NLP) was seen as a good way to improve IR precision rates. Nevertheless, continuous application of these techniques [12] [11] [13] over standard-based systems [10], did not improve gains proportionally to the research effort applied.

These approaches concluded first, that the use of documents as information units is often too large for satisfying many users precise information needs. And second, that shallow NLP techniques would probably be more effective applied for Question-Answering (QA) systems [7].

The so called QA systems are tools capable of extracting the answer to users queries directly from documents. Or at least, systems that extract text snippets from texts, from whose content it is possible to infer the answer to a specific question. In both cases, these systems try to reduce the amount of time users spend to locate a specific information.

The main aim of this paper is to analyze the importance of pronominal anaphora resolution over IR and QA systems. Our system resolves pronominal references in queries and documents. Later evaluation shows that resolution of these references improves IR precision and QA performance.

In the following section, the state-of-the-art of question answering systems will be summarized. Afterwards, importance of pronominal references in documents for IR and QA systems is analyzed. Next, our approach and system components will be described. Finally, actual evaluation results are presented and discussed.

2 Background

Interest in QA is very recent so, we have little information about systems that perform question-answering task. Nevertheless, we can classify actual approaches into two groups: sentence extraction systems and noun-phrase extraction systems.

Sentence extraction approaches are based on locating and extracting the most relevant sentences to the query by supposing that these sentences will probably contain the answer to the query. This approach is developed from different points of view in the following systems.

ExtrAns system [6] locates the phrases in a document from whose meaning it is possible to infer the answer to a specific question. This system suffers from several drawbacks: it uses expensive syntactic and semantic information and covers a very limited domain. At the moment, this system has only been able to process a small number of UNIX man pages.

The approach presented by Morton [9] is another example of sentence extraction system. This system models coreference relations between entities in query and documents. The coreference relations analyzed are identity, synonymy and part-whole. This system assigns different weights to terms depending on the obtained coreferences and as result, a ranked list of relevant sentences to the query is presented to users. In any case, pronominal references are not analyzed and its information is discarded.

The second group includes noun-phrase extraction systems. These approaches try to find a concrete information requested by questions performed in natural language whose answer is defined typically by a noun phrase.

MURAX is one of these systems [8]. It can use information from different sentences, paragraphs and even different documents to determine the answer to the question (the most relevant noun-phrase). It also uses information supported by interrogative terms to determine the type of answer the question needs (i.e. a place, a person's name, etc.). Another contribution of MURAX is the use of a lexico-syntactic pattern recognition module to help the system analyze different types of phrases.

None of the above described systems apply any process to resolve pronominal anaphora references. They do not take into account the information referred pronominally in documents. Simply, it is ignored.

The same situation remains when talking about IR systems. Due to the limited improvements achieved when applying other NLP techniques to IR systems, pronominal resolution approaches have not even been attempted. As result, actual IR systems include pronouns into the set of uncontent words that are refused for indexing (stop-word list).

With our system we want to determine the benefits of applying pronominal anaphora resolution techniques to IR and QA systems. Therefore, we apply the developed computational system, Slot Unification Parser for Anaphora resolution (SUPAR) [5] over documents and queries. SUPAR's architecture consists of three independent modules: lexical analysis, syntactic analysis, and a resolution module for natural language processing problems, such as anaphora.

For evaluation, a standard based IR system and a sentence-extraction QA system have been implemented. Both are based on Salton approach [10]. These systems process document collections with and without pronominal references solved in order to compare final performance.

As results will show, pronominal anaphora resolution improves performance of IR and QA systems, although the main benefits will be produced when applied for QA tasks.

3 Importance of pronominal information in documents

Trying to measure the importance of the information referred pronominally in documents, we have studied several public text collections used frequently for IR system testing. These collections are the following: TIME, CRANFIELD, CISI, CACM, MED and LISA. This analysis consists of determining the number of pronouns used in each of them. As average measure of pronouns used in a collection, we use the ratio between the quantity of pronouns used and the total number of sentences in each text collection. This measure approximates the level of information actual IR and QA systems ignore. Figure 1 shows the results obtained in this analysis.

As we can see, the amount and type of pronouns used in analyzed collections vary depending on the subject the documents talk about. TIME collection is composed from news published in Time newspaper. The ratio of pronominal reference used in this kind of documents is very high, a 57,24%. These documents contain a great number of pronominal references in third person (he, she, they, his, her, their) whose antecedents are people's names. In this type of documents, pronominal anaphora resolution seems to be very necessary for a correct modeling of relations between entities. A similar ratio level is presented in CISI collection. Extracts and general comments about document managing, classification, indexing, etc compose this set of documents. Although the ratio presented by this collection (48,68%) is also very high, "it" and "its" pronouns form the most important group of pronominal references used in this collection. In this

TEXT COLLECTION	TIME	CRANFIELD	CISI	CACM	MED	LISA
HE, SHE, THEY	2.148 (27,28%)	69 (6,20%)	359 (14,82%)	112 (8,73%)	147 (13,69%)	555 (12,22%)
IT, ITS	1.546 (19,63%)	877 (78,80%)	1.209 (49,92%)	849 (66,17%)	534 (49,72%)	2.076 (45,72%)
HIS, HER, THEIR	2.354 (29,90%)	101 (9,07%)	530 (21,88%)	195 (15,20%)	194 (18,06%)	1.319 (29,05%)
HIM, THEM	718 (9,12%)	15 (1,35%)	101 (4,17%)	36 (2,81%)	44 (4,10%)	234 (5,15%)
HIMSELF,ITSELF, THEMSELVES	221 (2,81%)	15 (1,35%)	48 (1,98%)	30 (2,34%)	14 (1,30%)	75 (1,65%)
WHO, WHOSE, WHOM	887 (11,26%)	36 (3,23%)	175 (7,23%)	61 (4,75%)	141 (13,13%)	282 (6,21%)
Total of pronouns	**7.874**	**1.113**	**2.422**	**1.283**	**1.074**	**4.541**
Number of Sentences	13.757	11.200	4.875	7.916	5.602	14.447
Ratio N.Pronouns / N.Sentences	**57,24%**	**9,94%**	**49,68%**	**16,21%**	**19,17%**	**31,43%**

Fig. 1. *Pronominal references in text collections*

case, antecedents of these pronominal references are mainly concepts represented typically by noun phrases. It seems again important solving these references for a correct modeling of relations between concepts expressed by noun-phrases. The lowest ratio results are presented by CRANFIELD collection with a 9,94%. The reason of this level of pronominal use is due to document contents. This collection is composed by extracts of very high technical subjects. Between the described percentages we find the CACM, MED and LISA collections. These collections are formed by abstracts and documents extracted from de CACM journal, from medical journals and from Library and Information Science Abstracts, respectively. We can notice that as more technical document contents are, the number of pronominal references used decreases.

After analyzing these results an important question may arise. Is it worth enough to solve pronominal references in documents? It would seem reasonable to think that resolution of pronominal anaphora would only be accomplished when the ratio of pronominal occurrence exceeds a minimum level. However, we have to take into account that the cost of solving these references is proportional to the number of pronouns analyzed and consequently, proportional to the amount of information a system will ignore if these references are not solved.

As results above state, it seems reasonable to solve pronominal references in queries and documents for IR and QA tasks. At least, when the ratio of pronouns used in documents recommend it. Anyway, evaluation and later analysis (section 4.5) contribute with empirical data to conclude that applying pronominal anaphora resolution techniques improve precision in IR and QA systems.

4 Our Approach

4.1 Overview

The main objective of this paper is to analyze if pronominal anaphora resolution can contribute to improve IR and QA systems performance. Our system is composed of three sub-modules. The first one will manage with anaphora resolution in both, queries and documents. For this purpose we use SUPAR computational system. This approach is described later (section 4.3). The second one is a standard IR that is combined with SUPAR module for anaphora resolution. The third one is a sentence-extraction QA system that interacts with SUPAR module too (section 4.4).

For the purpose of evaluation a baseline IR system has been implemented. This baseline is based on the standard information retrieval approach to document ranking described in [10]. For QA task, the same baseline is applied but using sentences as text unit. The TIME corpus has been selected as test collection due to its high level of pronominal references. Each term in the query and documents is assigned an inverse document frequency (*idf*) score based on the same corpus. This measure is computed as:

$$idf(t) = log(\frac{N}{df(t)}) \tag{1}$$

where N is the total number of documents in the collection and *df(t)* is the number of documents which contain term *t*. Query expansion consists of stemming terms using a version of Porter's stemmer. Document and sentence similarity to the query was computed using the cosine similarity measure.

4.2 SUPAR module

In this section, the NLP SUPAR modules are described lightly. The developed computational system, Slot Unification Parser for Anaphora resolution (SUPAR) [3], is described graphically in Figure 2. Its name is derived from the grammatical formalism used to store syntactic knowledge that is called Slot Unification Grammar.

SUPAR's architecture consists of three independent modules that interact with one other. These modules are lexical analysis, syntactic analysis, and a resolution module for Natural Language Processing problems, such as anaphora.

Lexical analysis module. This module takes each sentence to parse as input, along with a tool that provides the system with all the lexical information for each word of the sentence. This tool may be either a dictionary or a part-of-speech tagger (POS tagger). In addition, this module returns a list with all the necessary information for the remaining modules as output. SUPAR works sentence by sentence from the input text, but stores information from previous sentences, which it uses in other modules (e.g. the list of antecedents of previous sentences for anaphora resolution).

Syntactic analysis module. This module takes as input the output of lexical analysis module and the syntactic information represented by means of grammatical formalism Slot Unification Grammar (SUG) [1]. It returns what is called slot structure, which stores all necessary information for following modules, and it also returns some discourse information that corresponds to a list of antecedents that is used for different NLP problems resolution such as anaphora.

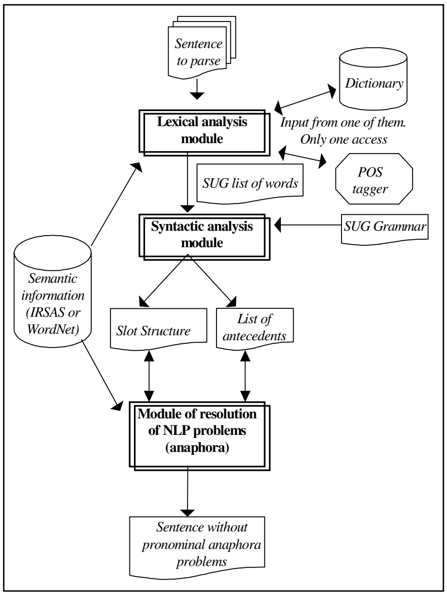

Sentence to parse
Dictionary
Lexical analysis module
Input from one of them. Only one access
SUG list of words
POS tagger
Syntactic analysis module
SUG Grammar
Semantic information (IRSAS or WordNet)
Slot Structure
List of antecedents
Module of resolution of NLP problems (anaphora)
Sentence without pronominal anaphora problems

Fig. 2. *SUPAR's architecture*

Slot Unification Grammar. SUG is a logical formalism based on unification, which is an extension of Definite Clause Grammars (DCG). It is called Slot Unification Grammar because of the slot structures in which it represents information. SUG has been developed with the aim of extending DCG in order to facilitate the resolution of NLP problems in a modular way. This formalism has been previously applied to anaphora resolution [2].

Partial parsing with SUG. One of the main advantages of this system is that it allows carrying out either partial or full parsing of the text. One can choose between these two kinds of parsing by configuring properly the system.

Module of resolution of NLP problems. In this module, NLP problems (e.g. anaphora, extra-position, ellipsis or PP-attachment) are dealt with. It takes the slot structure (SS) that corresponds to the parsed sentence as input. The output is an SS in which all the anaphors have been resolved. In this paper, only pronominal anaphora resolution has been applied.

The kinds of knowledge that are going to be used in pronominal anaphora resolution in this paper are: pos-tagger, partial parsing, statistical knowledge, c-command and morphologic agreement as restrictions and several heuristics such as syntactic parallelism, preference for noun-phrases in same sentence as the pronoun preference for proper nouns.

We should remark that when we work with unrestricted texts (as it occurs in this paper) we do not use semantic knowledge (i.e. a tool such as WorNet).

Presently, SUPAR resolves both Spanish and English pronominal anaphora with a success rate of 87% and 84% respectively [3] [4].

We should emphasize that SUPAR is based on the separation of different modules of a typical NLP system: lexical analysis, syntactic analysis and resolution of NLP problems. This allows one to produce modular NLP systems in which grammatical rules and the module of resolution of NLP problems are quite independent of one another. In this way, one can modify a module without significantly affecting the other modules in the system.

4.3 Anaphora resolution and IR

The object of this approach is to compare IR performance using pronominal anaphora resolution with baseline system. So, once documents are indexed by baseline system, SUPAR module is applied over text collection to solve pronominal references in these documents. This process will obtain the list of pronominal references in each document that refer to the same entity. These documents are then weighted with baseline scores, with the only difference that pronominal references (that baseline system ignore) are substituted with occurrences of the entity they refer and given their weight. As result we have two sets of documents indexed separately whose unique difference is the value given to pronominal information.

For comparing performance, the same question sets are processed through both sets of documents. Similarity between queries and documents is determined using cosine measure in both cases. In this way, we are sure that differences between system performance will only be caused by application of anaphora resolution.

4.4 Anaphora resolution and QA

Our QA approach provides a second level of processing after a set of documents have been retrieved by a standard IR system as baseline described above. The retrieved documents are then processed applying SUPAR module and pronouns are associated with their antecedents. Next, documents are split into sentences. Sentence is the unit of text used for QA task. As our approach is a sentence-extraction QA system, sentences will be scored and ranked depending on their relevance to the query. All this process is grouped into two general stages: Analyzing matching documents and Sentence ranking.

Analyzing Matching Documents. This step is applied over the best matching documents retrieved from an IR system. These documents are analyzed by SUPAR module and pronominal references are then solved. As result, each pronoun is associated with the noun phrase they refer to in the analyzed documents.

Once pronominal references are solved, documents are split into sentences as basic text unit for QA purposes. This set of sentences and the list of pronominal coreferences are sent to sentence ranking stage.

Sentence Ranking. Before sentence ranking, each term in the query is assigned a weight. This weight is the sum of inverse document frequency measure of terms based on its occurrence in the Time collection described earlier. Each document sentence is weighted the same way. The only difference with baseline is that pronouns are given the weight of the entity they refer. As we only want to analyze the effects of pronominal reference resolution, no more changes are introduced in weighting scheme. For sentence ranking, cosine similarity is used between query and document sentences. This measure is the same used for retrieving initial relevant documents.

Once sentences of relevant documents are evaluated, they are ranked based on its similarity with the query. The system returns to the user the first ranked phrases as the most relevant to the query.

4.5 Evaluation

Two different evaluations are proposed for measuring pronominal anaphora performance over IR and QA systems.

IR evaluation. For this evaluation, we used the 83 queries included in TIME collection. We obtain performance of baseline system using these evaluation queries for computing precision and recall rates. For measuring performance of anaphora resolution applied over IR systems, all documents and queries are processed by SUPAR module obtaining a set of documents and queries with pronominal references solved. These documents are processed the same way as initial collection. The only difference in document weighting is that instead of ignoring pronouns, they are given the weight of the entity they refer. Initial results are presented graphically in Figure 3.

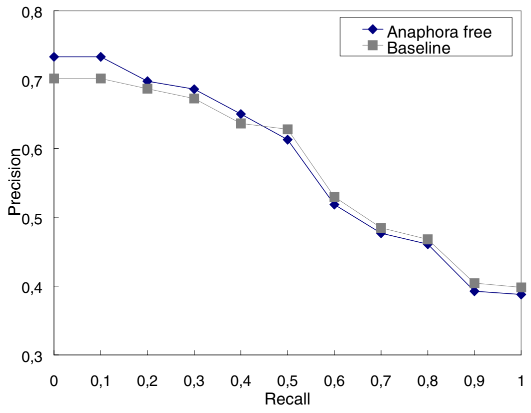


Fig. 3. *Initial IR results*

A question arises when analyzing obtained results: Has pronominal anaphora resolution improved IR precision or not? The answer was found analyzing test queries. When queries include terms that are referred pronominally in documents precision increases. This happens because the number of references to these terms increases when pronominal anaphora is solved. On the contrary, when queries do not include terms referred pronominally in documents, precision decreases a little. This occurs because the terms not referred pronominally lose relative importance in documents when pronominal references are solved and given the weight of the terms they refer. For measuring these facts, the collection of queries was split into two groups. First group was formed by queries containing terms that were referred pronominally in documents. ¿From the initial group of 83 queries, 21 pertained to this group. The rest of the queries were put together into the second group. A new evaluation for each group was carried out for comparing results. Figures 4 and 5 show precision-recall results for first and second group respectively.

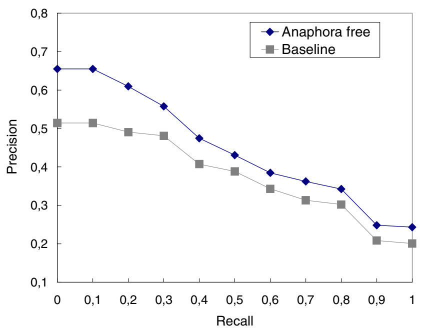


Fig. 4. Group 1 results

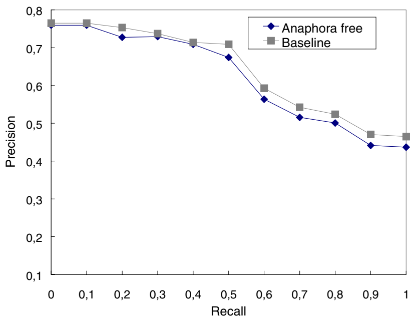


Fig. 5. *Group 2 results*

Final analysis of IR results shows that an improvement in precision is achieved when query includes terms that have been referred pronominally in documents and pronominal anaphora is solved. This improvement (a mean of 7,26%) is greater than the loss in precision that is obtained in the other case (a mean of 2,01%). With these results we can conclude that solving pronominal references for IR tasks is important when queries include terms that probably are referenced pronominally in documents.

QA evaluation. For this evaluation, twenty queries were proposed. These queries were selected based on their expressing the user's information need clearly and their being likely answered in a single sentence. ¿From the proposed queries, only eight were finally analyzed due to the use of pronouns in the sentence containing the correct answer. Remaining queries are not detailed here because no pronominal references appeared in the sentence containing the correct answer and anaphora resolution did not affect system performance. The analyzed queries were the following:

1. Who visited Walter Ulbricht on his 70th birthday?
2. How is personal relation between Mao Tse-Tung and Khrushchev?
3. What does Khrushchev think about Kennedy strategy?
4. Why Katanga's president left his country?
5. How was Kennedy welcomed in Berlin?
6. Which is the reason for Khrushchev's visit to Berlin?
7. Who advised Allen Howl about real Vietnam problem?
8. Has Moise Tshombe taken out money from his country?

In first four sentences, the information required as answer to each query was referred via pronominal anaphora in the relevant sentence (sentence in documents where answer to the query is located). In last four queries, any term in the query was referred pronominally in the relevant sentence.

The information requested by each query was searched in the baseline IR system constructed for retrieving the best fifteen matching documents. Our system processed these documents and solved pronominal anaphora references. Then the system measured similarity between the query and two groups of sentences and presented a ranked listing of sentences for each group. Group A is formed by the sentences of relevant documents with anaphoric references solved. Group B is composed by the same sentences but without solving pronominal references.

Anaphora resolution performance is presented as the top ranked sentence that contained the answer to the question (see Figure 6). Second column shows results for documents where pronominal anaphora has been solved (Group A). Third column shows results for the same documents but without solving pronouns (Group B).

QUERY	**First answer's rank**	
	Group A **Anaphora QA system**	**Group B** **Baseline**
1	1	1
2	3	2
3	2	2
4	3	3
5	1	3
6	2	3
7	6	9
8	2	7

Fig. 6. *QA evaluation results*

QA task results analysis shows several aspects we have to take into account. As we stated in the evaluation, first four queries asked for information referred pronominally in the correct answer sentence. Results obtained for these queries show us that relevance to the query is near the same as baseline system. So, it seems that pronominal anaphora resolution does not achieve important improvements for QA systems. This is not the whole truth. Although the answer sentence is ranked similarly, the precise answer will be displayed in the sentence via pronominal coreference and the user will not need to read more text to discover this answer. For noun-phrase extraction QA systems the improvement is greater. If the pronominal coreference is not solved, pronouns will not be analyzed and if co-occurrence metrics are applied to find the answer (MURAX), probably a wrong noun-phrase will be given as answer to the query.

Results get better if we analyze performance of last four queries. These queries have the following characteristic. Some of the query terms were referred via pronominal anaphora in the relevant sentence. As we can see in the results table, when this situation occurs relevant sentences are retrieved earlier in the final ranked list than in the baseline list. This improvement is due to the similarity increase between query and answer sentence when pronouns are weighted with the same score as the terms they refer to.

As results show, we can say that pronominal anaphora resolution improves QA systems performance in several aspects. First, precision increases when query terms are referred anaphorically in response sentence. Second, pronominal anaphora resolution reduces the amount of text a user has to read when the answer sentence is displayed and pronominal references are substituted with their coreferent noun phrases. And third, for noun phrase extraction QA systems it is essential to solve pronominal references if a good performance is pursued.

4.6 Conclusion and future research

The analysis of information referred pronominally in documents has revealed to be important to tasks where high level of precision is required. We have measured effects of applying pronominal anaphora resolution to IR and QA systems with different results. When applied over IR systems, precision improvement is achieved if queries include terms referred pronominally in documents. Its application over QA systems improves performance and seems to be essential in some cases.

Two main areas of future work have appeared while investigation has been developed. First, anaphora resolution algorithm has to be extended to different types of anaphora such as definite descriptions, surface count, verbal phrase and one-anaphora. And second, a noun-phrase QA system has to be developed to take advantage of anaphora resolution techniques described here.

References

1. A. Ferrández and M. Palomar and L. Moreno. Slot Unification Grammar. In *Joint Conference on Declarative Programming (APPIA-GULP-PRODE)*, 1997.

2. A. Ferrández and M. Palomar and L. Moreno. Slot Unification Grammar and anaphora resolution. In *Recent Advances in Natural Language Processing (RANLP)*, 1997.
3. A. Ferrández and M. Palomar and L. Moreno. A computationa approach to pronominal anaphora, one-anaphora and surface count anaphora. In *Second Discourse Anaphora and Resolution Colloquium (DAARC2)*, 1998.
4. A. Ferrández and M. Palomar and L. Moreno. Anaphora resolution in unrestriced texts with partial parsing. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Lingustics*, 1998.
5. A. Ferrández and M. Palomar and L. Moreno. An empirical approach to Spanish anaphora resolution. *To appear in Machine Translation*, 1999.
6. J. Berri and D. Moll and M. Hess. Extraction automatique de r ponses: impl mentations du syst me ExtrAns. In *TALN 'Traitement Automatique des Langues Naturelles'*, June 1998.
7. K. Spark Jones. *What is the Role of NLP in Text Retrieval?*, pages 1–24. In [14], 1999.
8. J. Kupiec. *MURAX: Finding and Organising Answers from Text Search*, pages 311–331. In [14], 1999.
9. T. Morton. Using Coreference for Question Answering. In *ACL Workshop on Coreference and Its Applications*, 1999.
10. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley, New York, 1998.
11. T. Strzalkowski and F. Lin and J. Perez-Carballo. Natural language information retrieval: TREC-6 report. In *Sixth Text REtrieval Conference*, 1998.
12. T. Strzalkowski and F. Lin and J. Wang and L. Guthrie and J. Leistensnider and J. Wilding and J. Karlgren and T. Straszheim and J. Perez-Carballo. Natural language information retrieval: TREC-5 report. In *Fifth Text REtrieval Conference*, 1997.
13. T. Strzalkowski and G. Stein and G. Bowden Wise and J. Perez-Carballo and P. Tapananinen and T. Jarvinen and A. Voutilainen and J. Karlgren. Natural language information retrieval: TREC-7 report. In *Seventh Text REtrieval Conference*, 1999.
14. T. Strzalkowsky. *Natural Language Information Retrieval.* Kluwer Academic, New York, 1999.

MB+tree: A Dynamically Updatable Metric Index for Similarity Search

Masahiro Ishikawa[1], Hanxiong Chen[2], Kazutaka Furuse[1], Jeffrey Xu Yu[3], and Nobuo Ohbo[1]

[1] Institute of Information Sciences and Electronics, University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan
{mi,furuse,ohbo}@dblab.is.tsukuba.ac.jp

[2] Department of Computer Science, Tsukuba International University
Tsukuba, Ibaraki 300-0051, Japan
chx@dblab.is.tsukuba.ac.jp

[3] Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
yu@se.cuhk.edu.hk

Abstract. One of the common query patterns is to find approximate matches to a given query object in a large database. This kind of query processing is referred as similarity search in a metric space. In this paper, we propose a new metric index MB+tree, called *Metric B+tree*, which supports near neighbour searching in a generic metric space. MB+tree is aimed at reducing both the number of I/O accesses and the number of distance calculations for similarity search in large databases, while allowing dynamic data updates. In this paper, we show that a B+tree, with an auxiliary tree, can be used as a metric index. Unlike other multi-dimensional (spatial) access methods, using our approach, we can partition data into disjoint partitions while building/maintaining a metric index, which can lead to a significant cost reduction since the number of metric sub-spaces to be searched is reduced. In order to use MB+tree, a slicing value is proposed. With the slicing value, in addition to space division information, a near neighbour searching can be systematically converted to a range search in B+tree. Several different slicing values are considered namely, one-focus-point scheme and two-focus-point scheme. We also conducted extensive experimental studies using synthetic data. Results are reported in this paper.

1 Introduction

Recently, the needs for using various types of data including textual data, image, picture, voice, time series data, etc., have drastically increased. One of query patterns is for users to find similar objects to a given query object in large databases, where the similarity between objects can be computed using an application-specific distance function. In order to minimize the query processing time for similarity queries, new index mechanisms need to be developed. Multi-dimensional indices such as R-tree [5,8] can be used to index such data, since in many cases data objects can be represented in a multi-dimensional vector space on which a distance function can be defined to measure dissimilarities between objects. However, there is an implicit assumption in multi-dimensional indices that L_2(Euclidean) distance is used. Thus, it is difficult to use such indices when non-L_2 distances (e.g. histogram distance) are used [3] or when InsDel and Edit distances on strings are considered. Metric indices such as GNAT [6] and MVP-tree [2] have been proposed for a generic metric space. However, the two indices are static, i.e. all data must be available at construction time, and dynamic updates are not taken into account. Furthermore, they process data in memory, and are not designed as an index for handling a large amount of data on secondary storage. M-tree [4], which is a paged index like R-tree, supports dynamic updates for large data sets to be stored on secondary storage. However in M-tree, a sub-space corresponding to an indexing node is a (hyper-)sphere, and therefore, two sub-spaces tend to be overlapped with each others. Thus, unnecessary distance calculations are needed in M-tree than needed in static indices [3].

In this paper, we propose *MB+tree* (*Metric B+tree*), which supports dynamic updates and handling large data sets on secondary storage. The MB+tree is built using an B+tree and an auxiliary tree, called a block tree. The block tree is considerably small, and is used to keep minimal space division information. Based on the information kept in the block tree, we generate keys for objects in a generic metric space, and use those keys to build, maintain, and search objects in B+tree. The key generation approaches and key structure will be discussed in this paper later. For a given similarity query, our approach is to systematically convert the similarity query to a range (sequential) query against B+tree.

The rest of this paper is organized as follows. Preliminaries are given in Section 2. In Section 3, MB+tree is introduced. Its key components, data structures and operations are presented. Section 4 introduces an optimal MB+tree which ensures the MB+tree to be balanced. The optimal MB+tree shows that our MB+tree can work efficiently when a whole set of data is given. Experimental results on synthetic data are shown in Section 5. Section 6 concludes this paper.

2 Preliminaries

Indexing a metric space means to provide an efficient support for retrieving objects which are "similar" to a reference (query) object, where the (dis)similarity between objects is measured by an application-specific metric distance function d. We adopted similar notations as those used in [4]. Formally, a metric space is a pair, $\mathcal{M} = (\mathcal{O}, d)$, where $\mathcal{O}$ is a set of data values and d is a total (distance) function with the following properties (*the metric postulate*):

$$\forall x, y \in \mathcal{O}; \qquad d(x, y) = d(y, x) \tag{1}$$

$$\forall x \in \mathcal{O}; \qquad d(x, x) = 0 \tag{2}$$

$$\forall x, y \in \mathcal{O}; x \neq y \rightarrow 0 < d(x, y) < \infty \tag{3}$$

$$\forall x, y, z \in O;\ d(x,z) \leq d(x,y) + d(y,z) \tag{4}$$

The main purpose of metric indices is for reducing both I/O accesses and the number of distance calculations at search time. The latter is significant since the evaluation cost of a distance function can be very expensive, particularly when multi-dimensional data are used. The two basic types of similarity queries in the literature are *r-neighbour search* and *k-nearest neighbour search.*

Definition 1 (r-neighbour search). *Given a query object q ($\in \mathcal{O}$) and a non-negative radius r, the r-neighbour search(r-N search) of q is to retrieve the objects in the r-neighbour of q: $RN(q,r) = \{o \,|\, o \in \mathcal{O} \wedge d(q,o) \leq r\}$.*

Definition 2 (k-nearest neighbour search). *Given a query object q ($\in \mathcal{O}$) and a positive integer k, the k-nearest neighbour search(k-NN search) of q is to retrieve the k nearest objects to q, such that: $KN(q,k) = \{o \,|\, o \in \mathcal{O} \wedge (\exists r; \forall r' < r; o \in RN(q,r) \wedge k \leq |RN(q,r)| \wedge |RN(q,r')| < k)\}$.*

Metric indices are built based on data partitioning which partitions data according to distances of the objects with respect to a reference point. Some values, for example, median of such distances, are used as a separator to partition objects into subsets. Repeating the same procedure results in a tree-structured index. The two basic types of partitioning are *vp-tree* (*vantage point tree*) and *gh-tree* (*generalized-hyper-plane tree*) [7].

The vp-tree partitions a data set by (hyper-)spheres. In a vp-tree, a vantage point is chosen from the target data set, and distances between the vantage point and other objects are calculated. The median value of such distances is used to partition the data space into two. On the other hand, the gh-tree partitions a data set by *generalized hyper-planes.* First, the gh-tree chooses two vantage points from the target data set, as the representatives of partitions. Second, objects are clustered to a partition if the distance to the corresponding vantage point of that partition is smaller. Repeating the above process also results in a tree. Notice that the vp-tree is a balanced tree, whereas the gh-tree is not necessarily balanced. In the literature, MVP-tree and GNAT are the generalization of vp-tree and gh-tree, respectively. Both MVP-tree and GNAT can partition data into more than two partitions at once. MVP-tree stores the distances calculated at construction time for candidate filtering at search time. GNAT stores the minimum and maximum distances between each vantage point and each sub-space at construction time.

With a metric index, candidates can be selected based on the formula (4) (triangle inequality). As for r-neighbour, a vantage point vp is chosen from data set $\mathcal{O}$. The objects, q', that can be possibly found in the r-neighbour in terms of a given querying object q, are those that satisfy the following condition, $R - r \leq d(vp, q') \leq R + r$, where $R = d(vp, q)$. It is illustrated in Fig. 1. Here, a few observations can be made.

– If distances between vp and other objects have been calculated and stored in the metric index at insertion time, only $d(q, vp)$ needs to be calculated at retrieval time.

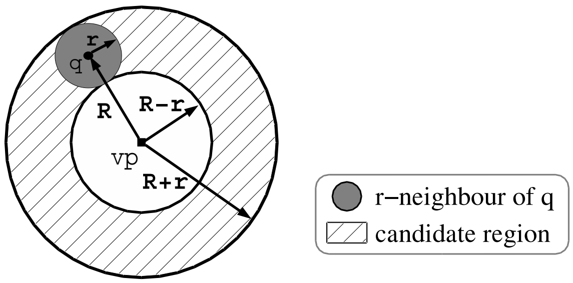


Fig. 1. Triangle Inequality and Candidate Filtering

- Only the triangle inequality can be used for candidate filtering in a generic metric space.
- Distances between the query object q and candidate objects need to be calculated, in order to obtain the answer set of objects.

3 MB+tree

As for metric indexing mechanisms, one of the most important issues is how to partition data and therefore build an index tree for later use. M-tree[4], as a dynamic balanced metric index, partitions objects in the basis of their relative distances. M-tree grows in a bottom-up fashion. The overflow of a node N is managed by allocating a new node, N', at the same level of N, partitioning the entries among these two nodes, and posting (promoting) to the parent node, N_p, two reference objects, to reference to the two nodes. The ideal split policy is to minimize both the volume and overlapping between the two nodes. However, in general, overlapping can not be avoided. The question that raises here is whether it is possible to avoid any possible overlapping for a dynamic balanced metric index. In this paper, we propose a dynamic metric index named MB+tree, which has the following unique features.

- A slicing value is used to partition data. More precisely, each object in the data space is associated with a slicing value generated from relative distances between objects. Based on the slicing value, disjoint partitioning can be done, which can significantly reduce disk accesses. As a result, index mechanisms for sequential data can be possibly used for metric data, like B+tree. The difference between hyper-sphere based partitioning used in M-tree and slicing value based partitioning used in our MB+tree are illustrated in Fig. 2.
- MB+tree consists of two separated but interrelated tree structures, namely, a block tree and a B+tree. The reasons of using two separated trees are given below. First, most commercial DBMSs support B+tree. Second, we can simplify our task to design a new metric index for efficiently processing

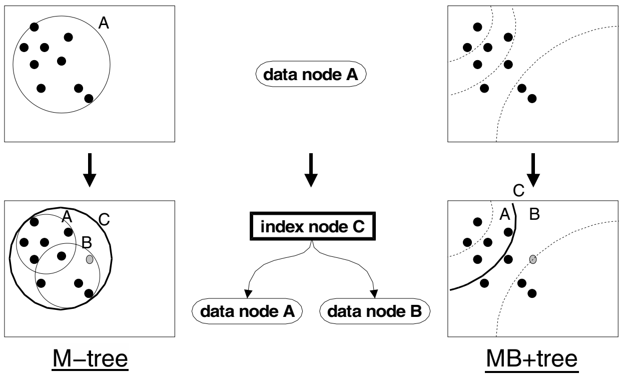


Fig. 2. Hyper-Sphere Partitioning vs. Slicing Value based Partitioning

similarity queries. Third, we can minimize the impacts of unbalanced trees. Note B+tree is always balanced. The block tree might not be always balanced but it is considerably small. Finally, some information can be stored in block tree for pruning purposes. With the block tree, for a given r-neighbour query, we can determine the corresponding range to be used to search data in B+tree. As a future work, we seek the possibilities of combining these two trees to a single index tree. However, in this paper, our main intension is to minimize query processing cost while handling dynamical updates, rather than the tree structure itself.

The block tree is used to represent space divisions. In addition, the information kept in the block tree is used to generate keys which consists of two kinds of information, sub-space and slicing value. The generated keys, in addition to distances being calculated, are stored into the corresponding B+tree. It is important to know that it is critical to generate keys in our proposal in a way that r-neighbour in terms of a query object needs to correspond to a range search with B+tree. All objects that can satisfy r-neighbour in terms of a reference query object must be all included.

An MB+tree example is shown in Fig. 3, in which figure (a) shows an example of division of spaces and slicing of sub-spaces, and figure (b) depicts the block tree of (a) representing the hierarchical structure of sub-spaces.

Another MB+tree example is given in Fig. 4. Fig. (c) shows how to split a data space into two sub-spaces. Fig. (b) is for the block tree, according to the space division. The root node of the block tree is for the whole data space. The block tree node with $\underline{0}0$ is for the upper-left sub-space, whereas the block tree node with $\underline{1}0$ is for the lower-right sub-space. Fig. (a) shows the corresponding B+tree, where $\underline{1}0.00$ is a key which consists of two parts, the block and the slicing value. In this example, the block value is $\underline{1}0$ which is the block number of the right child node. The slicing value is 00. The middle curve in Fig. (c) is determined by a slicing value 00.

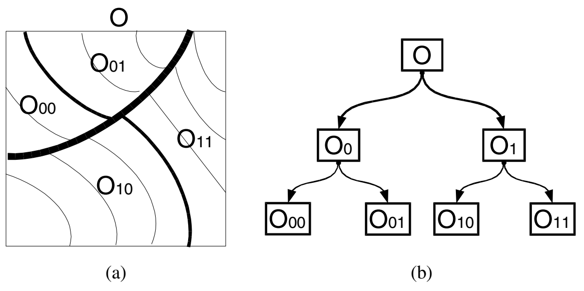


Fig. 3. Recursive Space Division and Slicing

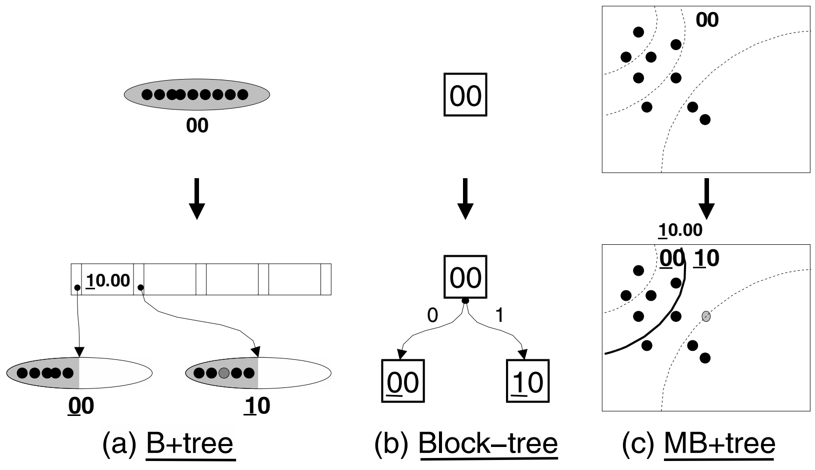


Fig. 4. An Example of MB+tree

In the following subsections, we will discuss the following issues: slicing value, key generation, searching and MB+tree construction, in details.

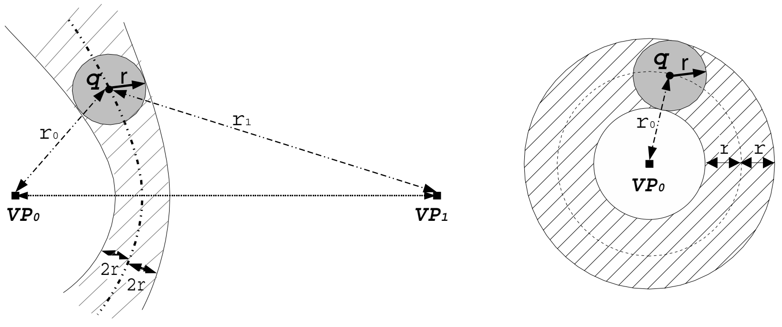
3.1 Slicing Values

Definition 3 (Slicing value). *Given two reference objects fp_o, fp_1 in a data space. A slicing value of an object q is defined as $d_r(q) = r_0(q) - r_1(q)$ where, $r_0(q) = d(fp_0, q)$ and $r_1(q) = d(fp_1, q)$.*

It is worth noting that the slicing value is chosen in a way that it exists in a certain range as given below.

$$-R \leq d_r(q) \leq R \tag{5}$$

where $R = d(fp_0, fp_1)$. An example of slicing value is shown in Fig. 5. Given a range r and a reference object q, the search space in the data set can be determined using two approaches, namely, the two-focus-point scheme and the one-focus-point scheme.



(a) Hyperbola Method (b) Circle Method

Fig. 5. Slicing Values of an Object q.

The two-focus-point scheme

The two-focus-point scheme is based on Lemma 1.

Lemma 1. *Given a range r and a reference object q. If $q' \in RN(q,r)$, then $d_r(q) - 2r \le d_r(q') \le d_r(q) + 2r$.*

Proof: The slicing value is determined in terms of two focus points fp_0 and fp_1. Let $r_i(p)$ be the distance between p and the focus point fp_i. Based on the triangle inequality, we have the following four equations.

$$0 \le r_0(q') \le r_0(q) + r \tag{6}$$
$$0 \le r_1(q') \le r_1(q) + r \tag{7}$$
$$0 \le r_0(q) \le r_0(q') + r \tag{8}$$
$$0 \le r_1(q) \le r_1(q') + r \tag{9}$$

Based on equation (6) and (9), we have

$$r_0(q') - (r_1(q') + r) \le (r_0(q) + r) - r_1(q)$$

In a similar fashion, based on equation (7) and (8), we have

$$r_0(q) - (r_1(q) + r) \le (r_0(q') + r) - r_1(q')$$

Then,

$$r_0(q') - r_1(q') \le (r_0(q) - r_1(q)) + 2r$$
$$r_0(q) - r_1(q) - 2r \le r_0(q') - r_1(q')$$

Therefore,

$$d_r(q) - 2r \le d_r(q') \le d_r(q) + 2r\ .$$

The following question is what is the optimality for selecting the two focus points, since they have significant impacts on performance for searching. As can be seen in expression (5), the upper and lower bounds of a slicing value are determined by the distance, R, between two focus points. The ratio of the slice range of r-neighbour is also dependent on the same distance value. Therefore, the possibility we have to search two neighbour sub-spaces is higher if a smaller R is chosen. This highlights the fact that, for a sub-space, it is better to choose two focus points from each other as far as possible. However, it is expensive to calculate distances between any pair of objects in a data space. A heuristic approach is proposed as follows. Let a data set be $\mathcal{O}_{\mathcal{N}}$. We iteratively proceed the following steps for n times.

1. Randomly choose a point fp_0 from $O_{\mathcal{N}}$.
2. Find another fp_1 such that it has the largest distance to fp_0, from $O_{\mathcal{N}}$.
3. Replace fp_0 with fp_1, then goto 2.

After n time iterations, we expect that the distance between fp_0 and fp_1 to be large enough, and use them as the focus points.

The one-focus-point scheme

Let fp_0 be a focus and let $r_0(q)$, the distance between q and fp_0, be used as a slicing value of q in O_i. Then, obviously, if $q' \in RN(q, r)$ then the following equation holds.

$$r_0(q) - r \leq r_0(q') \leq r_0(q) + r \tag{10}$$

Also, a heuristic approach is taken in order to select a focus. The procedure is given below.

1. Randomly choose a vantage point fp_0 from the data set.
2. Calculate distances between fp_0 and other points.
3. Let the median value of the distances be d_m. Calculate the deviations, dev, with respect to d_m such as $dev = \sum_{o \in O} |d(fp_0, o) - d_m|$.
4. Let fp_0 be the new focus point such that the distance between this new focus point and the previously selected focus-point is largest with max deviation.
5. Goto 2.

After several iterations, the fp_0 with a larger deviation is chosen.

3.2 Keys

A key is constructed using n bits. The first K_b bits are used to keep a block value which identify one of the sub-space, and the following K_s bits are used to keep a slicing value in the sub-space identified by the K_b bits. The number of K_b bits determines both the height of a block tree and the number of sub-spaces. The sum of K_b and K_s bits shall be n. In Fig. 6, eight bits are assigned to K_b. The

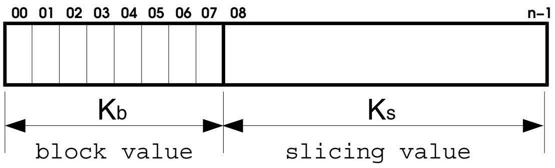


Fig. 6. Structure of a Key (example)

intension of designing keys in this way is that the ranges of keys for any pair of sub-spaces do not overlap. Thus, we can manage objects in different sub-spaces into a B+tree and data node, and therefore traverse can be performed for objects in a sub-space. Two key generation approaches are used. One is called *hyperbola method* which is based on the two-focus-point scheme. The other is called *circle method* which is based on the one-focus-point scheme. For a given slicing value, it is normalized to be represented with K_s bits as follows. The $d_r(q)$ is converted to a value, $S_v(q)$, in the range of $(0, 2^{K_s} - 1)$.

$$S_v(q) = \left\lfloor (2^{K_s} - 1)\frac{d_r(q) + R}{2R} \right\rfloor$$

The assignment of block bits is discussed in the following section.

3.3 The Block Tree

The following information is stored in each block tree node.

- Block values of sub-spaces: The block values are the K_b bits of keys. All block bits for the root node of the block tree are zeros. The zero repeats K_b times. If the data set represented by the root node is divided into two sub-spaces. The most significant bit of the block value for the left node remains unchanged. The most significant bit of the block value for the right node is reset to 1. Suppose the level of the root node is 0. The $j + 1$ most significant bit of a block value for the right node of a node at level j is reset to 1. All objects in the same sub-space have the same K_b bits.
- Pointers to child nodes (sub-spaces).
- Dividing slicing values of sub-spaces: A dividing slicing value is a slicing value, the space boarder, that represents one of sub-spaces. The dividing slicing values are needed to know which child nodes must be searched.
- Reference objects (or identifiers of the reference objects): These reference objects are used to determine a slicing value for an object. The slicing value of an object plus the block bits form a key to be inserted into B+tree and to be used for candidate filtering.

3.4 Near Neighbour Searching

Given a query object q and radius r, r-neighbour search starts from searching with the block tree, which is illustrated in Fig. 7. The `blockSearch` function

```
## B - block to search (a sub-space)
## q - query object
## r - query radius
## dist(,) - distance function
blockSearch(B, q, r)
{
    if (B is a leaf block) {
        record B, minSlice, maxSlice as an answer;
    } else {
        ## determine the slice range of (q,r)
        d ← dist(q, reference object of B);
        minSlice ← d - r;
        maxSlice ← d + r;
        ## search intersecting child blocks
        if (minSlice < split value of B)
            blockSearch(1st child block of B, q, r);
        if (maxSlice >= split value of B)
            blockSearch(2nd child block of B, q, r);
    }
}
```

Fig. 7. Procedure for Block Search

finds the leaf blocks in the block tree that intersect the query region of q with radius r. For such a leaf block, the `minSlice` and `maxSlice` values, as shown in Fig. 7, can be obtained, in addition to the block value. The block value and `minSlice` form a key, `minKey`, whereas the same block value and `maxSlice` form another key, `maxKey`. The `minKey` and `maxKey` are used as the range to do range search in B+tree. Finally, for those objects that are in the range specified by `minKey` and `maxKey`, we need to calculate the distances to check whether they are in the range of r in terms of the query object q.

Given a query object q and a positive integer k, k-nearest neighbour search can be processed using r-neighbour search as shown below.

1. Find the sub-space in which q should be located by using the function, `blockSearch`, as given in Fig. 7, with radius $r = 0.0$.
2. Traverse the data node entries in B+tree around q to find candidate objects of the answer set. Find an r value such that the distance is largest between r and q among all candidate objects in the sub-space.
3. Perform r-neighbour search of q with radius r. Find the k nearest objects from retrieved objects.

3.5 B+tree

B+tree is widely used technique in many database systems. In this section, we only discuss node splitting which is slightly different from the existing splitting procedure used in B+tree. Two cases are considered as follows.

Case-1 (Data Node Splitting with Sub-space Division):
A data node contains objects in the same sub-space. When a data node

is split, the corresponding sub-space is also divided where possible. Fig. 8 illustrates a data node splitting and the sub-space division. In Fig. 8(a), two

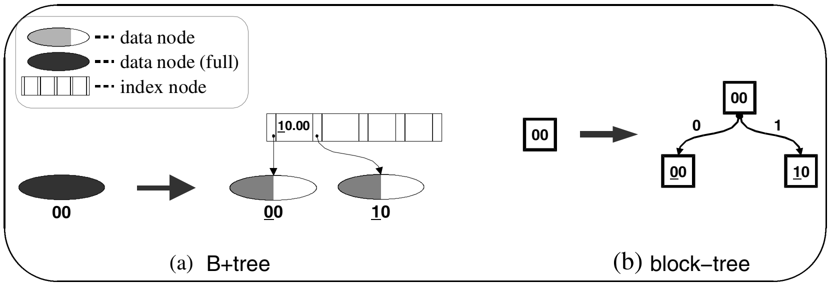


Fig. 8. Split of Data Node with Division of Sub-space (1)

digits under a data node are block bits, which identify the corresponding sub-space in the block tree shown as Fig. 8(b). If division of sub-space occurs, the block value of the right node is promoted to the new index node as a key. In Fig. 8(a), 10.00 located in the index node is the promoted key, – its integral part represents the block bits and the fraction part represents the slicing value.

Fig. 9 illustrates succeeding split. Note, B+tree is always balanced while the block tree may be unbalanced. The order of keys remains unchanged after

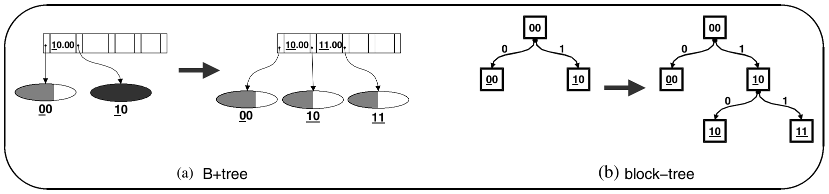


Fig. 9. Split of Data Node with Division of Sub-space (2)

succeeding splittings, because the block bits are used from left to right in order. However, block values of entries distributed to the new right node must be altered at splitting time.

Case-2 (Data Node Splitting without Sub-space Division):

Fig. 10 illustrates a process of splitting a data node without sub-space division. The reason why the sub-space is not split is that all block bits assigned to the data node has already been consumed. In this case, the data node is split by the median of slicing values and its key is promoted in the same way as discussed in B+tree.

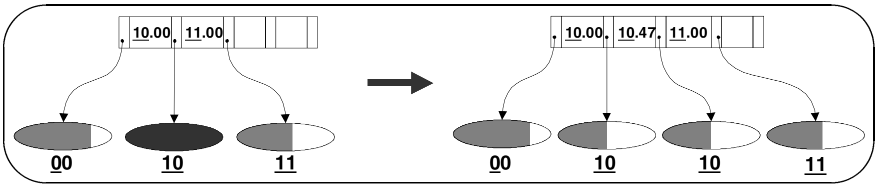


Fig. 10. Split of Data Node without Division of Sub-space

4 The Optimal MB+tree

Although MB+tree is designed to allow dynamic updates, the resultant MB+tree might be skewed. If an MB+tree was skewed, search time performance will be degenerated. This is due to the fact that reference objects cannot reflect the distribution of the set of whole indexed objects, because each reference object is chosen from a data node. However, when many objects are available at construction time, more suitable reference objects can be found by choosing them from the set of whole objects. The optimal MB+tree means that the MB+tree is completely balanced – both the block tree and B+tree are balanced.

Given a data set $\mathcal{O}$ and a distance function d, an optimal MB+tree on $\mathcal{O}$ can be constructed in the following way.

1. Let $\mathcal{O}$ be the root node of block tree, and $D[\,]$ be an array storing distances from data points to the focus point. Initially, $D[\,]$ is empty.
2. If $|\mathcal{O}|$ is under the limitation of a data node or $\mathcal{O}$ can not be further divided, then a key is generated based on the current block tree. Here, each $o \in \mathcal{O}$ is inserted to B+tree. The distance of o to the focus point is set in $D[o]$. Otherwise, two focus points fp_0 and fp_1 are chosen from $\mathcal{O}$, then all $D[o]$, for $o \in O$, are reset.
3. Let the middle value of the differences of distances from each point to the two focus points be d_m. $\mathcal{O}$ is then divided into two divisions,
$$\mathcal{O}_0 = \{o \,|\, o \in \mathcal{O} \,\wedge\, d(o, fp_0) - d(o, fp_1) \leq d_m\}$$
$$\mathcal{O}_1 = \{o \,|\, o \in \mathcal{O} \,\wedge\, d(o, fp_0) - d(o, fp_1) > d_m\}$$
Also, the sub-space $\mathcal{O}$ is divided into $\mathcal{O}_0, \mathcal{O}_1$. Focus points are selected using the aforementioned heuristic algorithm.
4. The above procedure repeats for $\mathcal{O}_0$ and $\mathcal{O}_1$, recursively.

The details of the algorithm for constructing an optimal MB+tree is given in Fig. 11.

5 Experimental Studies

We have done extensive performance studies. In this section, we show some experimental results of near neighbour searches with MB+tree. Experiments

```
## B   - block to search
## O   - set of objects
## D[] - array of distances from reference objects
construct(B, O, D[])
{
    if ((size of O is less than capacity of a data node)
       or (B cannot be split))
    {
        for o in O {
            key  ← generate key of o;
            data ← compose o and D[o];
            insert (key, data) into B+tree;
        }
    } else {
        choose reference objects from O;
        ## calculate and keep distances from reference objects
        for o in O {
            for vp in reference objects {
                D[o].append(dist(o, vp));
            }
        }
        O0, O1 ← divide O;
        B0, B1 ← divide B;
        construct(B0, O0, D[])
        construct(B1, O1, D[])
    }
}
```

Fig. 11. Procedure for Optimal Construction of MB+tree

are done with a Pentium III 500MHz PC and 384 MB main memory, running GNU/Linux. We use up to 32MB in our experimental studies. All MB+tree, MVP-tree and GNAT were implemented in C++ and compiled by using g++ 2.95.1 with Pentium patch. For MB+tree, we firstly implemented a B+tree as a base class, and MB+tree was implemented as a derived class of it.

The size of pages and the length of block bits are fixed to 4K bytes and 8 bits, respectively. In the experiments, we used clustered point data in Euclidean vector space in which each dimension range is $[0, 1)$. Data sets were generated using the algorithm given in Appendix H of [1] by which points in a cluster are generated with Gaussian distribution around a centroid. The centroids are randomly distributed in the space. In our experiments, variance of the Gaussian distribution is set to $\sigma^2 = 0.1$ and overlapping tolerance is set to 0.2. The number of clusters is fixed to 10 and all clusters have the same size. L_2 metric was used as a distance function. We used optimal MB+trees for the experiments.

Comparison of Circle Method and Hyperbola Method

First, we conducted a several tests to check the effectiveness of the two key generation approaches, namely, the circle method and the hyperbola method. The former is based on the one-focus-point scheme, while the latter is based on the two-focus-point scheme. Fig. 12 shows the results on 20,000 data in 10 dimensional spaces for retrieving 50 objects. The figure shows that the circle method is better in the number of accessed nodes, though the number of distance calculations are almost the same. This is due to the following fact. Using the circle method, the searching range is $d_r(q) - r \leq r_0(q') \leq d_r(q) + r$. While using the hyperbola method, it is $d_r(q) - 2r \leq d_r(q') \leq d_r(q) + 2r$. It suggests that the searching range is double when the hyperbola method is used instead of the circle method.

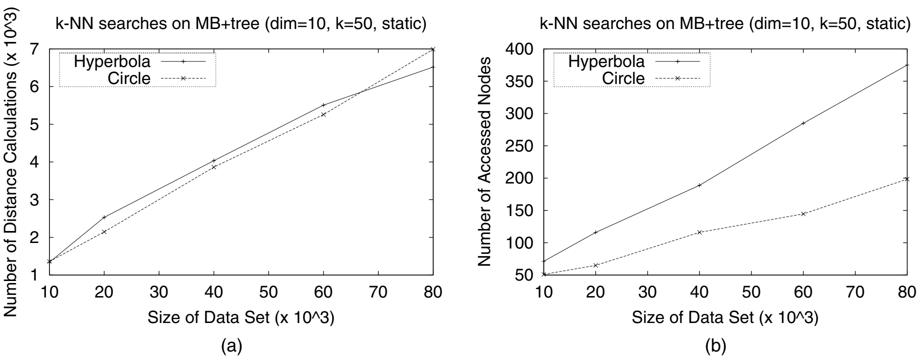


Fig. 12. Comparison of Circle and Hyperbola Methods (k=50)

We use the circle method in the following experimental studies.

Data Set Size vs. Performance

We investigated how the size of data sets affects the performance. We counted the number of distance calculations and the number of accessed nodes for k-nearest neighbour search, Dimensionality of data space and the size of a data set are varied in 2–50 and 10,000–80,000, respectively. The number of retrieved objects is varied in 10–1,000.

Fig. 13 (a) shows results on 10 dimensional data set. Keys are generated with the circle method using the one focus scheme. From the results, we observe that both the number of distance calculation and the number of page-based I/O accesses are proportional to the size of data sets. From Fig. 13, we can see that MB+tree is scalable in the size of data sets.

Dimensionality of Data vs. Performance

Fig. 14 shows how the dimensionality of data space affects the performance. In this experiment, the data size is fixed to 20,000, and other parameters are same as the above. From the results, we can see that both of them are logarithmically increase against the dimensionality of data space.

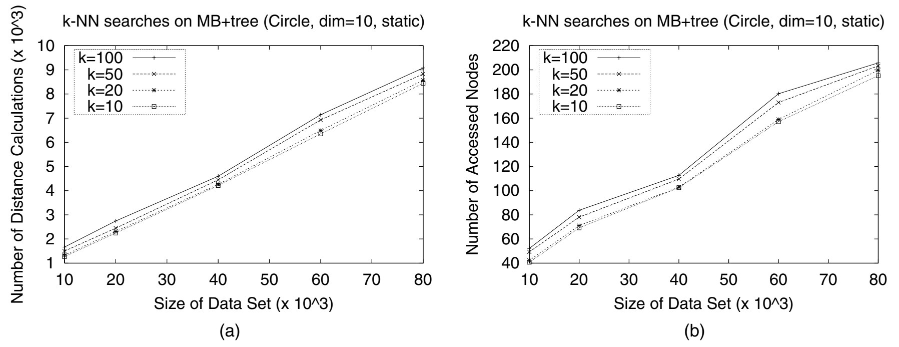


Fig. 13. k-NN Search Performance on Varied Data Set Sizes

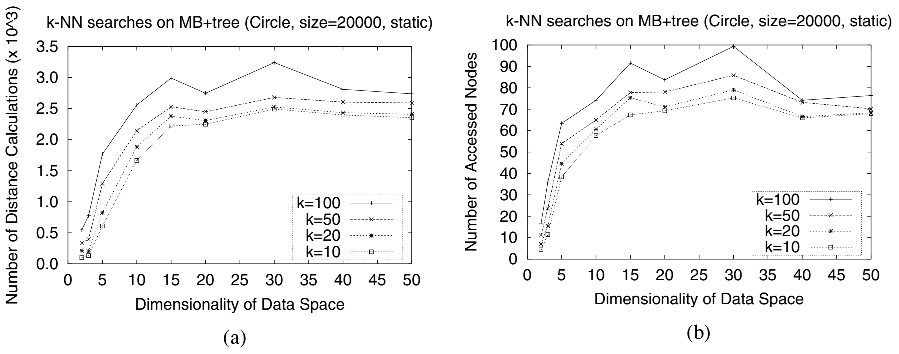


Fig. 14. k-NN Searches Performance on Varied Data Space Dimensions

Performance on Uniformly Distributed Data

It is well known that when multi-dimensional indices are applied to vector data, its performance degenerates as the number of dimensions increases. This phenomenon is known as *the curse of dimensionality*. 'Degenerate' means that, almost all data must be touched for a search. We conducted experiments on uniformly distributed data to see whether such an phenomenon appears in MB+tree. In this case, data set is uniformly distributed 20,000 points in vector spaces. The results in Fig. 15 shows that MB+tree is not effective for uniformly distributed data with 15 or any higher dimensionality. This phenomenon is the counterpart of the curse of dimensionality in multi-dimensional indices.

Comparison with MVP-tree and GNAT

We compared the performance among MB+tree, MVP-tree and GNAT. Since there is no description of k-nearest neighbour search in [2] and [6], extensive experiments were done on r-neighbour search, counting only the number of distance calculations, since GNAT and MVP-tree are memory resident indices.

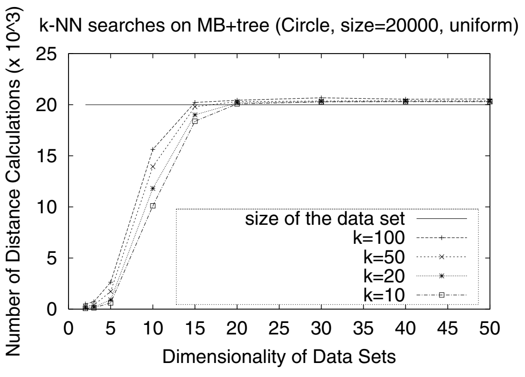


Fig. 15. k-NN Searches on Uniformly Distributed Data Set

- In MVP-tree, the number of vantage points and the number of partitions are used as parameters. In our experiments, we measured the performance in cases of (2,2), (3,2), (3,13) and (3,80), respectively. Here, the first number and the second number in a pair are the number of vantage points and the number of partitions. The parameters are chosen because they are reported as to give better performance results in the literature.
- In GNAT, the number of vantage points (called *division points* in the literature [6]) can be specified. We use 50 and 100 in our experimental studies.

Fig. 16 shows one of the results (dimensionality is 10 and the data set size is 20,000). For MVP-tree, only the best result (2,2) was plotted in the figure, and for GNAT results for 100 division points was omitted because it was worse than that of 50. Generally, the number of distance calculations of MB+tree with the circle method is almost same as that of MVP-tree(2,2). Fig. 17 shows other

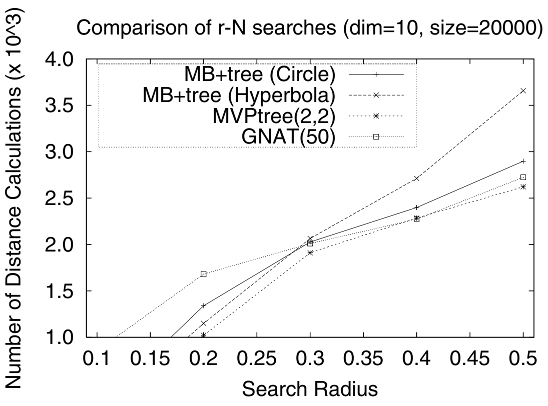


Fig. 16. Comparison among MB+tree, MVP-tree and GNAT

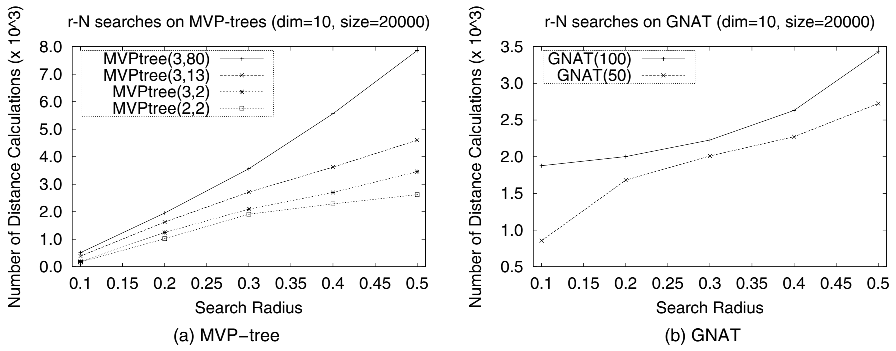


Fig. 17. r-N Searches on MVP-tree and GNAT

results of MVP-tree and GNAT with varied parameters.

Comparison among Construction Methods

MB+tree can be constructed by dynamic insertions. We compared the performance among three kind of construction method, the optimal MB+tree, MB+tree with random order insertions, and MB+tree with skewed order insertion. In skewed order insertion, data are inserted cluster by cluster in order. Recall that a cluster is a set of points generated with Gaussian distribution around the centroid of it. Fig. 18 shows the results of 20,000 points in 10 dimensional vector space. From Fig. 18, we can see that random insertion can even outperform the optimal MB+tree, whereas skewed order insertions cause performance degradation. This is due to an unbalanced block tree. It is worth noting that the circle method shows better tolerance to skewed order insertions.

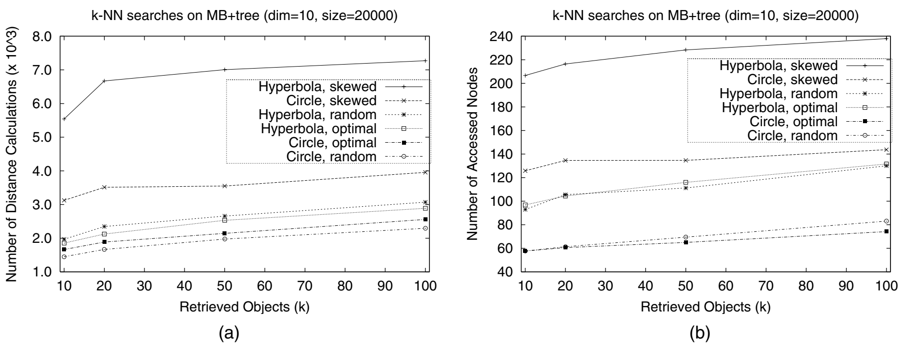


Fig. 18. Comparison of Construction Methods on MB+tree

6 Conclusion

In this paper, we proposed MB+tree which is a dynamically updatable page-based metric index. Its data structure and operations are discussed in detail. The unique feature of our approach is a) to convert a similarity query into a range query, and b) use MB+tree, as an extension of B+tree to support such range queries. For doing so, the key issue is how to generate keys for objects in a generic metric space. We proposed two approaches, namely, the hyperbola method and the circle method. Our result confirms that the circle method outperforms the hyperbola method. We have conducted extensive performance studies. Our experimental results show that MB+tree is competitive with MVP-tree and GNAT in the number of distance calculations at search time. Recall that both MVP-tree and GNAT are constructed when all data are available. We also showed that an MB+tree with random insertions can also achieve the similar performance like the corresponding optimal MB+tree. The main advantage of MB+tree is to be able to store data on disk and to allow dynamic updates.

References

1. A.K.Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1998.
2. Tolga Bozkaya and Meral Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. *Proceedings of the ACM SIGMOD Conference on Management of Data, Tucson, Arizona, May 1997*, pages 357–368, May 1997.
3. Masajiro Iwasaki. Implementation and evaluation of metric space indices for similarity search. *IPSJ Transactions on Databases, in Japanese*, 40:24–44, Feb 1999.
4. Marco Patella Paolo Ciaccia and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd VLDB Conference, Athenes, Greece, 1997*, pages 357–368, 1997.
5. Divyakant Agrawal Ravi Kanth and Ambuj K. Singh. Dimensionality reduction for similarity searching in dynamic databases. In *Proceeding of ACM SIGMOD Conference on Management of Data, Seattle, Washington, 1998*, pages 166–176, 1998.
6. S.Brin. Near neighbour search in large metric spaces. In *Proceedings of the 21st VLDB Conference, Zurich, Switzerland, 1995*, pages 574–584, 1995.
7. Jeffery K. Uhlmann. Satisfying general proximity/similary queries with metric trees. *Information Processing Letters*, 40:175–179, November 1991.
8. Masatoshi Yoshikawa Yasushi Sakurai and Shunsuke Uemura. High-dimensional nearest neighbour search based on virtual bounding rectangles. In *Proceeding of International Conference on FODO'98*, pages 258–267, 1998.

An Information Store and Retrieval Facility on CORBA[#]

Winston Lo[1], Yue-Shan Chang[2*], Chii-Long Chou[3], Ruey-Kai Sheu[2], and Shyan-Ming Yuan[2]

[1]Department of Computer and Information Science
Tung Hai University, Taichung, Taiwan
winston@mail.cis.thu.edu.tw
[2]Department of Computer and Information Science, National Chiao Tung University
Hsin-Chu, Taiwan 31151
ysc@mhit.edu.tw
{rksheu,smyuan}@cis.nctu.edu.tw
[3]Department of Computer and Information Engineering
Feng Chia University, Taichung, Taiwan

Abstract. In this paper, we propose an Information Storage and Retrieval Facility based on the CORBA. This facility provides information search, retrieval and storage service. To define these services, we referred the Z39.50 and defined a set of interfaces by CORBA IDL. In addition, other information storage respect services, such as information update, information update notify, and information version control etc., and more other services, such as initial service, close service, and search command forward and distribution etc., were provided in this facility. Based on this facility, programmer can add the capability of query Z39.50 server into applications and also can integrate various data sources and provide an uniform interface. Finally, we also implement an experimental library management system to verify our design.

1 Introduction

Z39.50 is a well-known standard of information retrieval protocol defined by the NISO (National Information Standards Organization), and widely used in information retrieve applications and library systems [1,2]. This protocol specifies a set of query syntaxes and attributes, and it works on the basis of the protocol TCP/IP, To implement such protocol based on TCP/IP or WWW+CGI for distributed environment need more efforts on concerning information transfer over the network. Programmer must take care the session and presentation semantics of query requests cross the network.

In this paper, we refer the Z39.50 protocol version 3 and define a set of Information Storage and Retrieval Facility (ISRF) interfaces based on the Common Object Request Broker Architecture (CORBA) [8]. These interfaces are compatible with Z39.50

[#] This work was supported by National Science Council under Contract No. 87-2213-E-035-008.

[*] He is also the lecturer of the Department of E.E. of Ming Hsing Institute of Technology

standard and provide information search, information retrieve, access control, accounting control, and resource control etc.; so that programmers can write programs to query information source via ISRF.

There are two motivations in our work. First, defining and creating a software component that based on a de-facto standard, general applications need to retrieve information on distributed environment or Internet can utilize this component. Because ISRF is based on the open standard in distributed object-oriented environment, programmer's efforts on the information retrieval over distributed environment or Internet can be reduced by this software component. Next, this architecture can apply to integrate various information sources based on the agent-based technique easily [3]. Due to the ISRF provides an uniform interface, system developer that needs to integrate various sources can implement their wrapper for specific information source on the CORBA in ease way.

Here we give a brief description of two well-known information search protocols and products that are designed and implemented by object-oriented paradigm.

ZORBA [4] is a technology of information retrieval using distributed object. It defines a set of interfaces that only includes information query by CORBA IDL. But, in the increasingly complicated applications, it is not enough that only use the simple search and retrieval mechanisms. Many mechanisms are also essential, such as accounting control, security control, and information update control etc. These mechanisms are not supported in the ZORBA. Our system has involved most of services of Z39.50 protocol.

The STARTS [1] is attempting to develop an informal IR standard. The STARTS developers have proposed a simple IR protocol that uses a highly cut down query structure and attribute set from the Z39.50 protocol and uses a simple existing text encoding to enable simple transport via the Internet. This standard is essentially aimed at solving immediate problems and will not cope with complex searches for non-document objects. By having no real compatibility with the Z39.50 standard, there is no opportunity to use sophisticated features of Z39.50 in the future.

2 Overview of Information Storage and Retrieval Facility

2.1 System Architecture

Z39.50 is a de-facto standard on information retrieval, that provides information search, information retrieve, access control, accounting control, and information control etc. We design the ISRF referring Z39.50 protocol, and most of interfaces are compatible with Z39.50. The ISRF consists of *Explain*, *Search*, *Retrieval*, *Storage*, *Resource Control*, *Access Control*, and *Accounting Control* etc. The relationship of these services in ISRF is shown as Figure 1. All of these services are defined in the CORBA IDL.

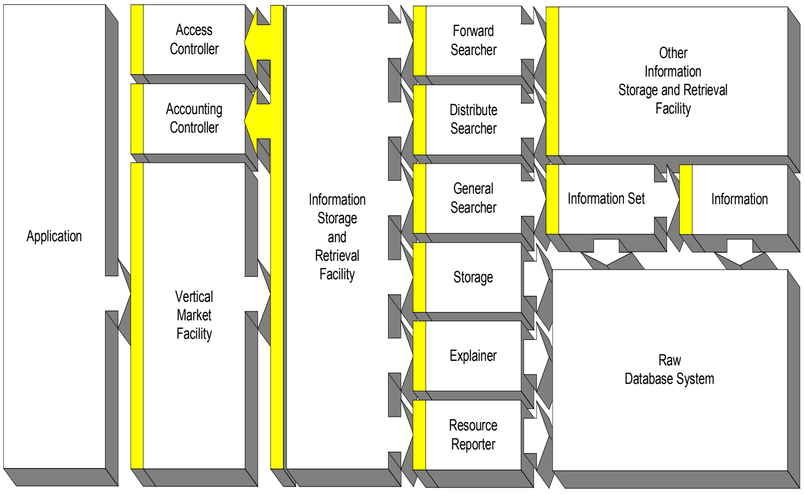


Figure 1. Architecture of Information Storage and Retrieval Facility

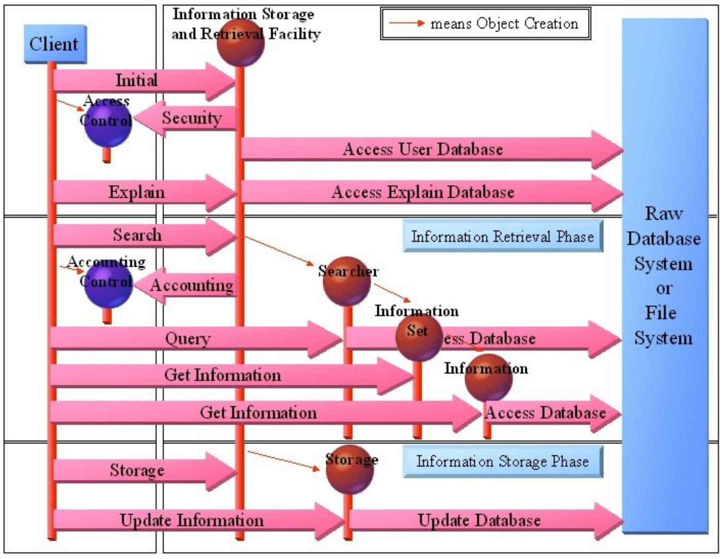


Figure 2. Scenario of Information Storage and Retrieval Facility

In order to increase the system scalability, an ISRF server can cooperate with other ISRF server via *Forward* and *Distribute* mechanisms. Other ISRF servers could dynamically join the system. When client issues a search request, ISRF server will create an information set to place the retrieved information. Information set involves some retrieved information.

2.2 Scenario of Information Storage and Retrieval Facility

Figure 2 shows the normal scenario of ISRF. From the viewpoint of a client program, it first binds to ISRF and invokes *initial()* method to create a connection between client and server. Server checks access ability of client by CORBA's callback mechanism. If check is ok, client therefore can execute information search, retrieve, and store operation.

The client program then gets some resources from ISRF in next step. These resources involve *Explainer*, *Searcher*, *Storage*, and *ResourceController* etc. These resources are used to search, retrieve and store information as described above. When a client issues *GetSearcher()* operation to ISRF, the ISRF create a *Searcher* object that can be used to query underlying database. Next, the client program can query information source via invoking *Query()* operation and get the Information Set via invoking *GetInformationSet()* operation. Then, the client can invoke *GetInformation()* operation to get *Information* object or invoke *CreateIterator()* operation to get the *Information Iterator*.

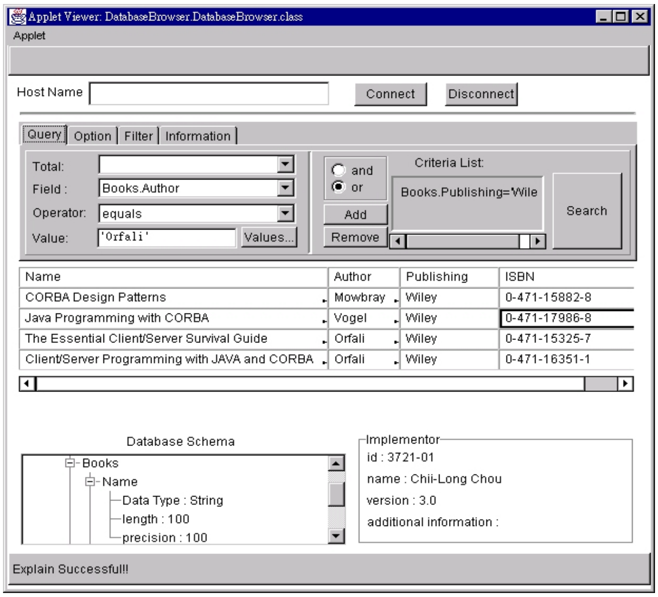
3 Application of ISRF

3.1 An Experimental Library Management System

In order to demonstrate the capability of the ISRF, we develop an experimental Library Management System based on the ISRF. Users can query and borrow books online and library manager can manage database on-line. Programming this system based on ISRF, the programmer only concerns the user interface and library system structure.

Beginning construct a library management system, we define a library management system interface (named Library). This interface has five methods: *Login(), Search(), Borrow(), NewBook(), NewUser()* respectively. This interface is based on ISRF and focus on the library management. All methods of the Library utilize the ISRF. In this interface, *Borrow()* and *Search()* are dedicated to user, *NewBook()* and *NewUser()* are for library manager.

By similar way, the interface of library management system must be compiled using CORBA's IDL compiler to generate client stub and server skeleton. We then implement client-side program and server-side object. In order to integrate the Library Management System with the increasingly popular WWW, user interface is implemented as a JAVA applet in our system. Figure 3 shows the query result of library management system.


Applet Viewer: DatabaseBrowser.DatabaseBrowser.class

Applet

Host Name [ ] Connect Disconnect

Query | Option | Filter | Information

Total:

Field : Books.Author

Operator: equals

Value: 'Orfali' Values...

( ) and [✓ or]

Add

Remove

Criteria List:

Books.Publishing=Wile

Search

Name	Author	Publishing	ISBN
CORBA Design Patterns	Mowbray	Wiley	0-471-15882-8
Java Programming with CORBA	Vogel	Wiley	0-471-17986-8
The Essential Client/Server Survival Guide	Orfali	Wiley	0-471-15325-7
Client/Server Programming with JAVA and CORBA	Orfali	Wiley	0-471-16351-1

Database Schema

```
⊟-Books
   ⊟-Name
      ├Data Type : String
      ├length : 100
      └precision : 100
```

Implementor

id : 3721-01

name : Chii-Long Chou

version : 3.0

additional information :

Explain Successful!!

Figure 3. Client side GUI of library management system

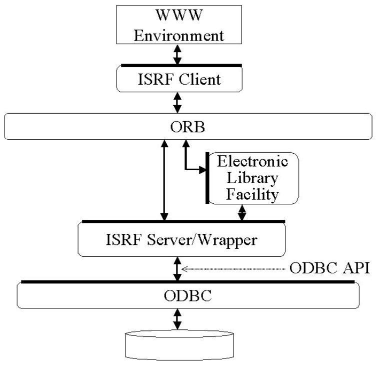


Figure 4. Library Management System Architecture

3.2 Implementation Issues

Figure 4 shows the architecture of Library Management System. In our implementation, the Library and the ISRF are implemented as CORBA objects. The ISRF client can invoke the methods of Library and ISRF when it accesses the backend database. On the other hand, this system uses Microsoft's ODBC [5] to connect to general raw file system or databases. The ODBC not only defines SQL syntax, but also defines the application interfaces between programming language and SQL. Therefore, when the

ISRF server receives a request from ISRF client, it has to dispatch this request to backend database and to wrap up the ODBC API in ISRF server.

In addition, this library management system applies multi-thread technology to CORBA client and server for improving system performance. Each request to server will make the ORB to create a thread to serve this request. In order to avoid data inconsistency among multi-clients requests, transaction processing and concurrency control on the CORBA [6] must be applied to this system.

4 Conclusions

In this paper, we propose an Information Storage and Retrieval Facility based on the CORBA. This facility provides information search, retrieval and storage service. To define these services, we referred the Z39.50 and defined a set of interfaces by CORBA IDL. In addition, many important information storage respect services, such as information update, information update notify, and information version control etc., and more other services, such as initial service, close service, and search command forward and distribution etc., were provided in this facility. Besides, an experimental digital library system based on this facility was implemented in our project to demonstrate how to use this facility.

References

1. Gravano L. et al.: STARTS: Standford Protocol Proposal for Internet Retrieval and Search, http://www-diglib.stanford.edu/diglib/WP/PUBLIC/DOC104.html.
2. Tomasic A. et al.: The Distributed Information Search Component (Disco) and the World Wide Web, SIGMOD'97, AZ, USA, 1997.
3. Laufmann S.C.: Towards agent-based software engineering for information-dependent enterprise applications, IEE Proc. Software Engineering, 144(1), (1997)38-50.
4. Ward N. et al.: ZORBA: Information Retrieval Using Distributed Object Technologies, EGOEO'98, 1998. http://www.dstc.edu.au/ZORBA.
5. Tomas: Inside ODBC API 3.0, Microsoft Press, 1995.
6. Liang K.C. et al.: Nested Transaction and Concurrency Control Services on CORBA, in Proc. of Joint International Conference on Open Distributed Processing (ICODP) and Distributed Platforms (ICDP), Toronto Ontario, Canada, pp. 27-30, May 1997.
7. Saleh K. et al.: The distributed object computing paradigm: concepts and applications, The Journal of Systems and Software 47, (1999)125-131.
8. Object Management Group, Inc: The Common Object request Broker (CORBA): Architecture and Specification, v2.2, February 1998.

Retrieving Content Directly from Antique Book Images*

Zhifeng Chen, Yong Wang, Liang Zhang, and Baile Shi

Computer Science Department, Fudan University
220 Handan Road, Shanghai 200433, P.R. China
{zfchen,wangyyong,zhangl,bshi}@fudan.edu.cn

Abstract. The digitization and utilization of antique books are of significance. WWW lays a foundation for the dissemination of images, but more comprehensive mechanisms, such as content-based retrieval, are necessary to reduce information overload. It is well known that the content retrieval of a Chinese antique book directly from its bitmap is very hard due to intrinsic handwritten nature. In this paper, an original method for content retrieval based on visual similarity is proposed. By extracting features from book images, the method makes up a feature space and applies spatial indexing on it. A range searching strategy is then applied to retrieve all analogs to the query example. The prototypical implementation demonstrates the feasibility and supports both retrieving and browsing via popular Web browsers.

1 Introduction

Like other civilization legacy, antique books are rare and precious resources. For China, a country with long history and splendid culture, the connotation above sticks out acutely. However, raw digitized antique books are voluminous, their transmission in WWW is prone to block networks and overwhelm end-users. More comprehensive facilities, such as Optical Character Recognition (OCR) or searching mechanism are needed to reduce the bandwidth requirement.

Many projects are seeking to make use of rare and precious antique books [3,4,7]. They fall into two categories: *cataloging method* and *attached-text method*. In the cataloging method, which is widely used in library practices, some entries for a book, such as the book name, authors, publishing information, are manually produced. A user has to rely on them to retrieve information. A major conflict with the method is the limited retrieval points and unforeseen specific query purposes. On the other hand, the attached-text method constructs a text file by OCR for a specific antique book [3,7]. An index is then built on this textual file and full-text retrieval technology is applied. However, in the context of Chinese antique books, this method does not work well due to the

* This work is a sub-project of the grant from Natural Science Foundation of China for the *Research on Key Techniques in Digital Library* project (No. 69933010). One of authors is supported by the “Chun-Tsung Scholar” Foundation.

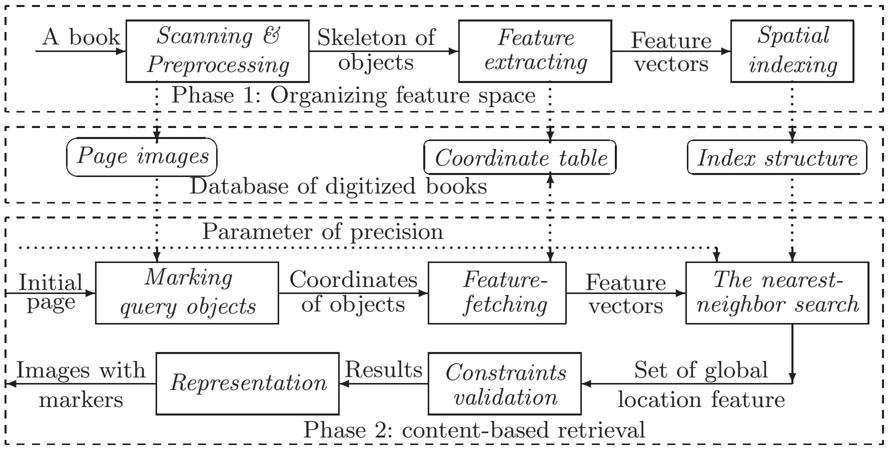


Fig. 1. Flowchart of contents retrieval by visual similarity criterion

brush-written holograph nature. Its other problems are the identity judgement between page images and the textual data, the size of lexicon and its coverage. In addition, *TONG JIA ZI* (interchangeable words or characters) might disable full-text retrieval. It is necessary to break a new path for content-based retrieval.

This paper is organized as follows: Section 2 gives the framework and its rationale. Section 3 elaborates on vital techniques we developed. Section 4 describes our prototype. Finally, we present the conclusion in Section 5.

2 Methodological Description

We believe that content-retrieval of Chinese antique books should proceed on the visual semantics of objects rather than on the linguistic semantics. This strategy enables full-text-like queries directly on page images and clears up previous problems. It can reduce the network overload by discarding irrelevant information and preserve the pleasure of reading as well.

In proposed method, we regard every virtual character or symbol, called an *object*, as an image that is a chunk of relatively clustered pixels in a page image, so the page image is composed of many such smaller images arranged in a certain order. As illustrated in Figure 1, the method consists of two phases: feature space construction and content-based retrieval. The former prepares a database for the latter. More specifically, during the *scan & preprocess* module, an original antique book is digitized into a series of page images. Further, every object is separated and is processed by thinning algorithm. Accessory functions in this module include layout alignment, noises elimination, and binarization. They are common techniques in Chinese OCR. Then, the *feature extraction* module extracts content features for every object skeleton, and feeds them into the high-dimensional *spatial indexing* module where the object is favorably organized by its morphological feature. This process continues until all objects in the antique book are dealt with and a global index structure is established.

From now on, content-based retrieval can work repeatedly. This phase is actuated by a page image identified by a given page number or other means (e.g. cataloging method). While browsing page images, a user marks objects as he/she wishes. Marked objects and their order form the query sample. Coordinates of marked objects are supplied to the *feature-fetching* module where features of each marked object are obtained by looking up the *coordinate table* [1]. The order alone also stands for constraint for the purpose of validation. Now, *the nearest-neighbor search* module can find analogs to each object in the query sample. It produces a collection of analog sets for further process. The *constraint validation* module takes this collection and previous constraint as its input, checking combinatorial validity. The *Representation* module receives valid results from its precursor and marks proper objects on page images.

In this way, the intricate retrieval is decomposed into four tractable consecutive sub-problems: feature extraction, index structure construction, the nearest-neighbor search and constraint validation.

Because the query sample comes directly from page images that act as a stylebook, the problems of identity, character set, special symbols, interchangeability of characters and the lexicon no longer exist. Through approximate matching, we avoid the difficulty which the *attached-text method* brings about, i.e. the obstacle of concept formation.

Another advantage of this method over the attached-text method is that the decision of recall/precision balance is delayed until the retrieval moment, and can be easily controlled. In the attached-text method, the meanings of objects are stiffened at the OCR stage, i.e. a decided linguistic semantics (Chinese character) must be assigned to an object. A user has no authority to re-mapping them at later retrieval stage. In the context of antique books, the re-mapping becomes more desirable because there are various factors that disturb the mapping.

Our method is coherent with the cataloging method that is extensively used in libraries. Guided by catalogs, users can find out a specific volume of ancient books. Then, content-based information retrieval is ready to help the user find targets in the book.

To summarize, the main contribution of our work is the original approach to the content-retrieval problem in the context of Chinese antique books, and some key techniques supporting the method.

3 Key Techniques

3.1 Extracting Feature

Three basic categories of feature, say the *global location feature*, the *page feature*, and the *morphological feature*, are combined to describe an antique book. An-

[1] The coordinate table is a set of triples (Page-number, Coordinate, Feature-vector)

other feature, *book feature*, should be defined if we bound several antique books into one processing unit. In our prototype, we only elaborate upon the basic ones.

Definition 1 (global location feature, GLF). *The GLF of an object in an antique book is the sequential order number of the object within page images of the book.*

Recall that objects are correctly separated in the *scan & preprocess* module. We can identify each object according to reading habit (usually page by page and in each page from the right to the left, then from the top to the bottom). In the case of complicated layout, we can first use a recursive-curve (e.g. Hilbert's) to scan areas, then serialize objects by conventional means. Hence, a GLF uniquely identifies an object.

Definition 2 (page feature, PF). *The last GLF on each page defines the PF of an antique book.*

Definition 3 (morphological feature, MF). *The MF for an object is formed by its accumulative stroke statistic within centroid-based multi-level partitions.*

We use the MF to describe the visual semantics of an object. This is based on our observation that the configuration and componential proportion of a Chinese character remain stable [6], yet various factors, such as disparate strokes, illegible and disturbed strokes, and variety in length call for error tolerant methods of feature extraction.

Before extraction, the bitmap of each object is normalized by a MBS (Minimum Bounding Square). The object is centered within a square whose side is the maximum of width and height of its foreground. Compared with conventional MBB (Minimum Bounding Box), MBS keeps the Width/Height information.

Next, the MBS is partitioned hierarchically according to the centroid of foreground. We select centroid as the splitting point owing to above observation. It is tolerant of the change of stroke in one or two.

Finally, our method extracts stroke elements within each partition to construct the feature vector. Stroke element is defined as a segment of typical strokes, namely, Heng (two adjacent pixels horizontally), Shu (two adjacent pixels vertically), Pie(three adjacent pixels diagonally), and Na (three adjacent pixels anti-diagonally). Extracting stroke elements rather than strokes themselves facilitates the uniform processing of characters and symbols. It is also tolerant of stroke variations.

Since Heng and Shu appear more frequently than Pie and Na [6], it is adequate to reduce the spatial dimension through the multiform partition. In our prototype, partition is applied only one level to Pie and Na while two-levels to Heng and Shu. So, feature vectors of an antique book spans a 40-dimensional vector space. The feature vector f can be formulated as:

$$\begin{aligned} f(i) &= \sum_{1\le k\le i} \frac{h(k)}{p_2(k)}, & f(i+16) &= \sum_{1\le k\le i} \frac{s(k)}{p_2(k)},\ i=1,2,\ldots,16; \\ f(j+32) &= \sum_{1\le k\le j} \frac{p(k)}{p_1(k)}, & f(j+36) &= \sum_{1\le k\le j} \frac{n(k)}{p_1(k)},\ j=1,2,3,4. \end{aligned} \tag{1}$$

where, $p_1(k)$ and $p_2(k)$ are respectively the number of foreground points of one and two levels partition. While $h(k), s(k), p(k), n(k)$ are foreground points of Heng, Shu, Pie and Na stroke elements, respectively, within partition area k. The feature vector describes the layout of stroke elements within an object.

3.2 Indexing Feature Space

All MFs make up a vector space, in which each MF contributes to the location of a point. The point contains the GLF[2]. We use Euclidean distance in our prototype. Generally, points of visually similar objects are adjacent and those of visually distinct are faraway in the feature space. Analogs to a given sample can be obtained by comparing the distance between a point and the sample.

To accelerate the comparison, a spatial index structure is designed to organize feature vectors. In principle, all the spatial index structures [2] contribute to efficient indexing. However, the *dimension cruces* demand efficient algorithms. In the prototype, we choose the PK-tree for its better performance in terms of storage and query speed [5]. For further information about constructing a PK-tree, please refer to [5].

3.3 Querying Similar Objects

When the user issues a query containing several objects, the proposed method decomposes it into separated objects and looks for analogs via index structure for each of them. The nearest-neighbor search of PK-Tree [5] returns a collection of sets of GLFs.

During the process, the searching range which influences the recall/precision is decided by a parameter ϵ. In the prototype it spans 11 levels. Level 0 ($\epsilon = 0$) means strictest matching and level 10(ϵ=1) means the most relaxed query. Between the two extremes, ϵ is adjusted by step 0.1. Because the user can get feedback immediately, a satisfactory balance between recall and precision will be reached.

3.4 Validating Constrains

The constraint is the relative order of objects that the user chooses as the query sample. It is applied to the result of previous phase to filter out improper combinations. What are left is a set of GLFs of the first objects. *Constraint validation* can be formulated as:

```
1. First, assume that the query consists of $m$ objects, and $m$ lists of GLFs are
   obtained by *the nearest-neighbor search* module, denoted as $L_1, L_2, \ldots, L_m$;
2. Assign $L_1$ to $\mathbf{L}$
3. $i$ iterates from 2 until $m$, do
   3.1 For each $e \in \mathbf{L}$, suppose its GLF is $j$. If there is no element of $L_i$ whose
       GLF is $j + i - 1$, delete $e$ from $\mathbf{L}$;
4. Return $\mathbf{L}$
```

[2] PFs are maintained separately for the sake of memory efficiency.

4 Prototypic Implementation

To verify the feasibility of proposed method, we implement a prototype in the WWW environment. It consists of the server-side services and the client-side interface. The client-side interface can run on Java-enabled browser. It supports some operations, such as Zoom-In/out, Object Marking/Retrieving, Precision control. The server-side services are made up of a three-layered organization, i.e., a web server, static HTML files, and Java servelet. The static files provide fixed information such as registration, book-selecting information. Java servelet is a daemon that handles the user requests, invokes file & database retrieval module to obtain data and produces dynamic HTML files. The third layer includes the file & database retrieval daemon and the database of antique books.

5 Conclusion

An original method of content retrieval for Chinese antique book is proposed. By vital techniques we developed, it extracts content features of a Chinese antique book and constructs spatial index structure to accelerate the searching process. A user can make the recall/precision balance at retrieval moment, and the constraint validation ensures the precision of query results. The prototype demonstrates the feasibility of the method, and shows the superiority of visual similarity criterion to conventional methods. The method can be improved further by introducing dimension reduction mechanism, such as FastMap [1].

References

1. Faloutsos,C., Lin,K.: FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets, In Proc. SIGMOD'95 (1995) 163–174 385
2. Gaede,V., Günther,O.: Multidimensional Access Methods. ACM Computing Surveys, **30**(2) (1998) 170–231 384
3. Gladney,H., Mintzer,F., Schiattarella,F.: Safeguarding Digital Library Contents and Users: Digital Images of Treasured Antiquities. D-Lib Magazine. (1997)July/August *http://www.dlib.org/dlib.html* 380
4. Thibadeau,R., Benoit,E.: Antique Books. D-Lib Magazine (1997) September *http://www.dlib.org/dlib.html* 380
5. Wang,W., Yang,J., Muntz,R.: PK-tree: a spatial index structure for high dimensional point data. In: Proc. of 5th intl. Conf. on Foundations of Data Organization (FODO'98) (1998) Kobe Japan, November 12–13, 1998. *http://dml.cs.ucla.edu/projects/oasis/PK-Tree/pk-tree.html* 384
6. Zhang,X: Chinese Character Recognizing Techniques (in Chinese). Tsinghua University Press, Beijing (1992) 383
7. Zhu, Y.: Experiences of Electronic Version of SiKu Quan Shu (Complete Library of the Four Branches of Literature. The Journal of the Library Science in China (in Chinese) 6 (1999) 82–84 380

Schema Based Data Storage and Query Optimization for Semi-structured Data[1]

QinKe Wang[1], LiZhu Zhou[2]

[1] Department of Computer Science and Technology, Tsinghua University,
Beijing, 100084,
China, People's Republic
wangqk@263.net
[2] Department of Computer Science and Technology, Tsinghua University,
Beijing, 100084,
China, People's Republic
dcszlz@mail.tsinghua.edu.cn

Abstract. Semi-structured data is generally modeled as labeled graphs. Data in such models is self-describing and dynamically typed, and captures both schema and data information. Such models, although flexible, evoke severe efficiency penalties compared to querying structured database, such as relational databases. In order to improve the efficiency of data manipulation by utilizing structure information, we propose a technique to rearrange semi-structured data according to its schema and to store data in simple relations. With the help of schema based data storage, queries for semi-structured data can be evaluated by relational operations. The optimization effect is also inspected in this paper.

1 Introduction

Recent research has focused on data with irregular, unknown, or frequently changing structure; such data is called semi-structured data [1][5]. A typical site on the World Wide Web will be a good example. The data is generally modeled as labeled graphs [2][4], and several query languages [2][4][10] based on such models have been proposed. All of them allow some kind of recursion to traverse arbitrarily long paths, typically in the form of regular path expressions.

Semi-structured data is self-describing. That is, the schema is embedded with the data, and no a priori structure is assumed. This gives us flexibility to process any data, and we can deal with changes in the data's structure seamlessly. But this extreme view has a few drawbacks: data is inefficient to store, since the schema needs to be replicated with each data item; queries are hard to evaluate efficiently: even a simple regular path expression may require the entire graph to be traversed.

Researchers have looked for ways to describe and exploit regularities in the data's structure, without having to fully specify a rigid schema ahead of time. Examples are

[1] National Science Foundation (69773027)

graph schemas [3], data guides [9], description logics [6]. In this paper, we propose a technique to rearrange semi-structured data according to its structure and to store data as simple relations. The semi-structured data is divided into several relations: some of them contain internal edges and others contain meaningful atomic values. Most relations are statically typed. The data rearrangement is based on paths from the root to data in the graph database, so it is well suited for query languages that access data by regular path expressions. Exploiting schema based data storage, queries on semi-structured data can be evaluated by relational operations. We show that this method will significantly optimize query execution and provide a good chance to apply relational query optimization techniques to semi-structured data.

1.1 Related Work

Our contribution extend initial work presented in [9], which gives a formal definition of data guides and provides simple algorithms to build data guides and keep them consistent when the underlying database changes. The technique of data storage in this paper exploits structure information described by a data guide like schema.

Other related theoretical research is presented in [7], which discusses storage schemas for graph-structured database. The paper introduces the notion of storage schema, which is a technique allowing a semi-structured database to be stored as a collection of structured relations. This method is closely related to graph schemas [3], which are described about the data's structure ahead of the actual data. Hence data storage and query optimization depend on having a database conform to an explicitly specified schema. In contrast, our work focuses directly on the case where it is inconvenient to specify and maintain a schema: our data storage are dynamically generated and maintained to always represent the current state of the database.

Other researchers have also considered schema-based query optimizations. [8] describe a technique called query pruning to restrict navigation to only a fragment of the data, based on partial knowledge about the data's structure specified by graph schemas. But query pruning has only been described for single path queries and a restricted kind of schemas.

2 Foundations

2.1 Data Model

Semi-structured data is best modeled by edge-labeled graph described in [4]. A ***graph database*** DB, is a rooted, edge-labeled graph, with labels from an infinite universe **Label = Int** $\cup$ **String** $\cup$ … We assume that each node is accessible from the root and may be referred by an unique ID in DB. We denote the DB by its root node. Each graph can be viewed as a set of label-graph pairs: $\{ l_1 ! \ \ t_1, l_2 ! \ \ t_2, \ldots l_n ! \ \ t_n \}$, where $l_1, \ldots, l_n \in$ **Label** and $t_1, \ldots, t_n$ are other graphs. Fig. 1 depicts a graph database DBGroup for an example Web site of some DB research group. In general, DB may contain shared nodes and cycles.

Edge-labeled graph provides a simple but general model to represent widely varied data formats. In practice, for example XML files, meaningful data usually appears on the lowest edges. Hence, here we assume labels on ***internal edge***s have a single type **Symbol,** and that only ***leaf edge***s are labeled with meaningful atomic values.

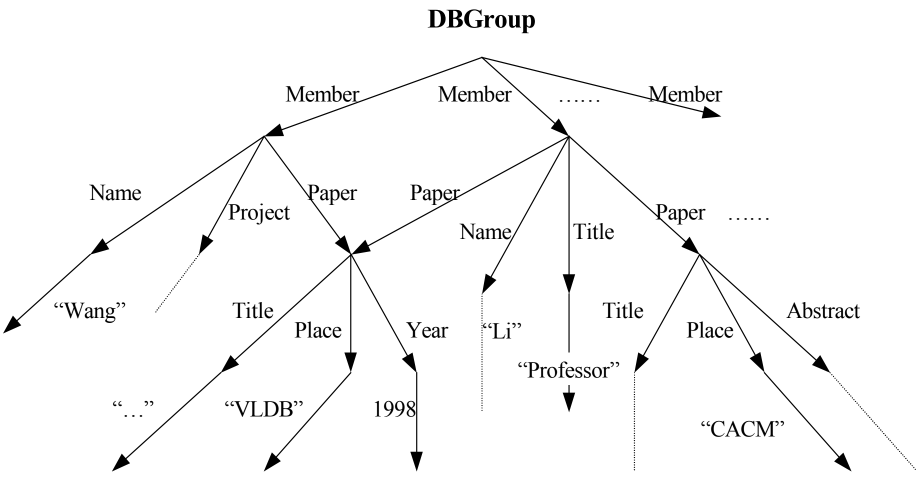


Fig. 1. An example for edge-labeled graph

Next, we give several simple definitions and notations useful for describing a semi-structured database and subsequently for specifying our technique of schema-based data storage.

Definition 1. A ***data path*** of node t is an alternating sequence of nodes and labels of the form $t.l_1$! $t_1.l_2$! $t_2.l_3$! … ! t_n such that we can traverse from t a path of n edges e_1 , e_2 , … , e_n through n nodes t_1 , t_2 , … , t_n where edge e_i has label l_i.

Definition 2. A ***label path*** is a sequence of arrrow-separated labels l_1 ! l_2 … ! l_n, such that we can traverse a path of n edges e_1 , e_2 , … , e_n from node t where edge e_i has label l_i.

Definition 3. A data path d ***match***es a label path l if the sequence of labels in d is equal to l, and we denote the set of data paths from node t that matches l by t.l.

Definition 4. A ***target set*** is the set of all nodes that can be reached by traversing a given label path l of t, denoted by TS(t.l).

Definition 5. A ***target-edge*** of a label path l is the last edge e_n reached by a data path d that matches l. And a ***target-edge set*** is the set of target-edges of l, denoted by TES(t.l).

2.2 Query Language

We describe our query optimization technique in terms of a query language with regular path expressions. A ***regular query*** Q is described by the grammar:

Q ::= (*select* Q *where* C, … ,C) | var
C ::= R !   var *in* var
R ::= **Pred** | (R !   R) | (R | R) | R*

All variables denote nodes in the data graph; DB is the variable denoting the root and ε indicates a empty graph. A ***condition*** C of the form " R !   x in y " means that there exists a path from node y to node x whose labels match the regular path expression R. Note that regular path expressions permit predicates on edge labels. The result of a regular query " Q = select Q' where …" applied to a database DB is a set of nodes in DB obtained as follows. Considering all substitutions of Q's variables with nodes from DB that satisfy the *where* conditions, evaluate Q' to obtain a set of nodes.

2.3 Schema

Our storage and optimization techniques exploit information about data structure described by a ***schema***, which is similar to the concept of data guides in [9]. A schema provides a concise, accurate and convenient summary about the structure of a semi-structured database. Fig. 2(a) represents the schema of the semi-structured database DBGroup above.

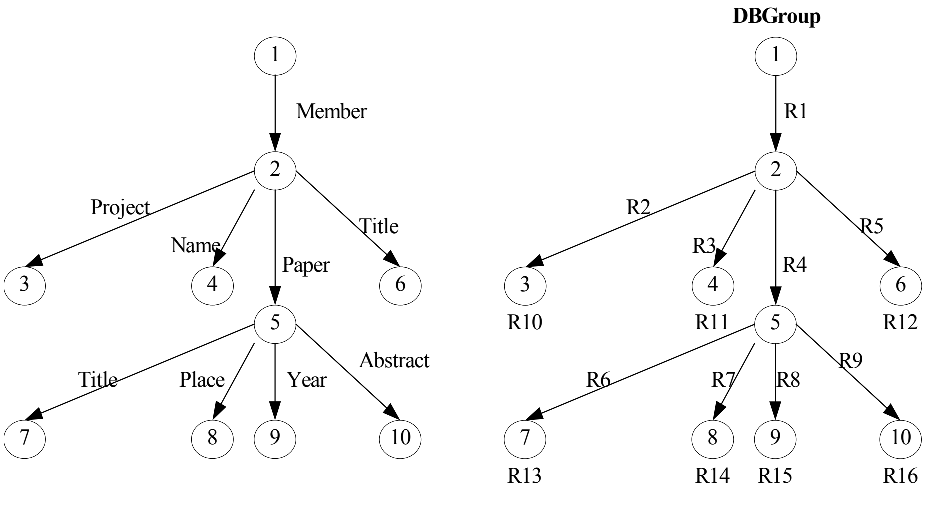


Fig. 2. (a) Schema for DBGroup (b) Schema-based data storage

The schema s of a database t is defined according to the following conditions:

1. Every label path of t has exactly one matching data path in s, and every label path of s is a label path of t;
2. Given a label path l, TS(t.l) and TS(s.l) are the target sets of l in t and s, respectively. Let L(t.l) is the set of all label paths in t that share the same target set as l, this is, L(t.l) = { m | TS(t.m) = TS(t.l) }. Similarly, let L(s.l) = { m | TS(s.m) = TS(s.l) }. Then for all label paths l, we have L(t.l) = L(s.l).

The first condition reduces all data paths matching the same label path in the source database to exactly one data path in its schema; the second condition prohibits the existence of multiple graphs satisfying Condition 1 and finally induces a straightforward one-to-one correspondence between source target sets and schema nodes. For example, node 2 of the schema in Fig. 2(a) corresponds to all member nodes in DBGroup. Using such schemas, we can check whether a given label path of length n exists in a source database by traversing at most n nodes in its schema. Beyond summarizing the structure of a source, the schema of semi-structured database also provides key benefits afforded by conventional schemas, such as guidance to the user for query formulation and guidance to the query processor for query optimization, if we annotate each node of the schema with appropriate additional information about the corresponding target set.

3 Schema Based Data Storage

3.1 An Example

We propose a technique for representing semi-structured data according to its structure. To illustrate, assume the external data source for Fig. 3(a) is a relation with null values. Here the leaf edges labeled with meaningful data V1, V2, …, V7 are abbreviated. Then we can describe the graph database's structure using the schema in Fig. 3(b). Information annotated on the leaf nodes indicates types of corresponding atomic values. It is also a graph representation of the source relational schema.

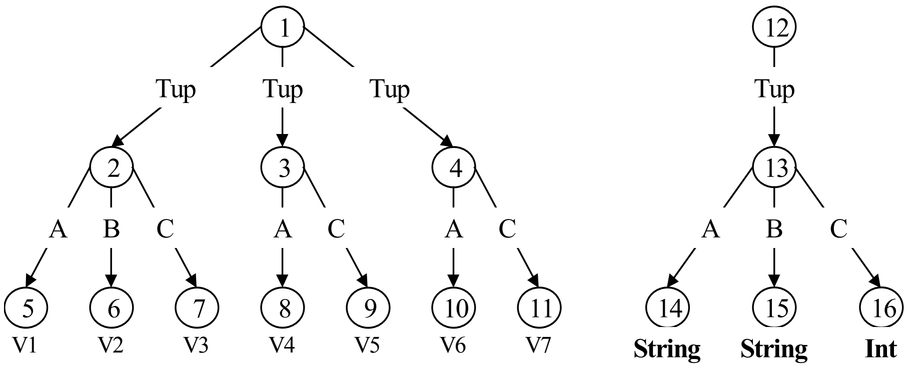


Fig. 3. (a) graph database (b) schema

We push the concept of schema one step further and propose a method for schema based data rearrangement. Labeled graphs modeling semi-structured data can be represented naively as a ternary relation of internal edges E (NodeID, Symbol, NodeID) and a binary relation of atomic values V (NodeID, Label). We divide the edges and values further according to label paths and give the corresponding data storage in Fig. 4. Each of the seven relations R1, R2, … , R6 in Fig. 4 stores a fragment of the original edge and value relations. Relations R0, R1, R2 and R3 divide internal edges, giving node data of corresponding target-edges in the source database for each edge in the schema. Relations R4, R5 and R6 divide leaf edges, and include actual values for each attribute, thus preserving type information for semi-structured data: notice that the new value relations are strongly typed, e.g. R4 and R5 have the type (NodeID, String), and R6 have the type (NodeID, Int).

A

id1	id2
2	5
3	8
4	10

R1

B

id1	id2
2	6

R2

C

id1	id2
2	7
3	9
4	11

R3

Tup

id1	id2
1	2
1	3
1	4

R0

id1	val
5	V1
8	V4
10	V6

R4

id1	val
6	V2

R5

id1	val
7	V3
9	V5
11	V7

R6

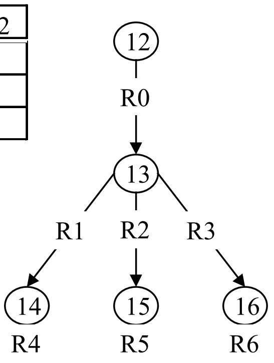


Fig. 4. Schema-based storage for example graph database

3.2 Schema Based Data Storage

First, we will prove that there exists a one-to-one correspondence between source target-edge sets and schema edges. To be simple, special disposal of leaf edges are ignored here.

Theorem 1. Suppose s is the schema for a source t. Given any target-edge set of t, it is by definition the target-edge set of some label path l. Compute TS(s.l) , the target-edge set of l in s, which has a single element e. Let F describe the procedure, which takes a source target-edge set as input and yields a schema edge as output. Then F induces a one-to-one correspondence between source target-edge sets and schema edges.

Proof. We will show that F is (1) a function , (2) one-to-one.

(1) To show F is a function, we prove that for any two source target-edge sets tes1 and tes2, if tes1 = tes2 then F (tes1) = F (tes2). By definition, tes1 is the target-edge set of some label path l, and tes2 the targe-edge set of m, so tes1= TES(t.l) and tes2= TES(t.m). Assume ts1 is the target set of l, and ts2 is the target set of m. If tes1 = tes2, then ts1 = ts2, and l and m are both elements of L(t.l), the set of label paths in t that share TS(t.l). Since L(t.l) = L (s.l), m is also an element of L(s.l). Therefore TS(s.l) = TS(s.m), TES(s.l) = TES(s.m), and their single elements are equal. Hence F(tes1) = F(tes2).

(2) We show that F is one-to-one using the same notation and a symmetrical argument. If F(tes1) = F(tes2), by construction we know that TS(s.l) = TS(s.m), l and m are therefore both elements of L(s.l), and by definition of schemas are also elements of L(t.l). Therefore TS(t.l) = TS(t.m), i.e. tes1 = tes2.

Based on Theorem 1, we describe schema-based data storage as follows:

(1) For each target-edge set of a source database t (leaf nodes and edges ignored), build a relation R of type (NodeID, NodeID), in which each tuple includes node ids of an edge in the target-edge set. And then, label R on the correspondent edge in the schema.

(2) For each target set that includes nodes leading leaf edges, build a relation R of type (NodeID, Label), in which each tuple is comprised of id of such a node and its related atomic values. And then, label R on the correspondent node in the schema. Usually, all values in such a relation have the same type, then the relation could be strongly typed.

We also prove that sequential joins of edge relations along a label path l in the schema give information of actual data paths matching l in the source database. This property is useful for query processing.

Theorem 2. Suppose s is the schema for a source t. Given a label path l of length n in t, there is a single data path b in s matching l. Suppose R1, R2, … , Rn are relations labeled on the edges of b sequentially. Let R = R1 $\bowtie_{id2=id1}$ R2 $\bowtie$ … $\bowtie$Rn. Then each tuple u of R give node ids traversed by a data path d of t sequentially, this is, each tuple determine a data path as the following: let $l = l_1 ! \; l_2 \ldots ! \; l_n$ and u = (t, id_1, … , id_n) , then d = $t.l_1 ! \; id_1.l_2 ! \; id_2.l_3 ! \; \ldots ! \; id_n$. We have, such data paths comprise t.l, the set of data paths from node t that matches l.

Proof.

(1) For n = 1, R = R1 by definition gives node ids of edges in TES(t.l), the target-edge set of l in t. A data path d of t matching l is only determined by an edge in TES(t.l). Therefore, R exactly fixes t.l.

(2) Suppose it is true for n = k.

(3) For n = k + 1. Given any data path d of t matching l, suppost d = $t.l_1 ! \; id_1.l_2 ! \; id_2.l_3 ! \; \ldots ! \; id_{k+1}$, then we show (t, id_1, … , id_{k+1}) is a tuple of R.

It is apparent that data path $t.l_1 ! \; id_1.l_2 ! \; id_2.l_3 ! \; \ldots ! \; id_k$ matches the label path $l_1 ! \; l_2 \ldots ! \; l_k$ of length k. Based on the assumption in (2), tuple (t, id_1, … , id_k) $\in$ R1 $\bowtie_{id2=id1}$R2 $\bowtie$ … $\bowtie$$R_k$. Since d matches l, edge (id_k , id_{k+1}) is a target-edge of l, this is, tuple (id_k , id_{k+1}) $\in$ R_{k+1}. Hence, we have tuple (t, id_1, … , id_{k+1}) $\in$ R.

3.3 Building

Schema-based data storage are easy to build. In a depth-first fashion, we examine the source target sets reachable by all possible label paths. Each time we encounter a new target set ts for some path l, we create a new node o for ts in the schema, add a new edge, create a relation R for the target-edge set of l, and label R on the edge. If we ever see ts again via a different label path m, rather than creating a new schema node, we instead add an edge to the schema such that m will also refer to o, and similarly label the relation for the target-edge set of m on the edge.

It is apparent that the complexity of build a schema-based data storage is equivalent to conversion of a non-deterministic finite automation (NFA) to a deterministic finite automaton (DFA), a well-studied problem. When the source graph is a tree, this conversion takes linear time. However, in the worst case, conversion of a graph-structured database may require time exponential in the number of nodes and edges in the source. Due to shared nodes and circles, a node in the data graph may belong to multiple target sets, namely mapping to multiple node in the schema. It is the same for edges. Hence there will usually be information redundancy to some extent in the rearranged data. Fortunately, according to experiments made in [9] on data guides computing, the building time and result size are very reasonable for typical semi-structured database.

4 Query Optimization

In this section, we will discuss how the information described by a schema and schema based data storage can be used to significantly speed up query processing for a broad class of queries. Queries of semi-structured data will be evaluated in relational algebra, so relational query optimization technique can be widely utilized to improve query evaluation performance.

Example 4.1. We begin by tracing a very simple query over graph database DBGroup in Fig. 1. Suppose we wish to execute the following query to find titles of all publications in DBGroup.

```
  select t where member !  paper !  title !  t in
DBGroup
```

In this example, the query result is exactly the target set of member ! paper ! title in DBGroup. To find the target set, we simply traverse the label path from the root of the schema for DBGroup in Fig. 2, and we know there is only one such path. Hence we need examine only three nodes to reach the node corresponding to the target set, and get the result: π_{id2} (R6).

While a naive query evaluation approach has to examine exhaustively all member and paper nodes in the source to get the result and the efficiency may be much lower, compared to our method. In general, for a label path of length n, at most n nodes need to be examined to get the target set. For complex regular path expressions that usually appear in queries of semi-structured data, such superiority still exists.

Example 4.2. We now show a somewhat more interesting query. Suppose we wish to find titles of papers about database:

```
  select t where  member !  paper !  title ! t in
DBGroup,
                    l !  ε in t , IN( l, "database")
```

This query is similar to the previous example, but introduces a filtering condition. First consider processing the query with the naive method. A “top-down” exploration will need to examine the values of all paper titles and as in the previous example we might examine many member and paper nodes that do not lead to a result and even have no appropriate subnodes. Alternatively, we can build a query plan to take advantage of the schema-based data storage. Note that value relations divide atomic values according to label paths and all title values are stored in the relation R13. So we need only examine R13 to find all nodes whose values satisfy the filtering condition and the result is $\pi_{id} (\sigma_{IN (value , \text{“database”})} (R13))$. We can typically expect titles to be stored contiguously on disk, so such computation will be fast, with few additional random disk accesses. In addition, indexes could be applied to further speed up filtering of atomic values, for they are stored in relations.

Example 4.3. Suppose we now wish to find authors of papers with certain title:

```
select p where member !  p in DBGroup,
                paper !  title !  t in p,
                l !  ε in t , l = "database"
```

The third query is more difficult to deal with than the two formers, for we need not only to traverse down the schema and filter the relation of titles, but also need to traverse back to the result member nodes. Based on Theorem 2, we can achieve it with the help of R4 $\bowtie_{id2=id1}$R6: we can traverse “bottom-up” from valid title nodes to their corresponding member nodes along paths specified by tuples of R4 $\bowtie_{id2=id1}$R6. Hence we can get the query result by computing:

$$\pi_{R4.id1} (R4 \bowtie_{id2=id1} R6 \bowtie_{id2=id} \sigma_{value = \text{“database”}} (R13)).$$

The three examples illustrate how the schema-based data storage can be used to significantly speed up common queries. These techniques can be generalized to other queries as well.

5 Discussion

We have presented a technique to rearrange semi-structured data according to its schema, and have showed how such data storage can be used to support query optimization. Divide semi-structured data into relations based on its structure has a few benefits:

- Most relations are statically typed, thus provide an efficiency advantage for data management.
- Such rearrangement is based on paths from the root to data in the graph database, so it is well suited for query languages that access data by regular path expressions. We show that this method will significantly optimize query execution.
- Storing data in relations provides a good chance to apply relational techniques to semi-structured data, e.g., index and relational query optimization techniques.

We are considering the following areas for future research.

- Due to shared nodes and circles in the graph database, there may be information redundancy to some extent in the rearranged data. Although in practice the building time and result size are very reasonable for typical semi-structured database, we would like to investigate the possibility of performance guarantees over certain classes of databases. Ideally, we could formalize database characteristics that guarantee good performance. Strategies for dealing with poor cases are also important.
- With regard to query optimization, we plan to contrive a specific algorithm to implement query evaluation by relational operations, and to assess its performance.

References

1. Serge Abiteboul. Querying semi-structured data. In ICDT, 1997.
2. S Abiteboul, D Quass, J McHugh, et al. The Lorel query language for semistructured data. International Journal on Digital Libraries, 1997, 1(1):68~88
3. Peter Buneman, Susan Davidson, Mary Fernandez, and Dan Suciu. Adding structure to unstructured data. In ICDT, pages 336~350, Deplhi, Greece, 1997. Springer Verlag.
4. P Buneman, S Davidson, G Hillebrand and Dan Suciu. A query language and optimization techniques for unstructured data. In SIGMOD. San Diego, 1996.
5. Peter Buneman. Tutorial: Semistructured data. In PODS, 1997.
6. D.Calvanese, G.Giacomo, and M.Lenzerini. What can knowledge representation do for semi-structured data ? In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), 1998.
7. Mary Fernandez , Lucian Popa , Dan Suciu. A structure based approach to querying semistructured data. In Proceedings of the Workshop on Database Programming Languages, 1997
8. Mary Fernandez and Dan Suciu. Optimizing regular path expressions using graph schemas. In Proceedings of the International Conference on Data Engineering, 1998.
9. R Goldman and J Widom. DataGuides: enabling query formulation and optimization in semistructured databases. In VLDB, 1997.
10. J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. SIGMOD Record, 26(3): 54~66, September 1997.

Automatic Wrapper Generation for Web Search Engines

Boris Chidlovskii[1], Jon Ragetli[2]*, and Maarten de Rijke[2]**

[1] Xerox Research Centre Europe
6, Chemin de Maupertuis, 38240 Meylan, France
`chidlovskii@xrce.xerox.com`
[2] ILLC, University of Amsterdam
Pl. Muidergracht 24, 1018 TV Amsterdam, The Netherlands
`{ragetli,mdr}@wins.uva.nl`

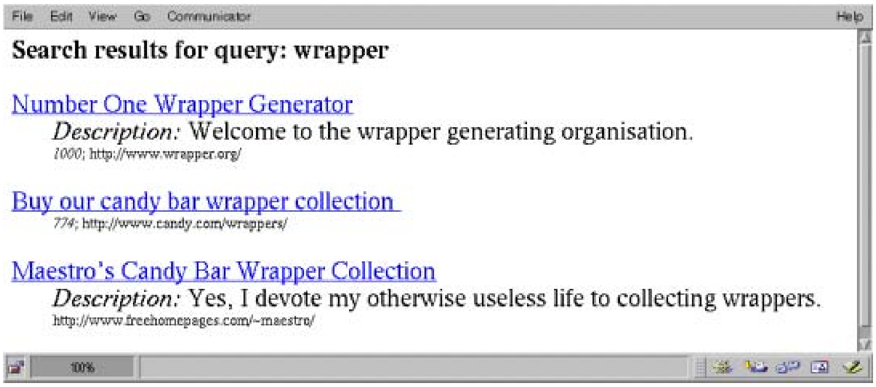
Abstract. To facilitate effective search on the World Wide Web, several 'meta search engines' have been developed which do not search the Web themselves, but use available search engines to find the required information. By means of wrappers, meta search engines retrieve relevant information from the HTML pages returned by search engines. In this paper we present an algorithm to create such wrappers automatically, based on an adaptation of the *string edit distance*. Our algorithm performs well; it is quick, it can be used for several types of result pages and it requires a minimal amount of interaction with the user.

1 Introduction

As the amount of information available on the World Wide Web continues to grow, conventional search engines expose limitations when assisting users in searching information. To overcome these limitations, mediators and meta search engines (MSEs) have been developed [2,6,7,8]. Instead of searching the Web themselves, MSEs exploit existing search engines to retrieve relevant information and combine it in a way which better satisfies the user's needs. This relieves the user from having to contact those search engines manually and knowing their native query languages; knowledge of the MSE's query language suffices. The MSE combines the results of the connected search engines and presents them in a uniform way.

MSEs are connected to search engines by means of so-called *wrappers*: software modules that take care of the source-specific aspects of the MSE. For every search engine connected to the MSE, there is a wrapper which translates a user's query into the native query language and format of the search engine. The wrapper also extracts the relevant information from the HTML result page of the search engine. We will refer to the latter as 'wrapper' and do not discuss the query translation (see [5] for a good overview). An HTML result page from

* Supported by the Logic and Language Links project funded by Elsevier Science.
** Supported by the Spinoza project 'Logic in Action.'



Search results for query: wrapper

Number One Wrapper Generator
Description: Welcome to the wrapper generating organisation.
1000; http://www.wrapper.org/

Buy our candy bar wrapper collection
774; http://www.candy.com/wrappers/

Maestro's Candy Bar Wrapper Collection
Description: Yes, I devote my otherwise useless life to collecting wrappers.
http://www.freehomepages.com/~maestro/

Fig. 1. Sample result page

a search engine contains zero or more *answer items*; an answer item is a group of coherent information making up one answer to the query. A wrapper extracts each answer item from the textual content and attributes of certain tags on the page as a tuple consisting of attribute/value pairs. For example, from the result page in Fig. 1 three tuples can be extracted, the first of which is displayed in Fig. 2. Like most result pages, the page in Fig. 1 shows variation in the items, as the second item lacks a description and the third a relevance ranking.

Manually programming wrappers is a cumbersome and tedious task [4], and since the presentation of the search results of search engines often changes, it has to be done frequently. Hence, there have been various attempts to automate this task [3,10,11,13,14]. The approach we describe is based on a simple incremental grammar induction algorithm. As input, it requires one result page of a search engine, from which the first answer item is labeled: the start and end of the item need to be indicated, as well as the attributes to be extracted. After this, the incremental learning of the *item grammar* starts, and using an adapted version of the *edit distance* measure, further answer items on the page are found and updates to the grammar are performed. Once this process is ready, the algorithm returns a wrapper for the entire page after some post-processing.

The key features of our approach are limited user interaction (labeling only one answer item) and good performance: for many search engines it generates working wrappers, and it does so very quickly. In the following sections, we will

```
⟨ url = "http://www.wrapper.org",
title = "Number One Wrapper Generator",
description = "Welcome to the wrapper generating organisation",
relevance = "1000" ⟩
```

Fig. 2. An item extracted

```
<HTML><HEAD><TITLE>Search results for query: wrapper</TITLE></HEAD>
<BODY bgcolor = "white" text= "black">
<H3>Search results for query: wrapper</H3>
  <dl>
     ^BEGIN^ <dt> ^URL^ <a href="http://www.wrapper.org/"> ^^
     ^TITLE^ Number One Wrapper Generator ^^ </a><br>
       <dd><i>Description:</i> ^DESCR^ Welcome to the wrapper
       generating organisation. ^^ <br>
       <font size="-3"><I> ^REL^ 1000 ^^ </I>;
       http://www.wrapper.org/</font> ^END^
  </dl>
  <dl>
    <dt><a href="http://www.candy.com/wrappers/">
    Buy our candy bar wrapper collection </a><br>
                 ⋮
  </dl>
</BODY></HTML>
```

Fig. 3. Labeled HTML source of result page

first discuss our wrapper generating algorithm. After this, experimental results will be described, and we compare our method to other approaches. Finally, we conclude and discuss future work.

2 The Wrapper Generator

In this section we discuss the input and output of the Wrapper Generator (WG), after which we give an overview of its algorithm. In the next section we will go into more detail. All known approaches for automatically generating wrappers require as input some labeled HTML pages: all or some of the attributes to be extracted from the page have to be marked — manually, or automatically by a labeling program. As it is hard to create labeling programs for the heterogeneous set of search engines an MSE must be connected to, and the labeling is a boring and time-consuming job, we have restricted the labeling for our algorithm to a single answer item only. Figure 3 shows the labeled source of the HTML page in Fig. 1. The labeling consists of an indication of the begin and end of the first answer item (`^BEGIN^` and `^END^`, respectively), the attribute names (e.g. `^URL^`), and the end of the attributes (`^^`).

The output of the WG is a wrapper, whose task is to output zero or more *tuples* consisting of relevant information. Each element of the tuple is an attribute/value pair; the attribute names are provided by the human who labeled the page. Figure 2 shows the first tuple extracted from the result page in Fig. 1.

Our wrapper generation algorithm, shown in Fig. 4, works as follows. First, the result page is abstracted to a sequence of tokens, after which an *item grammar* is initialized. Then, using this item grammar, other items on the page are

```
1. $AP$ := abstract($LP$)
2. $G$ := initialize($AP$)
REPEAT
   3. $I$ := find-next-item($AP$, $G$)
   4. IF $I \neq \emptyset$
         THEN $G$ := incorporate-item($G$, $I$)
UNTIL $I = \emptyset$
5. $GP$ := grammar-whole-page($G$, $AP$)
6. $W$ := translate-to-wrapper($GP$)
7. return $W$
```

- AP is the abstracted page
- LP is the HTML page labeled by the user
- G is the item grammar
- I is an item on the abstracted page
- GP is a grammar for the whole page in an abstract format
- W is the same grammar, translated into a working wrapper

Fig. 4. The wrapper generating algorithm

found, and the grammar itself is updated to cover those items. Once all the items have been found, the grammar is extended to cover the entire page, and finally translated into a working wrapper.

Steps 2, 3 and 4 show that the grammar building algorithm is incremental: the grammar G is updated whenever a new item is encountered. The fact that the algorithm itself finds the items on the page saves the user the burden of labeling every item on the page, a feature that other approaches (e.g. [3,13,14]) lack. Another feature is that the user does not have to indicate the begin and end of the item precisely. When an item has been indicated more narrow than it actually is, step 5 takes care of finding the real item size. In Fig. 3 the item is indeed indicated too narrow; the fragment of HTML between `<DL>` and `</DL>` is repeated on the page, instead of the smaller item indicated there.

3 Details of the Algorithm

For presentational purposes we do not describe the steps in the same order as the algorithm performs them. The paragraphs describe steps of the algorithm as well as underlying theory.

Step 1: Abstract. In the `abstract` step, the entire HTML page is transformed into a string of tokens. Each tag is abstracted to one token for that tag,[1] and everything else on the page is abstracted to the token C (for Content). By representing each token by a symbol, we can now view the HTML page as a string of symbols. This allows us to use an algorithm for comparing strings, as described below, to find the items on the page other than the labeled first one.

[1] This abstraction generalizes over the contents of the tag. E.g. both tags `<font face="Arial">` and `<font face="Helvetica">` are abstracted to `<FONT>`.

The Item Grammar. The item grammar is an important foundation of the WG algorithm. It both represents the structure of the items already encountered on the page, and it is used to find the next item on the page. It is defined as follows:

Definition 1 (Item Grammar).

1. *A symbol is an item grammar.*
2. *If G_1 and G_2 are item grammars, then G_1G_2 is an item grammar, meaning G_1 concatenated to G_2.*
3. *If G is an item grammar without substrings of the form '$[S]$', where S can be any string, $[G]$ is an item grammar.*

Since an HTML page is abstracted into a string of symbols, the item grammar represents sequences of content and HTML tags. The square brackets represent optionality; e.g. $a[b]c$ covers both the strings *abc* and *ac*. The third clause of the definition prevents optional parts from being nested. By including or discarding optional parts, the item grammar defines sequences of HTML and content that can appear on the result page. Although item grammars are less expressive than regular grammars, they are expressive enough to grasp variations in the item structure.

Step 2: Initialize. In the `initialize` step, the item grammar is initialized by making it equal to the string of symbols that represents the first, labeled item on the page. The user has labeled the attributes in the first item with a name; the algorithm remembers their position and name.

Edit Distance. The simple form of the item grammar makes it possible to use the grammar to find other items on the page in `find-next-item`, using an adapted form of *edit distance*. We will first recall the edit distance algorithm for strings [1]. Next, we discuss our adapted form of edit distance.

Definition 2 (Edit distance). *The* edit distance $D(s_1, s_2)$ *between two strings of symbols s_1 and s_2 is the minimal number of insertions or deletions of symbols, needed to transform s_1 into s_2.*

For example, $D(abcd, abide) = 3$, because in order to transform *abcd* into *abide* at least three insertion or deletion operations have to be performed: delete the c in *abcd*, and insert an i and an e. Algorithms calculating the edit distance can also return the difference between the two strings in the form of a so called *alignment*. This difference is used in `incorporate-item` to indicate how to adapt the item grammar on the basis of the new item (see the next subsection). As an example, for *abcd* and *abide* the alignment is the following:

$$\begin{array}{cccccc} \hline a & b & c & - & d & - \\ a & b & - & i & d & e \\ \hline \end{array}$$

Here we omit the details of the edit distance and alignment algorithm and refer the interested reader to [1,15].

Let G_i denote the item grammar constructed for the first i items. In steps 3 and 4 of our algorithm (Fig. 4), we calculate the distance between a grammar

item grammar	a	b	–	d
string	a	b	c	d
new item gr.	a	b	$[c]$	d

(a)

item grammar	a	b	$[c]$	d
string	a	b	c	–
new item gr.	a	b	$[c]$	$[d]$

(b)

item grammar	a	b	–	d	
string	a	–	c	d	
new item gr.	a	$[b]$	$[c]$	$[b]$	d

(c)

Fig. 5. Three alignments

G_i and a string representing the $i+1$-th item, in order to construct the item grammar G_{i+1} covering all $i+1$ items. Therefore, we have adapted a method to calculate edit distance to work for an *item grammar* and a string instead of only for strings. The adaptation amounts to first simplifying the item grammar by removing all brackets, while remembering their position. Now the edit distance between the item and the simplified grammar can be calculated as usual: both are strings of symbols. Using the alignment and the remembered position of the brackets, the new grammar is calculated, as will be described next.[2]

Step 4: Incorporate-item. The algorithm detects and processes different cases in the alignment between G_i and $i+1$-th item. Since the full algorithm description is extensive and space is limited, we can only illustrate how it works by means of some examples (Fig. 5). A more elaborate description is given in [15].

In Fig. 5 (a), the original item grammar does not contain a c, while the string to be covered does. Therefore, the new item grammar has an optional c in it, so that it covers both *abd* and *abcd*. Now suppose the string *abc* has to be covered by the new item grammar, as in Fig. 5 (b). The reason for making the d optional is that the new string shows that it does not occur in every string. Note that the new item expression now covers the strings *ab*, *abc*, *abd* and *abcd*, which is a larger generalization than the simple memorization of all the examples.

The situation in Fig. 5 (c) is less straightforward, as the new item grammar $a[b][c][b]d$ is a large generalization; besides the original examples *abd* and *acd* it covers *ad*, *abcd*, *abbd*, *acbd* and *abcbd* as well. The reason for this large generalization is that based on the examples we can conclude at least that the b and c are optional. Moreover, they may also occur at the same time and in any order.

Step 3: Find-next-item. Incorporate-item generates an item grammar for a set of items on the page. The WG also uses the item grammar for finding these items. The finding mechanism is iterative, it starts with the first labeled item and uses grammar G_i to find $i+1$-th item. The finding of the items is crucial for our algorithm, as it allows us to have only one item labeled on the search result page. We have implemented three different strategies for finding the answer items, but as space is limited we only describe the one that works for most sources: the *Local Optimum Method* (LOM). The other two are simpler and mostly quicker methods, but even with the LOM wrappers are generated quickly; see Section 4.

[2] We have also adapted the edit distance algorithm to deal with labeled attributes in the grammar, that correspond with unlabeled content in the item.

```
1. $D_{newlocal} := 998$, $D_{local} := 999$, $D_{best} := 1000$
2. $i_b$, $i_e := 0$
3. local-best-item $:= \emptyset$, best-item $:= \emptyset$
WHILE $D_{local} < D_{best}$ and not at end of page
    4. $D_{best} := D_{local}$
    5. best-item := local-best-item
    6. $i_b :=$ next occurrence begin tag(s)
    WHILE $D_{newlocal} < D_{local}$
        7. $D_{local} := D_{newlocal}$
        8. local-best-item $:= (i_b, i_e)$
        9. $i_e :=$ next occurrence end tag(s)
        10. $D_{newlocal} := D($item grammar$, (i_b, i_e))$
11. IF $D_{best} >$ Threshold THEN best-item $:= \emptyset$
12. return best-item and $D_{best}$
```

- $D_{newlocal}$ stores the distance of the item grammar to the part of the page between the latest found occurrence of the begin and end tag(s)
- D_{local} stores the distance of the item grammar to `local-best-item`
- D_{best} stores the distance of the item grammar to `best-item`
- i_b, i_e are the indexes of the begin and end of a (potential) item
- `local-best-item` stores the potential item starting at i_b that has the lowest distance to the item grammar of the potential items starting at i_b
- `best-item` stores the potential item that has the smallest distance to the item grammar so far

Fig. 6. The Local Optimum Method

All our methods for finding items are based on an important assumption: *all items on the page have the same begin and end tag(s).* Consequently, the task of finding items on the page is reduced to finding substrings (below the last found or labeled item) that have the same start and end delimiters. The user can decide for how many tags this assumption holds by setting the parameter *SeparatorLength.* If SeparatorLength is increased, it will be easier to find the items on the page; the chance of finding for example a sequence of two tags is smaller than that of finding one tag. However, setting the parameter too high will result in not finding all items.

The LOM finds items on the page that are *local*, i.e., below and close to the item that was found last, and *optimal* in the sense that their distance to the item grammar is low. Figure 6 shows the algorithm. In the first three steps, a number of variables are initialized. The outer *while* loop makes the start of potential items vary, while the inner while loop does the same for the end of the potential items. Thus, several combinations of begin and end tags are considered, and the one that is not far below the previous found item (*local*) and has a low distance to the grammar (*optimal*) is selected to be incorporated into the grammar.

In step 11, if the distance of the best candidate item to the item grammar exceeds *Threshold*, the algorithm will return $\emptyset$ instead of the item, thus preventing the grammar from adjustment, and the process of finding the items stops (see Fig. 4). *Threshold* depends on two parameters: *HighDistance* and *Variation*. *HighDistance* is the maximum distance from the grammar evaluated among all items incorporated previously. The initial value of *HighDistance* is set by user, and it is incremented whenever a new incorporated item has a distance higher than *HighDistance*. *Variation* is set by user as well, but it does not change during the process of finding the items. Combined together in *Threshold*, *HighDistance* and *Variation* form a flexible way for finding and incorporating new items.

Step 5: Making a Grammar for the Entire Page. Once the WG has found all the items on the page, and adjusted the item grammar accordingly, the item grammar can only extract the useful information from one item — not yet from the entire page. Here we recall that HTML pages for which a wrapper is generated, are assumed to contain a sequence of items, possibly mixed with irrelevant information. Above and below the sequence there might be irrelevant data as well. This assumption translates in a natural way into a skeleton for a wrapper:

```
1. skip the top of the page
WHILE there is another item
   2. parse item
   3. skip irrelevant data if present
```

It is not necessary to recognize the irrelevant data at the bottom of the page explicitly; the wrapper stops when it does not recognize any further items. For step 2, parsing the items, we already have the item grammar. But, as mentioned before, the user might have labeled the first item smaller than it actually is. By the assumption that all the items on the page have the same begin and end tags, the found items (and the resulting item grammar) will also be too small. Therefore, the item grammar will be extended by finding common prefixes and suffixes of the HTML between the found items.

Two parts of the wrapper are still missing: a part that skips the top of the page, and a part that skips irrelevant data within the item list. For skipping the top of the page and recognizing the start of the list of items, the smallest fragment of HTML just above the first item is taken that does not occur higher on the page as well. For skipping the useless HTML between the items in the list, another kind of item grammar is constructed — the *trash grammar*. The indices of the (extended) items that were found have been stored, so this process is a straightforward repetition of `incorporate-item`. Once this trash grammar has been constructed, it is appended to the end of the item grammar. When the item and trash grammars have been generated, the WG will detect repetitive patterns in them and generalize them accordingly. This kind of generalization is appropriate, as certain fragments can re-occur arbitrarily often on result pages, like author names with enclosing tags.

After all these processing steps the WG translates the abstract grammar into a working wrapper. In our implementation this is a JavaCC parser [12], for the Knowledge Brokers meta search engine developed at Xerox Research Centre Europe is programmed in Java. However, our WG is not restricted to generate JavaCC parsers; the translation step can easily be replaced.

Table 1. Experimental results

Successfully generated wrappers				
source	URL	size (kB)	NI *	time (sec)
ACM	www.acm.org/search	12	10	8.0
Elsevier Science	www.elsevier.nl/homepage/search.htt	11	11	2.6
NCSTRL	www.ncstrl.org	9	8	32.5
IBM Patent Search	www.patents.ibm.com/boolquery.html	19	50	5.3
IEEE	computer.org/search.htm	26	20	3.7
COS U.S. Patents	patents.cos.com	17	25	5.4
Springer Science Online	www.springer-ny.com/search.html	36	100	32.1
British Library Online	www.bl.uk	5	10	2.6
LeMonde Diplomatique	www.monde-diplomatique.fr/md/index.html	6	4	2.5
IMF	www.imf.org/external/search/search.html	10	50	5.3
Calliope	sSs.imag.fr**	22	71	4.1
UseNix Association	www.usenix.org/Excite/AT-usenixquery.html	16	20	4.3
Microsoft	www.microsoft.com/search	26	10	4.5
BusinessWeek	bwarchive.businessweek.com	13	20	3.9
Sun	www.sun.com	20	10	3.7
AltaVista	www.altavista.com	19	10	4.1

Sources for which the algorithm failed to generate a wrapper	
source	URL
Excite	www.excite.com
CS Bibliography (Univ. Trier)	www.informatik.uni-trier.de/˜ley/db/index.html
Library of Congress	lcweb.loc.gov
FtpSearch	shin.belnet.be:8000/ftpsearch
CS Bibliography (Univ. Karlsruhe)	liinwww.ira.uka.de/bibliography/index.html
IICM	www.iicm.edu

* NI is the number of items.
** Only accessible to members of the Calliope library group.

4 Experimental Results

We have tested our wrapper generator on 22 search engines, a random selection of sources to which Knowledge Brokers had already been connected manually. It was quite successful, as it created working wrappers for 16 of the 22 sources (73%). For 2 other sources the generated incorrect wrappers could easily be corrected. Since working wrappers were created with only one answer item labeled, good generalizations are made when the item grammar is induced: labeling only *one* item of *one* page is sufficient to create wrappers for many other items and pages of the same source. Table 1 summarizes the experimental results; the times were measured on a modest computer (PC AMD 200MMX/32 Mb RAM).

The time needed to generate a wrapper is very short; the maximum time is 32.5 seconds, the average 7.8 seconds. Together with the small amount of labeling that has to be done, this makes our approach very rapid. In general, it is not the case that it takes more time to create a wrapper for larger pages than for smaller ones. A large page may contain a lot of irrelevant data at the top, where no items are sought, thus not increasing the time to create a wrapper. The same holds for pages with more items versus pages with less items. For a page with a lot of items that has very simple structure (for example when the begin and end delimiters only occur for real items) a wrapper can be learned more quickly than for a more complex page with less items.

Increasing the *SeparatorLength* parameter (see Section 2) speeds up our algorithm, as fewer fragments of HTML are considered. For NCSTRL, the time to generate a wrapper is shown with a *SeparatorLength* of 1 (32.5 seconds), as 1 is the default *SeparatorLength.* However, with a *SeparatorLength* of 2, it takes 22.5 seconds, with 3 it takes 21.4 seconds, and with 4 17.1 seconds.

Robustness of the Wrappers. An important aspect of the generated wrappers is the extent to which the result pages of the search services may change without the wrapper breaking down. For our wrappers, little is allowed to change in the list with search results, because the wrapper for that list is generated by making not too large generalizations. But even if the wrappers are not very robust, it is easy to create a new one whenever the search engine's result pages change, since the algorithm is fast and requires limited user interaction.

Incorrect Wrappers. The wrapper generated for Excite did not work because the code to extract the URL from `<a href = "...">` tags was not general enough and did not recognize the unquoted URL in the Excite answer page. After manually correcting the wrapper, it worked properly. A similar correction produced a working wrapper for IICM. For the Library of Congress, the fact that the WG only distinguishes between tags and textual content caused it to fail, as the attributes on the result pages were only separable by textual separators.

The WG did not create a working wrapper for the Computer Science Bibliography (Univ. of Trier) and FtpSearch because the right items were not found due to too much variation in the items, causing the distance between the item grammar and the item found to be too high. Increasing the parameters *HighDistance* or *Variation* could not change this, because then fragments of HTML that did not correspond to an item were incorrectly incorporated in the grammar. The problem with the Computer Science Bibliography (Univ. of Karlsruhe) concerned the detection of repetitions in the item expression. Complex repetitions on the page make the wrapper generator create a repetitive part with only optional parts. This would cause the wrapper to enter an infinite loop. Another reason was that not all information belonging to an answer item was located with the item itself; some of it was shown in a header above a number of results.

5 Comparison to Other Approaches

In [9], Hammer et al. present a simple approach to semi-automatically generating wrappers that lies in between hand-coding the wrappers and creating them fully automatically: specifying them at a high level.

Kushmerick et al. [13] present a template-based approach for building wrappers for HTML sources. They use recognizers to label the page automatically. That is very useful, as their algorithm requires entirely labeled pages .

In [3], Ashish and Knoblock present quite a different approach to generating wrappers, focusing on making static HTML pages queriable. Their wrappers are constructed without labeling, but by *structuring* the page, using certain assumptions about how the nesting hierarchy within a page is reflected in the layout. Their algorithm is not well-suited for making wrappers for our domain (pages with search results), because there are no general heuristics applicable to multiple search engines. Soderland [16] also uses lay-out cues to construct wrappers. Furthermore, Soderland's system uses a semantic lexicon which makes the approach very different from ours. Besides automatically generated pages, his domain consists of less structured hand-crafted pages.

Muslea et al. [14] discuss the automatic generation of hierarchical wrappers. A drawback of their approach is that the user has to label several pages entirely, albeit in a graphical interface. The hierarchical wrappers do not suffer from the problem we mentioned with respect to the Computer Science Bibliography at the Univ. of Karlsruhe, that attributes belonging to several different items cannot be extracted. The hierarchical form of the wrappers makes it possible to decompose the problem of generating wrappers for entire result pages into smaller problems. While our approach is bottom-up, the approach of Muslea et al. is top-down.

The approach of Hsu et al. [10,11] is similar to ours. Their finite-state transducers, called *single-pass SoftMealy extractors* resemble the grammars that we generate. Their abstraction of textual content on the pages is a more fine-grained. This makes their approach more widely applicable than to HTML pages. Experimental results show that their approach does not need many labeled items, albeit more than ours. It seems that their approach is better in handling differences in the order of the attributes, but we have not fully tested this. Further investigation — both empirical and analytical — of the differences between the two approaches should make this clear.

6 Conclusion and Further Work

We have presented an approach to automatically generate wrappers, which uses grammar induction based on an adapted form of *edit distance*. Our wrapper generator is language independent, because it relies on the structure of the HTML code to build the wrappers. Experimental results show that our approach is accurate — 73% of the wrappers generated is correct (allowing minor modifications: 82%). Furthermore, our generator is quick, as the average time to generate a wrapper is less than 8 seconds.

The major advantage of our approach is the small amount of labeling by the user: labeling only one item suffices. The other items are found by the wrapper generator itself. Comparing our algorithm to others, we conclude that it creates good wrappers with little user interaction. The approach of Hsu et al. [10,11] seems to create better wrappers, but at the price of more extensive user input.

Although the Wrapper Generator performs well, several improvements and extensions are possible. For example, a program with a graphical interface can simplify the labeling, that is currently performed with a text editor. One of the assumptions underlying our wrapper generator is that all attributes can be separated by HTML tags, but not all result pages satisfy this assumption. By making finer-grained abstractions, we should be able to generate wrappers for such pages. On the other hand, the HTML separability causes the wrapper generator not to rely on specific textual content on the pages. That makes this approach natural language independent.

If many search engines for one domain have to be connected to a meta searcher, it is worthwhile to create *recognizers* [13] that find and label the attributes automatically. Finally, we have deliberately investigated the power of our method with minimal user input, but further research is needed to clarify the trade-off between user interaction and quality of the generated wrappers.

References

1. Aho, Alfred V. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 255–300, Elsevier, 1990.
2. Andreoli, J.-M., Borghoff, U., Chevalier, P-Y., Chidlovskii, B., Pareschi, R., and Willamowski, J. The Constraint-Based Knowledge Broker System. *Proc. of the 13th Int'l Conf. on Data Engineering*, 1997.
3. Ashish, N., and Knoblock, C. Wrapper Generation for Semi-structured Internet Sources. *SIGMOD Record* 26(4):8–15, 1997.
4. Chidlovskii, B., Borghoff, U., Chevalier, P.-Y. Chevalier. Toward Sophisticated Wrapping of Web-based Information Repositories. *Proc. 5th RIAO Conference, Montreal, Canada*, pages 123–135, 1997.
5. Florescu, D., Levy, A., and Mendelzon, A. Database techniques for the World-Wide Web: A Survey. *SIGMOD Record 27(3)*:59–74, 1998.
6. Garcia-Molina, H., Hammer, J., and Ireland, K. Accessing Heterogeneous Information Sources in TSIMMIS. *AAAI Symp. Inform. Gathering*, pages 61–64, 1995.
7. Gauch, S., Wang, G., Gomez, M. ProFusion: Intelligent Fusion from Multiple Distributed Search Engines. *J. Universal Computer Science*, 2(9): 637–649, 1996.
8. Gravano, L., Papakonstantinou, Y. Mediating and Metasearching on the Internet. *Data Engineering Bulletin 21(2)*, pages 28–36, 1998.
9. Hammer, J. Garcia-Molina, H., Cho, J., Aranha, R., and Crespo, A. Extracting Semistructured Information from the Web. *Proceedings of the Workshop on Management of Semistructured Data*, 1997.
10. Hsu, C.-N., and Chang, C.-C. Finite-State Transducers for Semi-Structured Text Mining. *Proc. IJCAI-99 Workshop on Text Mining*, 1999.
11. Hsu, C.-N., and Dung, M.-T., Generating finite-state transducers for semistructured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.
12. JavaCC – The Java parser generator. URL: `http://www.metamata.com/JavaCC/`.
13. Kushmerick, N., Weld, D.S., and Doorenbos, R., Wrapper Induction for Information Extraction. *Proc. IJCAI-97*: 729–737, 1997.
14. Muslea, I., Minton, S., Knoblock, C. STALKER. *AAAI Workshop on AI & Information Integration*, 1998.
15. Ragetli, H.J.N. Semi-automatic Parser Generation for Information Extraction from the WWW. Master's Thesis, Faculteit WINS, Universiteit van Amsterdam, 1998.
16. Soderland, S. Learning to Extract Text-based Information from the World Wide Web. *Proc. KDD-97*, pages 251–254, 1997.

Multi-level Schema Extraction for Heterogeneous Semi-structured Data

Jong P. Yoon and Vijay Raghavan

Center for Advanced Computer Studies
University of Louisiana, Lafayette, LA 70504-4330
{jyoon, raghavan}@cacs.louisiana.edu

Abstract. Heterogeneous information sources are organized in various different degrees from well-structured data, to unstructured and semi-structured data. Such information sources do not have rigid schema available in advance or even if each source has its own schema, there are no enforced modeling constraints or formats for data across information sources. In this paper, we propose a novel method for abstracting schemas for heterogeneous information sources. At the most detailed level, information sources are represented in a labeled directed graph. We develop several abstraction operations for label generalization and aggregation. One of more of these operations can be applied to a labeled directed graph to "levelize" schemas. Each such level of the schemas is a potentially useful paradigm for query formation and optimization.

1 Introduction

The number of information sources accessible electronically is growing rapidly in web-based information systems. Many of these sources store and export unstructured data in addition to or instead of structured data. We consider information sources to be semi-structured data when there is no schema fixed or known in advance and when the data may be incomplete or irregular. Semi-structured information sources are usually marked by tagged elements. The proliferation of such un-/well-structured and semi-structured information sources together on the Internet can make it difficult to formulate what to retrieve and how. We propose a method for generating schemas of heterogeneous information sources: structured and semi-structured data, and internal and external data. Due to substantial variability in types of data formats and sources available on the Internet, it is essential to develop methods for levelizing schemas for those heterogeneous information sources.

1.1 Problem Statement

EXtensible Markup Language (XML) is an emerging standard for data representation and exchange on the Internet. Due to the inherent flexibility of XML [3], semi-structured information sources can be represented in XML. Even so, because of the co-existence of semi-structured data with un-/well-structured data, queryformulationisnoteasyandefficientwithoutknowinginadvance the common structure of the data. The following is a statement of such problems that this paper aims to address.

- No Uniform Information Schema Extraction. Each different structure of information sources has different schema extraction and representation. Well-structured data may have a relational schema, and unstructured data does not usually have a schema, for example. However, although semi-structured data inherently has a (complex) structure, such a data does not have schema in advance. These heterogeneous information sources therefore do not have uniform method for information schema extraction.
- No Interoperation among Schemas. Some researchers propose a technique for approximate schema generation [10] that falls in between the most abstracted schema (e.g., DTD) and the least abstracted schema (e.g., [9]). However, a weakness of existing work is that there has been no method developed for making a broad spectrum of schemas interoperate with one another and useful for query formulation and optimization in heterogeneous information systems.

1.2 Contributions of this Paper

To retrieve semi-structured information sources together with un-/well-structured information on the Internet, users need to know the schema of those data. Broadly speaking, we initially extract schemas for each such information source by capturing relationships between the interesting regions (or material enclosed by tagged elements), and by abstracting, generalizing, and aggregating those initially extracted schemas. The major contribution of this paper is schema abstraction operations. These operations are applied to a schema that is initially extracted from structured data. These operations allow us to obtain what we refer to as levelized schemas.

1.3 Organization of this Paper

The remainder of this paper is organized as follows: Related work is described in Section 2. Section 3 describes preliminaries. Section 4 describes schema extraction, and Section 5 describes a method of schema levelization. Section 6 describes integration of the levelized schemas. Finally, this paper concludes our accomplished work with future research work in Section 7.

2 Related Work

With the recent emergence of *XML* [3], a proposed standard for exchanging information on the Web [11], and the remarkable similarity of XML to typical models for semi-structured data, support for query languages for semi-structured data – and the performance of such queries over large semi-structured databases – is of increasing importance.

The extraction of the schema from a semi-structured data follows either unsupervised categorization or supervised categorization. The former we call *clustering*, the latter *classification*. Many researchers focus on classification of semi-structured data [4, 8]. One example of classification is summarization of information sources [6]. The approach presented in this paper is instead to "cluster" information sources. In general, semi-structured information sources are clustered according to tagged-elements. After a schema implicit in the cluster of semi-structured data is extracted, abstraction techniques are applied to the initial schema.

A *generalized path expression*, useful in the context of XML-like semi-structured databases, allows label wildcards and regular expression operators [1]. Generalized path expression optimization has been studied in [5, 7]. [7] describes a query rewrite technique that transforms generalized path expressions to simpler forms prior to optimization. In [5], an algebraic optimization framework is proposed with the goal to specifically avoid exponential blow-up in the presence of closure operators. Our work is similar in spirit, but not in details, to [7]. In [7], a cross-product is computed between a *graph schema* – a summary of the database that must be small and reside in memory – and a representation of the query. The work in this paper, instead, employs not only generalization of label paths but also aggregation of and redundancy elimination in them. In addition, this paper deals with a way of integration of levelized schemas in order to guide users to formulate queries and also to make it possible to process the queries efficiently.

3 Preliminaries

3.1 XML DTD and XML Data

A DTD (Document Type Declaration) defines a class of XML data using a language that is essentially a context-free grammar with several restrictions. We represent semi-structured data in XML using DTD. For example, one may use the DTD declaration as in Figure 1 to constrain XML (`paper`) example.

The first line (1) in the figure expresses a *document type declaration*. Line (2) expresses an *element type declaration* that contains the contents of the `Paper` element. Notice that DTD and XML follow notational convention, such as `?`, `*` and `+` denoting respectively zero or one, zero or more, and one or more occurrences of the preceding construct. The `Paper` element contains one `title` element, one or more `author` elements, one `body` element, zero or more `references` element, and zero or one `external_info` element. In Line (3), the type of the `title` element is PCDATA (parsed character data). The `references` element in Line (6) has an attribute list where the types of the two attributes "title" and "author" are CDATA (character data). The `external_info` consists of six subelements in Line (7). The `lang` element may have two attributes in Line (9): 1) code character data type with default value ASCII, 2) three possible sources of English (default value), Spanish and French. A physical storage in DTD is called *entity*. In line (10), the entity as a storage unit contains the parameter `Xlink`. The `body` element in Line (11) may consist of one or more sections. Each such section in

```
(1)  <!DOCTYPE Paper [
(2)  <!ELEMENT Paper (title, author+, body, references*, external_info?)>
(3)  <!ELEMENT title (#PCDATA)>
(4)  <!ELEMENT author (#PCDATA|(lastname,firstname)|fullname)>
(5)  <!ELEMENT references (#PCDATA)>
(6)  <!ATTLIST references title CDATA author CDATA>
(7)  <!ELEMENT external_info (year,volume?,month?,price,publication,lang)>
(8)  <!ELEMENT lang (#PCDATA)>
(9)  <!ATTLIST lang code CDATA   source (eng|spn|frn) #DEFAULT eng>
(10) <!ENTITY % Xlink  " xml:link CDATA #FIXED 'simple'
             href CDATA #REQUIRED  img      CDATA #REQUIRED">
(11) <!ELEMENT body    (section)+>
(12) <!ELEMENT section (title,(para+|section*))>
(13) <!ELEMENT para    (size?, object*)>
(14) <!ELEMENT size    (#PCDATA)>
(15) <!ATTLIST para    %Xlink;>
(16) <!ELEMENT object  (((image|video|figure|table),caption*)|object*)>
(17) <!ELEMENT image   %Xlink;>]>
```

Fig. 1. DTD Example for Papers

Line (12) consists of the (section) `title` element, one or more para(graph) element, or zero or more subelements called "`section`." The `object` element can contain one or more (sub)objects within itself in Line (16). That is, the `image` element can contain one or more images within the image. This enables an image multimedia data to be composed of one or more component images. XML data often specifies nested and cyclic structures, such as trees, directed graphs, and arbitrary graphs. We assume that the type of all the other elements (e.g., firstname) is PCDATA.

To ground our examples, let us illustrate a tiny portion of simple semistructured data. Consider an XML data shown in Figure 2 according to the DTD in Figure 1.

3.2 Data Graph for XML Data

Semistructured data, like XML-based instances, can be thought of as a labeled directed graph. The data defined in the previous subsection can be depicted as given in Figure 3. We call this a *data graph*. The nodes in the graph are *objects*; each object has a unique *object identifier* (oid), such as &01. *Atomic objects* have no outgoing edges and contain a value from one of the basic atomic types such as `integer`, `string`, `gif`, `video`, etc. All other objects may have outgoing edges and are called *complex objects*. Object &14 is complex and its *subobjects* are &15 and &16. Object &15 and &16 are atomic objects and have values, say "Henry" and "Smith." There is a label to an arc between objects. For example, the label to the arc from the object &14 to the object &15 is called "firstname."

```
<?xml version="1.0"?>
<Paper>
<title American History />
<author>
  <lastname> Smith </lastname> <firstname> Henry </firstname>
  <fullname> Matthew Johnston </fullname>
</author>
<body>
  <section> <title>19th Century </title>
            <para> About 150 years,.... </para> </section>
  <section> <title>19th Century </title>
            <para> About many decades, there are ... </para>
    <section> <title>... </title>
      <para> This figure depicts ... </para>
      <para>
        <object><figure><img src="www.a.b.c/f/jefferson.gif"></figure>
          <object><figure><img src="www.a.b.e/j/statue.jpg">
             <img src="www.a.b.e/c/constitution.jpg"></figure>
           <caption>The constitution is marked in a rock.</catptison></object>
           <object><figure><img src="www.a.b.e/c/memorial_dome.jpg"></figure>
           <caption>The Jefferson statue ..is around the wall of...</caption>
        </object>
</para>
  </section></section>
</body>
<references> <title> ... </title><author> ... </author></references>
</Paper>
<Paper>
<title Modern History />
<author> Smith Henry </author>
<lang source = ‘‘French’’>
<body>
  <section><title>Late centry</title>
           <para> About 200 years ago,.... </para></section>
  <section><title>Last centry</title> <para> In those decades,... </para>
    <section> <title>..... </title>
      <para>..... </para>
      <para> <object> <figure>
      </figure></object> </section> </section>
  <section> ....... </section>
</body>
<references> <title> t1 </title><author> a1 </author>
             <title> t1 </title><author> a2 </author>
                                <author> a1 </author>
</references>
</Paper>
```

Fig. 2. Two XML Data Example

3.3 Label Path Expression

In the data graph, there exists a path from the root node in terms of labels. A label path is expressed a sequence of $x.ly$, where x and y denote sets of oids, l denotes a label for an arc from x to y. That is, y is the set of all l-labeled subobjects of x. Notice that a label expression $x.l_1.l_2.z$ is rewritten as $x.l_1y, y.l_2z$. y is an sub-object of x and also a super object of z. If x is an atomic object, or if l is not an outgoing label from x, then $x.l$ is the empty set.

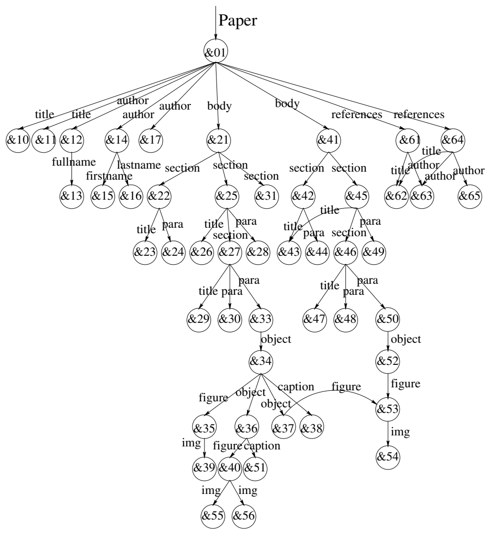


Fig. 3. Semistructured Data

3.4 Lattice

Suppose we are given a set of tagged-elements $\mathcal{L}$, and a partial ordering $\prec$ on the elements in $\mathcal{L}$. Consider two labels l_1 and l_2. When $l_1 \prec l_2$, we say that l_2 is a more general label than l_1. We call l_2 a *ancestor* of l_1 if $l_1 \preceq l_2$. When $l_1 \prec l_2$, we say that l_1 is a more specific label than l_2. We call l_1 an *descedant* of l_2 if $l_1 \preceq l_2$. For example, in Figure 3, the label *Paper.author.lastname* can be obtained using only the values bound to the label *Paper.author*. Thus *Paper.author.lastname* $\preceq$ *Paper.author*.

Note that $\preceq$ imposes a partial ordering on the label expression, and it is transitive. We shall talk about the relationships of label expression as forming a lattice [13]. In order to be a lattice, any two elements (i.e., tagged elements or label) must have a least upper bound and a least lower bound according to the $\preceq$ ordering. However, in practice, we only need the assumptions that

1. $\preceq$ is a partial order, and
2. The is a *top* ($\top$) element, a label upon which every label is dependent.

The *least lower bound* of labels l_1 and l_2, denoted by $llb(l_1, l_2)$, is the least common ancestor of l_1 and l_2 if one exists. The *least upper bound* of labels l_1 and l_2, denoted by $lub(l_1, l_2)$, is the least common descedant of l_1 and l_2 if one exists.

4 Schema Extraction

We consider in this paper XML data which contain 1) well-structured data, 2) unstructured data, 3) semi-structured data, and 4) external data electronically reachable. For each different type of information sources, there exists a schema extracted as follows. Each data can be represented as a data graph.

4.1 Schema for Structured Data

Typical bibliography-based searchable data can be easily modeled by a relational data model and stored and managed in a relational database. As in typical databases, a rigid database schema is created as shown in Figure 4(a). From the DTD in Figure 1, all contents in Line (7) can be regarded as attributes of a table in a relational database.

4.2 Schema for Unstructured Data

Secondly, there are unstructured data, which are not tagged by elements nor structured by a rigid schema. The data may be in simply text or in image. Then the schema for such data can be depicted as in Figure ??(b). The label * denotes the label for arbitrary edge.

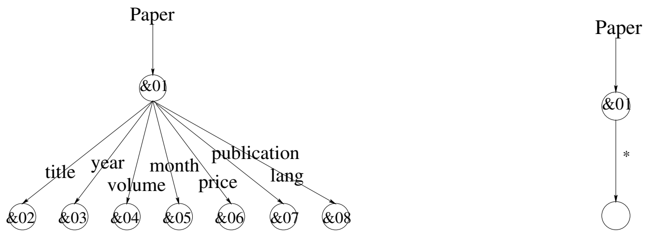


Fig. 4. (a) Structured Data Schema, (b) Un-structured Data Schema

4.3 Schema for Semi-Structured Data

We also consider information sources that are tagged by elements like XML. Although the DTD can be regarded as a schema, there may be several different levels of schemas available.

A data schema is defined to be a *concise*, *accurate*, and *convenient* summary of the structure of a database. We first employ the concept of a so-called DataGuide schema, in Lore [9]. The schemas presented in this paper can be depicted in a directed graph as data are depicted. To achieve conciseness, Lore specifies that a DataGuide describes every unique label path of data instances exactly once, regardless of the number of times it appears in those data instances. To ensure accuracy, Lore specifies that the DataGuide encodes no label path that does not appear in the data instances. For convenience, a DataGuide itself is an object so the object can be stored and accessed by using the same techniques available for processing typical databases. One of the schemas is shown in Figure 4.4(a).

We imagine that although the schema described by [9] is complete enough, it is not yet concise. The DTD is more concise and less complete than that schema. Between these two extreme ends of schemas, there are many gray-level schemas. Levelizing schema is discussed in the next section.

4.4 Schema for External Data

Finally, we also assume that information sources are interconnected and accessible to one another. Data stored in other sites are possibly shipped to a local site and stored in the cache. For example, data in local cache or available in other sites may be depicted as a schema in Figure 4.4(b).

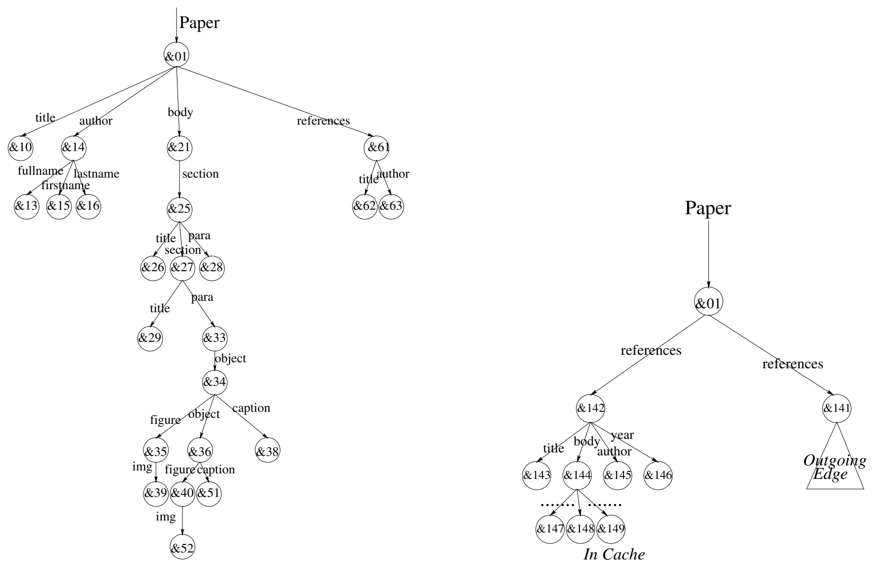


Fig. 5. (a) Semistructured Data Schema, (b) (Assumed) External Data Schema

5 Schema Levelization

Given a schema initially extracted from semi-structured data, gray-level schemas can be generated. DTD is a schema that is most abstracted, while the schema that is extracted from individual data instances is the one that is least abstracted. Between these two extreme schemas, there may exist zero or more gray-level schemas.

We describe two levelization methods of a schema for semi-structured data: 1) topological structure-based levelization and 2) frequency counting-based levelization. Each method uses different information. Structure-based levelization makes use of topological structural information of a schema [9], while counting-based levelization makes use of frequency information of data occurrences. The core concept of counting-based generalization is based upon the frequency of object occurrence. We realized that the topological structure-based levelization is a special case of the frequency counting-based levelization.

5.1 Structure-based Generalization

Each label l of an object x has associated with it a set of subobjects of x. We denote the set of subobjects of x by $\mathsf{DOM}(x.l)$. Also associated with each pair of labels l_1 and l_2 such that $l_1 \prec l_2$ is a *generalization operation*, which is a partial function from $\mathsf{DOM}(x.l_1)$ to $\mathsf{DOM}(y.l_2)$. In Figure 6(a), for example, we see that (a1) is generalized to (a2) if $article = llb(article, paper)$ and $\mathsf{DOM}(x.article) \subseteq \mathsf{DOM}(y.paper)$. We also see that (a1) can be generalized to (a3) if $article = llb(article, paper)$ and $\mathsf{DOM}\ (y.paper) \subseteq \mathsf{DOM}\ (x.article)$. This is a similar case of logical generalization by dropping conditions [12].

5.2 Counting-based Generalization

Let ϵ be a user-given threshold ($0 \leq \epsilon \leq 1$). Consider a label path expression $x.l_1y, y.l_2z$. Notice that the expression $x.l_1$ (a label in the directed graph) is bound to a set of all l-labeled subobjects y of x. Such subobjects y are reached via the element l_1 from the object x. It is likely from object sets y to z through the label l_2. Let the size of $x.l_1$ and $y._2$ be $|x.l_1|$ and $|y.l_2|$, respectively. Notice that $|x.l_1| \equiv |y|$ iff all the objects in y are from x through l_1 only.

If $\frac{|x.l_1|}{|x|} \not\geq \epsilon$, in other words, if sub-elements l_1 do not appear in ϵ percent of data x, then the label l_1 can be removed. We call this generalization *upward generalization.* If all the objects in the set y are incoming from the label l_1, then the object set y can be removed as well.

However, if $\frac{|x.l_1|}{|x|} \not\geq \epsilon$ does not hold and $\frac{|y.l_2|}{|y|} \geq \epsilon$, then the object set y and the label l_2 can be removed. In other words, because more than ϵ percent of the object set y are incoming from the label l_2, representing y only without x turns out to be sufficient within the threshold of ϵ. We call this generalization *downward generalization.* In Figure 6(a), (a2) is downward-generalized from (a1), and (a3) is upward-generalized from (a1).

5.3 Structure-based Aggregation

Each label l of an object x has associated with it a set of subobjects of x. The *aggregation operation* is a partial function from $\mathsf{DOM}(x.l_1)$ to $\mathsf{DOM}(x.l_2)$, associated with each pair of labels l_1 and l_2 such that $l_1 \prec l_2$. In Figure 6(b), for example, we see that (b1) is aggregated to (b2) if $object = lub(figure, picture)$ and $\mathsf{DOM}\ (x.object) = \mathsf{DOM}\ (x.figure) \cup \mathsf{DOM}\ (x.picture)$.

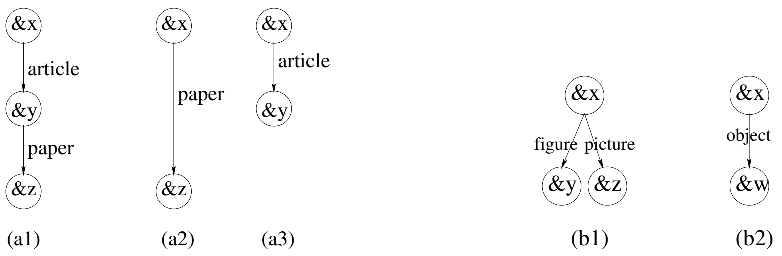


Fig. 6. (a) Generalization Example, (b) Aggregation Example

5.4 Counting-based Aggregation

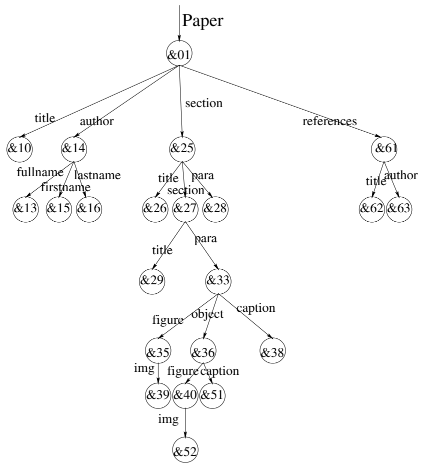
Consider a label path expression $x.l_1y_1, x.l_2y_2$. If $\frac{|x.l_1 \cap x.l_2|}{|x.l_1|+|x.l_2|} \geq \epsilon$, then the two sub-object sets y_1 and y_2 can be aggregated, say as y. This operation can be extended to n number of sub-objects. If $\frac{|x.l_1 \cap x.l_2 \cap \ldots \cap x.l_n|}{|x.l_1|+|x.l_2|+\ldots+|x.l_n|} \geq \epsilon$, then the n sub-object sets can be aggregated. In Figure 6(b), (b1) is aggregated into (b2) by aggregation.

5.5 Example of Multi-level Schemas

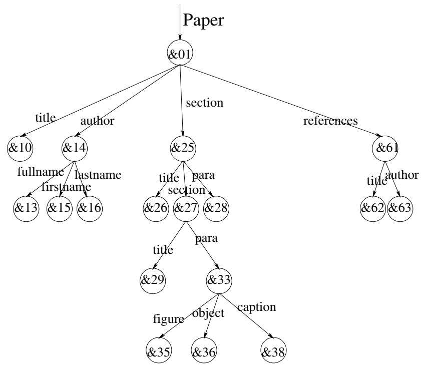
By applying the generalization and aggregation operations, we may have a number of gray-level schemas. One example is in Figure 7. Depending upon the combination of these operations and the user given threshold as stated in the previous subsection, there may be more levels of schemas available. Notice that the schema for the case $\epsilon = 0$ is the one that can be extracted by [9].

6 Integration of Levelized Schemas

Consider two schemas. Each schema is depicted in a directed graph. Let L_1 denote the label expressions in one schema, and L_2 the one in the other schema. If there exists only one $llb(L_1, L_2)$ and there is no ancessor for both schemas, then two labels are simply integrated by being linked to $llb(L_1, L_2)$. In this case, $llb(L_1, L_2)$ is the top ($\top$) node. The aggregation operation, especially the sub-object aggregation, is applied to the labels. If there exists only one $llb(L_1, L_2)$ which is not $\top$, then the two schemas are linked at the object node for $llb(L_1, L_2)$



(a) Generalization $\epsilon = .05$



(b) Generalization $\epsilon = .1$

Fig. 7. Example of Multi-level Schemas

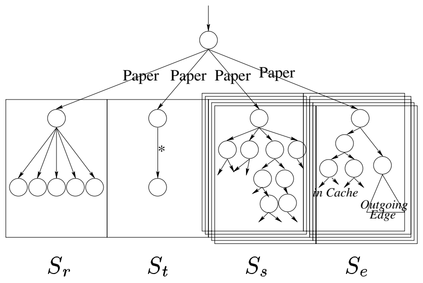


Fig. 8. Integration of Levelized Schemas

followed by applying the super-object aggregation to the ancestor labels in the two schemas.

However, if there exist more than one $llb(L_1, L_2)$, then 1) two schemas are linked at each object node for $llb(L_1, L_2)$; 2) the sub-object aggregation is applied to the descendant labels of the schemas while the super-object aggregation is applied to the ancestor labels; 3) the labels between any two $llb(L_1, L_2)$s can be generalized by the generalization operations if they do not have the same label expressions. This is the most complicated case because the given schemas have no similar topological structure although they have some common object nodes. Note that since the top node is in the lattice of labels, there must be at least one such node in the integration. If there exists no common object node, then we can simply have two schemas linked at the top ($\top$) node as described earlier.

7 Conclusion

This paper described how heterogeneous information sources are modeled together, how schemas can be extracted to reflect well-structured, semi-structured, and unstructured data along with accessing external sources, and how they can be levelized. The schema levelization was developed by taking two information into account: topological structure and frequency counting information. Our work presented in this paper can serve as a pre-processor of a typical query optimization. Further, although we have implemented our algorithms in semi-structured data together with un- or well-structured data, they can easily be adapted to multimedia information sources available on the Internet.

References

1. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. *International Journal of Digital Libraries*, 1(1):68–88, 1997.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Sushil Jajodia, editor, *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 207–216, Washington D.C., 1993.
3. T. Bray, J. Paoli, and C. Sperberg-McQueen. Extensible markup language (XML) 1.0. *World Wide Web Consortium Recommendation. Available at* `http://www.w3.org/TR/REC-xml`, 2 1998.
4. S. Chakrabarti, B Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 307–318, 1998.
5. V. Christophides, S. Cluet, and G. Moerkotte. Evaluating queries with generalized path expressions. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 413–422, Montreal, Canada, 1996.
6. R. Dolin, D. Agrawal, and A. Abbadi. Scalable collection summarization and selection. In *Proc. of ACM Conference on Digital Libraries*, pages 49–58, 1999.
7. M. Fernandez and D. Suciu. Optimizing regular path expressions using graph schema. In *IEEE Data Engineering*, pages 14–23, 1998.
8. D. Florescu, D. Koller, and A. Levy. Using probabilistic information in data integration. In *Proc. Intl. Conf. on Very Large Data Bases*, 1997.
9. R. Goldman and J. Widom. DataGuides: Enabling query formulation and optimization in semistructured databases. In *Proc. Intl. Conf. on Very Large Data Bases*, 1997.
10. R. Goldman and J. Widom. Approximate dataGuides. Technical report, Stanford University, 1999.
11. R. Light and T. Bray. *Presenting XML.* Sams, Indianapolis, Indiana, 1997.
12. R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning*, pages 331–363. Tioga, 1983.
13. J. P. Tremblay and R. Manohar. *Discrete Mathematical Structures with Applications to Computer Science.* McGraw Hill, New York, 1975.

Rough Set Based WebCT Learning

Aileen H. Liang, Brien Maguire, and Julia Johnson

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
{hliang,rbm,johnson}@cs.uregina.ca

Abstract. WebCT is a web-based instruction tool that enables instructors to create and customize their courses for distance post-secondary education. Students do assignments, quizzes, and a final examination on the World Wide Web (WWW). If a student fails the final examination, then the student needs to study the course material again. Questions that arise are "in what areas is the student weak" and "where should the student focus his/her efforts to obtain the necessary background for the next module/section." If the answers to these questions can be found automatically based on the performance of previous students, then students will be able to focus their study and the instructor will be able to reorganize the course material. In this paper, we discuss how to use Rough Sets and Rough Set based Inductive Learning to assist students and instructors with WebCT learning. The scores of quizzes are treated as conditional attributes and the final examination score as a decision attribute. Decision rules are obtained using Rough Set based Inductive Learning to give the reasons for student failure. For repeating students, these rules specify which sections need to be emphasized for the second round. For new students, these rules inform them about those sections requiring extra effort in order to pass the final examination. Hence, Rough Set Based WebCT Learning improves the state-of-the-art of Web learning by providing virtual student/teacher feedback and making the WebCT system much more powerful.

1 Introduction

WebCT stands for Web Course Tools. Developed at the University of British Columbia, it is a web-based instruction tool that enables instructors to create and customize their courses for distance post-secondary education [1,3].

WebCT has become more and more popular as it is both a useful and inexpensive education tool. Universal Learning Technologies (ULT) of Boston recently purchased WebCT and is expending the capabilities. There are more than 1300 colleges and universities in more than 55 countries offering WebCT courses. In 1999, the total WebCT usage was 6 million students over 97,000 WebCT courses. Its popularity is due to the following reasons: WebCT solves students' scheduling conflicts. It allows beginners to work and practice at their own pace. Students can be accepted from different cities or countries and there is no limit on the number of students. Also, people can easily update their knowledge from home by using a web browser. Plus, the cost to the education institute is low. For example, it only costs four US dollars to register a WebCT course account at the University of Regina.

However, WebCT has a deficiency common to distance education delivery [2,4]. The main problem is students do not know what they do not know because there are no lectures given. If a student fails, he/she does not know where the problem is. If students need to repeat the course again, they do not know the section on which to focus on. The other possibility is that the course designer might not know whether the notes are clear or presented in the best possible order as there is no feedback from students other than the informal chat facility in WebCT.

Since the number of WebCT users is rapidly increasing, the performance of WebCT students is becoming a big issue. The self learning ability is important, but if there is no guidance from the WebCT system, students may not obtain satisfactory results. This may dull the enthusiasm for WebCT. While online instruction is good, guided online instruction is better.

To solve the lack of contact problem between students and instructor, some general rules on reasons for failure are needed. Such general rules can advise students based on the rules which apply to them in the course grade history. If students pass, general rules nevertheless advise them on weak areas so they can prepare for future courses because passing does not mean one hundred percent comprehension of the course material. If students fail, general rules inform them of the areas in which they are weak and suggest to them what sections they should focus on if they repeat the course. Student specific information is not as useful to students because it does not tell them which concepts (sections) are needed as prerequisites for other concepts (sections).

Rough Set theory provides an approach to generate the general rules needed by both students and instructors in a web-based learning environment. Students are informed of the reasons in general for success or failure. A general rule is one such as “if a student failed the final examination, then there is a 70% chance that he/she failed Quiz 6” (suppose it is the last quiz). Special cases can be deduced as well. For example, if a student failed Quizzes 1 and 6 and also the final exam, then the student may have failed Quiz 1 because he had a bad day. In fact, Quiz 1 is not the background material needed to pass the final. Quiz 1 may simply be introductory material. Finally, general rules help the instructor to reorganize the material to provide a better prerequisite ordering of the material.

In this paper, we discuss how to use Rough Sets and Rough Set based Inductive Learning to assist students and instructors with the WebCT learning by discovering such general rules based on the performance of previous students.

The paper is organized as follows. Section 2 gives an overview of WebCT. Section 3 introduces Rough Set and Inductive Learning. Section 4 presents our Rough Set based WebCT Learning algorithm. Section 5 concludes the paper.

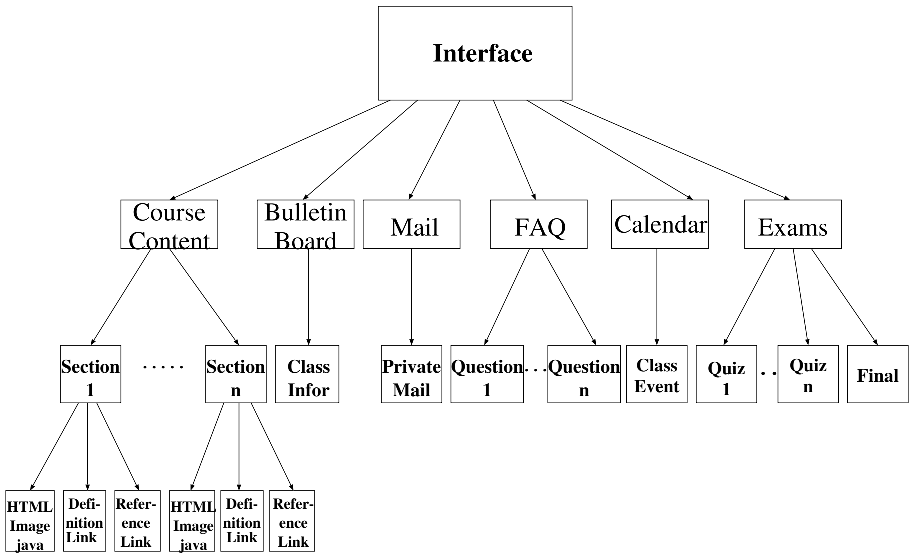


Fig. 1. Hierarchy of WebCT System from Students' Side

2 Overview of WebCT

WebCT consists of WebCT software and a number of hierarchically organized files for each course. The WebCT software resides on and runs from a WebCT server providing an environment to cover all aspects of a course such as tools for creating the course materials, lectures, exercises, quizzes, lab materials, discussion groups, and reference materials.

WebCT provides a great deal of flexibility for students, instructors, and teaching assistants to access WebCT software without the need to download or install any kind of software. WebCT is an interface for designing a course; it provides a set of educational tools to facilitate learning, communication, and collaboration; it provides a set of administrative tools to assist the instructor in the process of management and continuous improvement of the course. It is also password protected.

A Java language WebCT course has been designed at the University of Regina. Figure 1 presents the hierarchy of the WebCT system from the students' point of view based on this WebCT course. When students enter their user ID and password, they come to a homepage that contains six hyperlinks. The first is the *Course Content* link that allows students to access the notes organized by the course designer. The second is the *Bulletin Board* link that is used to post any class information such as assignment or Quiz information. Both the instructor and students can access the *Bulletin Board*. The third hyperlink, *Mail*, allows private email to be sent between student and student and between student and instructor. The fourth hyperlink, *FAQ*, contains frequently asked questions from students. The next link, *Calendar*, marks special events for the course. The last one, *Exams*, links to the quizzes and final examination.

Since the quizzes and final examination are the main focus in this paper, the following gives detailed information on the quizzes and final examination. There are six quizzes, one final examination, and twenty section notes in this Java course. The first section teaches students the editor that is used to write Java programs. The second is an introduction to the Java language. The third introduces variables and data types. The fourth is about the selection statement. The fifth is about the String class. The sixth deals with repetition statements. The seventh and eighth introduce one dimensional and two dimensional arrays respectively. The ninth talks about file access. The tenth introduces objects and classes. The eleventh is about Information Hiding. The twelveth deals with objects within objects. The thirteenth is about Queues. The fourteenth introduces Packages. The sixteenth is about Linked Lists. The seventeenth introduces Inheritance. The eighteenth is about Polymorphism. The nineteenth introduces Applets and the last is about the Graphical User Interface. There is a Quiz after every three or four sections, depending on how each section is related to the others. For example, the materials in Quiz 6 are from section seventeen to twenty covering Inheritance, Polymorphism, Applets, and the Graphical User Interface. These are the core sections in this class.

Figure 2 presents the internal hierarchical structure of the WebCT system that is visible only to the course designer. The course designer creates course materials and course settings by manipulating the hyperlinks in the diagram.

3 Overview of Rough Sets and Inductive Learning

The Rough Set [7] theory was introduced by Zdzislaw Pawlak as a mathematical tool for dealing with vagueness and uncertainty [8]. Given a set E of examples described by an information table T, we classify objects in two different ways: by a subset C of the condition attributes and by a decision attribute D in the information table to find equivalence classes called indiscernability classes $\Omega = \{\Omega_1, ..., \Omega_n\}$. Objects within a given indiscernability class are indistinguishable from each other on the basis of those attribute values. Each equivalence class based on the decision attribute defines a concept. We use $Des(\Omega_i)$ to denote the description, i.e., the set of attribute values, of the equivalence class Ω_i.

Rough Set theory allows a concept to be described in terms of a pair of sets, lower approximation and upper approximation of the class.

Let Y be a concept. The lower approximation $\underline{Y}$ and the upper approximation $\overline{Y}$ of Y are defined as

$$\underline{Y} = \{e \in E \mid e \in \Omega_i \text{ and } X_i \subseteq Y\}$$

$$\overline{Y} = \{e \in E \mid e \in \Omega_i \text{ and } X_i \cap Y = \emptyset\}$$

In other words, the lower approximation is the intersection of all those elementary sets that are contained by Y and the upper approximation is the union of all those elementary sets that are contained by Y.

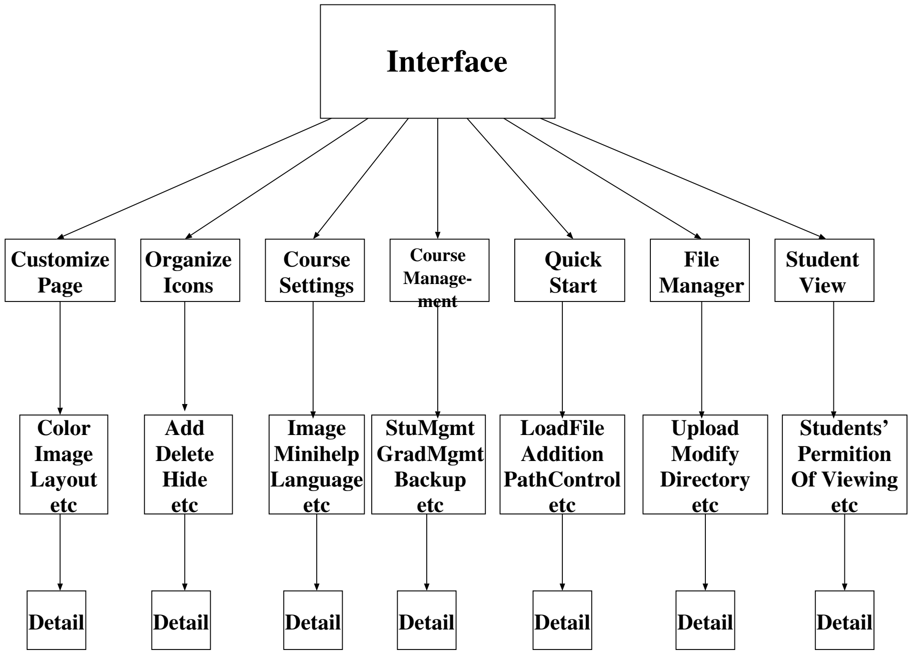


Fig. 2. Hierarchical Structure of WebCT System

Inductive Learning is a well-understood area in artificial intelligence. It is used to model the knowledge of human experts by using a carefully chosen sample of expert decisions and inferring decision rules automatically, independent of the subject of interest [6]. Rough set based Inductive Learning uses Rough Set theory to find general decision rules [12,9]. Their closeness determines the relationship between the set of attributes and the concept.

Many applications of Rough Set based Inductive learning are reported in the literature, for example Automated Extraction of Medical Expert System Rules from Clinical Databases based on Rough Set Theory [10], Discovery of Rules about Complications — A Rough Set Approach in Medical Knowledge Discovery [11], and A Review of Rough Set Models [14]. In [10], a rule induction method is introduced, which extracts not only classification rules but also other medical knowledge needed for diagnosis. It was evaluated on three clinical databases, whose experimental results show that the authors' proposed method correctly induces diagnostic rules and estimates the statistical measures of rules. In [11], the authors explain the problem in a medical context, that is, when a patient suffers from several diseases and has complicated symptoms, making a differential diagnosis is very difficult. Therefore, three models for reasoning about complications are introduced and modeled by using characterization and a rough set model. Finally, in [14], the authors provide a review of the Pawlak rough set model and its extension, with emphasis on the formulation, characterization, and interpretation of various rough set models.

4 Rough Set Based WebCT Learning

WebCT software together, with a number of hierarchically organized files for each of a number of courses, creates a web instruction system that enables an instructor to create and customize courses for distance post-secondary education. We augment the WebCT system to automatically generate rules to help students focus their study and to help instructors reorganize their course materials. The scores of quizzes are treated as conditional attributes and the final examination score as a decision attribute. Decision rules are obtained using Rough Set based Inductive Learning proposed in [12] to give the reasons for student failure.

Before discussing the algorithm, let us first analyze a real student information database, Table 1, from a Java class at the University of Regina in which students used WebCT. In this table, there are 115 students, 6 quizzes, and one final examination. Students receive one of two grades, either pass(p) or fail(f) on each of the components of the course[1].

Table 1 Student Information Database

	Quiz 1	Quiz 2	Quiz 3	Quiz 4	Quiz 5	Quiz 6	FINAL
s_1	p	p	p	p	p	p	p
s_2	p	p	p	p	p	p	p
s_3	p	p	f	f	p	f	f
s_4	p	p	p	p	p	p	p
s_5	f	p	p	p	p	p	p
s_6	p	p	p	p	p	p	p
s_7	p	p	p	p	p	p	p
s_8	p	p	f	p	p	p	p
s_9	p	p	p	p	p	p	p
s_{10}	p	p	p	f	f	f	f
s_{11}	p	p	f	p	p	p	p
s_{12}	p	f	f	p	p	f	f
s_{13}	p	p	p	p	p	f	p
..	..	..	..	..	..	..	..
s_{115}	p	p	p	p	p	p	p

Based on this real database, we combine the students and derive the following collapsed student information table that contains 8 sample classes. The total number of students in the table is 104, other 11 students received failure scores in all areas and do not need to be considered here.

[1] In fact, students' grades are in percentage

Table 2 Collapsed Student Information Table

	Quiz 1	Quiz 2	Quiz 3	Quiz 4	Quiz 5	Quiz 6	FINAL	Total
e_1	p	p	p	p	p	p	p	74
e_2	p	p	f	f	p	f	f	4
e_3	p	f	f	p	p	f	f	3
e_4	p	p	f	p	p	p	p	6
e_5	p	p	f	p	p	f	f	1
e_6	p	p	p	f	f	f	f	5
e_7	f	p	p	p	p	p	p	7
e_8	p	p	p	p	p	f	p	4

By studying this table, we obtain the domain E and two concepts Y_{pass} and Y_{fail} from the decision attribute (FINAL):

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\} \quad Y_{pass} = \{e_1, e_4, e_7, e_8\} \quad Y_{fail} = \{e_2, e_3, e_5, e_6\}$$

As we are only interested in what sections cause students to fail the course, we use the fail concept in our discussion; that is, $Y = Y_{fail}$.

We first find the indiscernbility classes based on Quiz 1 that are $\{e_1, e_2, e_3, e_4, e_5, e_6, e_8\}$ and $\{e_7\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \phi$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_1, e_2, e_3, e_4, e_5, e_6, e_8\}$.

The discriminant index of a concept Y is defined using the following formula:

$$\alpha_{Q_1}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E|$$

Therefore, the discriminant index of Quiz 1 is $\alpha_{Q_1}(Y) = 1 - (97 - 0)/104 = 0.07$ that determines how well the singleton set of attributes consisting of Quiz 1 specifies the membership in Y (the fail concept).

We continue to find indiscernbility classes based on Quiz 2 that are $\{e1, e_2, e_4, e_5, e_6, e_7, e_8\}$ and $\{e_3\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_3\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$. The discriminant index of Quiz 2 is $\alpha_{Q2}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (104 - 3)/104 = 0.03$

The indiscernbility classes based on Quiz 3 are $\{e_1, e_6, e_7, e_8\}$ and $\{e_2, e_3, e_4, e_5\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \phi$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$. The discriminant index of Quiz 3 is $\alpha_{Q6}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (104 - 0)/104 = 0$.

The indiscernbility classes based on Quiz 4 are $\{e_1, e_3, e_4, e_5, e_7, e_8\}$ and $\{e_2, e_6\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_2, e_6\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$. The discriminant index of Quiz 4 is $\alpha_{Q3}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (104 - 9)/104 = 0.09$.

The indiscernbility classes based on Quiz 5 are $\{e_1, e_2, e_3, e_4, e_5, e_7, e_8\}$ and $\{e_6\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_6\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$. The discriminant index of Quiz 5 is $\alpha_{Q5}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (104 - 5)/104 = 0.05$.

The indiscernbility classes based on Quiz 6 are $\{e_1, e_4, e_7\}$ and $\{e_2, e_3, e_5, e_6, e_8\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \phi$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_2, e_3, e_5, e_6, e_8\}$. The discriminant index of Quiz 6 is $\alpha_{Q4}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (14 - 0)/104 = 0.84$.

By comparing the discriminant indices of all quizzes, we know that the discriminant index of Quiz 6 has the highest value $\alpha_{Quiz6}(Y) = 0.84$, that best determines the membership in Y. Therefore, we obtain the first rule:

$$R_1 : \{Quiz6 = f\} \Rightarrow \{Final = f\}$$

Since we know that the Quiz 6 is the most important condition attribute, we then merge this condition attribute with all the other condition attributes to find a new domain and compute new rules. To find the new domain, we first use the following formula to find all elements that are not needed:

$$(E - \overline{Y}) \cup (\underline{Y}) = \{e_1, e_4, e_7\} \cup \phi$$

Why are they no longer needed? By checking against Table 2, we can see that the first set does not involve elements in the fail concept and the second set is empty. Therefore, the new set of elements are:

$$(E - [(E - \overline{Y}) \cup (\underline{Y})] = (E - \{e_1, e_4, e_7\} \cup \phi) = \{e_2, e_3, e_5, e_6, e_8\}$$

We then have the horizontal selection of the collapsed information table (Table 3) where Quiz 1 is omitted as it is a redundant attribute. The total number of students now is 14.

Table 3 Horizontal Selection of Collapsed Table

	Quiz 2	Quiz 3	Quiz 4	Quiz 5	Quiz 6	FINAL	Total
e_2	p	f	f	p	f	f	4
e_3	f	f	p	p	f	f	3
e_5	p	f	p	p	f	f	1
e_6	p	p	f	f	f	f	5
e_8	p	p	p	p	f	p	4

The concepts for this selected information table are $Y_{pass} = \{e_8\}$ and $Y_{fail} = \{e_2, e_3, e_5, e_6\}$. The domain is $E = \{e_2, e_3, e_5, e_6, e_8\}$.

We find the indiscernbility classes based on Quizzes 2 and 6 are $\{e_2, e_5, e_6, e_8\}$ and $\{e_3\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_3\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_2, e_3, e_5, e_6, e_8, \}$. The discriminant index of Quizzes 2 and 6 is $\alpha_{Q2,Q6}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (17 - 3)/17 = 0.18$.

The indiscernbility classes based on Quizzes 3 and 6 are $\{e_2, e_3, e_5\}$ and $\{e_6, e_8\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_2, e_3, e_5\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_2, e_3, e_5, e_6, e_8\}$. The discriminant index of Quizzes 3 and 6 is $\alpha_{Q3,Q6}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (17 - 8)/17 = 0.47$.

The indiscernbility classes based on Quizzes 4 and 6 are $\{e_2, e_6\}$ and $\{e_3, e_5, e_8\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_2, e_6\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_2, e_3, e_5, e_6, e_8\}$. The discriminant index of Quizzes 4 and 6 is $\alpha_{Q4,Q6}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (17 - 9)/17 = 0.53$.

The indiscernbility classes based on Quizzes 5 and 6 are $\{e_2, e_3, e_5, e_8\}$ and $\{e_6\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_6\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_2, e_3, e_5, e_6, e_8\}$. The discriminant index of Quizzes 5 and 6 is $\alpha_{Q3,Q6}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (17 - 5)/17 = 0.29$.

By comparing the discriminant indices, we find that $\alpha_{Q4,Q6}(Y) = 0.53$ best determines the membership in Y. Thus, we obtain the second rule:

$$R_2 : \{Quiz4 = f, Quiz\ 6 = f\} \Rightarrow \{Final = f\}$$

We have found two rules. Let us see if we can find new domain and compute new rules by using the same method used previously.

The elements that are not needed are:

$$(E - \overline{Y}) \cup (\underline{Y}) = \phi \cup \{e_2, e_6\}$$

By checking against Table 3, we can see that the first set is empty and the second set has been handled by rule 2. Therefore, the new set of elements is:

$$E - [(E - \overline{Y}) \cup (\underline{Y})] = \{e_3, e_5, e_8\}$$

Based on this, we obtain the further collapsed information table where Quiz 5 is omitted as it is a redundant attribute.

Table 4 Further Horizontally Collapsed Reduction Table

	Quiz 2	Quiz 3	Quiz 4	Quiz 6	FINAL	Total
e_3	f	f	p	f	f	3
e_5	p	f	p	f	f	1
e_8	p	p	p	f	p	4

The concepts for this further collapsed information table are $Y_{pass} = \{e_8\}$ and $Y_{fail} = \{e_3, e_5\}$ and the domain is $E = \{e_3, e_5, e_8\}$.

The indiscernbility classes based on Quizzes 2, 4 and 6 are $\{e_3\}$ and $\{e_5, e_8\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_3\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_3, e_5, e_8\}$. The discriminant index of Quizzes 2, 4 and 6 is $\alpha_{Q2,Q4,Q6}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (8 - 3)/8 = 0.625$.

The indiscernbility classes based on Quizzes 3, 4 and 6 are $\{e_3, e_5\}$ and $\{e_8\}$. The lower approximation is $\underline{Y} = \bigcup_{\Omega_i \subseteq Y} \Omega_i = \{e_3, e_5\}$. The upper approximation is $\overline{Y} = \bigcup_{\Omega_i \cap Y \neq \phi} \Omega_i = \{e_3, e_5\}$. The discriminant index of Quizzes 3, 4 and 6 is $\alpha_{Q3,Q4,Q6}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (4 - 4)/8 = 1$.

By comparing the discriminant indices of the quizzes, the second one has the higher value. Therefore, we obtain the third rule that combines both discriminant indices:

$$R_3 : \{Quiz3 = f, Quiz4 = p, Quiz6 = f\} \Rightarrow \{Final = f\}$$

We have successfully induced the rules. A question that rises is how much we can believe in the three rules. Since the rules themselves do not tell us this, we need to evaluate the strength of the three rules.

The strength of a rule is commonly measured [8,5] as follows:

$$\frac{\#\ of\ positive\ students\ covered\ by\ the\ rule}{\#\ of\ students\ covered\ by\ the\ rule\ (including\ both\ positive\ and\ negative)}$$

By this definition the first rule above has a strength of $\frac{13}{17}$, that is, 76%. Classes e_2, e_3, e_5, and e_6 from Table 2 are positive examples covered by the rule. Class e_8 is a negative example covered by the first rule. The second rule has a strength of $\frac{9}{9}$, that is, 100%. The third rule has a strength of $\frac{4}{4}$, that is, 100%.

Suppose that we do not have information on Quizzes 3, 4 and 5 for a student who failed the final exam because the student skipped those modules but we do have information that he/she failed Quiz 6. In applying the first rule to this student, there is a 76% chance that the reason for failure was solely the failure of Quiz 6. However, there is a higher probability that the reason for failure was due to additionally failing Quiz 3 or Quizzes 4 because the strength of both the second and third rules is 100%. They are both certain rules. Hence they would be applied in favor of the one of 76% strength.

The instructor also gains information from the rules that would normally be obtained from students in a conventional lecture setting. There is assumed to be an ordering on the modules tested by the quizzes. For example, Quiz 6 requires knowledge of modules tested by at least some of the previous quizzes. Rules induced from the information table might indicate that, in fact, the previous modules are not valuable for preparing the student for a given module.

The WZY algorithm [12] does not fit our WebCT learning situation. The following algorithm is derived from this algorithm. The difference between their algorithm and ours is: their algorithm does not output nondeterministic rule at the intermediate level, but we output both deterministic and nondeterministic rule at the intermediate level because we want to tell students the importance at each step. Additionally, since we are interested in only one concept, the failure concept, and the WZY algorithm covers multiple concepts, we have adapted the algorithm to a special case by removing the loop to handle multiple concepts.

```
INPUT: Concept fail, Domain E, Set of condition attributes C,
ALGORITHM:
Step 1. LET $E' = E$, $C' = C$, $Q = \emptyset$, and Y be the concept;
Step 2. WHILE $(C' \neq \emptyset)$ {
    1) COMPUTE the set of discriminant indices
       $\{\alpha_{Q'}(Y) \mid Q' = Q \cup \{c\}, \forall c \in C'\}$;
    2) SELECT the set of attributes $Q' = Q \cup \{c\}$ with the highest
       values $\alpha_{Q'}(Y)$
    3) LET $Q = Q'$;
    4) IDENTIFY those equivalence classes: $\{X_1, ..., X_r\}$ contained by $\underline{Y}$;
    5) OUTPUT deterministic or non-deterministic decision rules:
       $\{Des(X_k) \Rightarrow Des(Y) \mid k = 1, 2, ..., r\}$;
    6) LET $E' = E' - [(E' - \overline{Y}] \cup \underline{Y}]$
    7) IF $(E' \neq \phi)$
           LET $C' = C' - Q$;
       ELSE
           STOP;
    }
Step 3. Output deterministic or non-deterministic decision rules:
    $\{Des(X'_i) \Rightarrow Des(Y) \mid \forall$ equivalence class $X'_i$ of the relation C' on E'$\}$
END ALGORITHM.
OUTPUT: Decision rules for the concept.
```

5 Conclusion

This paper demonstrates that by using Rough Set based Inductive Learning methods, the lack of student/teacher feedback with current WebCT can be overcome, making Web based learning more powerful. Rough Set Based WebCT Learning permits decision rules to be induced that are important to both students and instructors. It thus guides students in their learning. For repeating students, it specifies the areas they should focus on according to the rules applied to them. For new students, it tells them which sections need extra effort in order to pass the course. For example, if Quiz 6 is failed, there is a 76 percent chance that the student will fail the final. Therefore, students are made aware of the importance of Quiz 6 related materials. Rough Set Based WebCT can also guide the instructor about the best order in which to present the notes. Based on this analysis, the instructor may reorganize or rewrite the course notes by providing more examples and explaining more concepts.

Rough Set Based Inductive Learning improves the state-of-the-art of Web learning by providing virtual student/teacher feedback and making the WebCT system much more powerful.

In the present work, we just distinguish two values: pass and fail. It would be useful to have more values such as grade levels of A, B, C, D and F. Identifying near passes on quizzes would add another useful attribute. Some relevant works in this respect include [13].

Acknowledgments

The research was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant OGP0151850. Thanks to Wojciech Ziarko and Yiyu Yao for their valuable comments and suggestions. Thanks also to Emily Schudel, the WebCT developer extraordinaire.

References

1. Getting Started Tutorial for WebCT Version 1.2. http://www.webct.com/. 425
2. J. Carrasquel. Teaching CS1 On-Line: the Good, the Bad, and the Ugly. In *30th SIGCSE Technical Symposium on Computer Science Education*, pages 212–216, New Orleans, Lousiana, 1999. 426
3. Web Developer. Using WebCT. http://www.ualberta.ca/webct/tutorials, 1999. 425
4. K. Jarvinen, J. J. Kyaruzi, and E. Sutinen. Between Tanzania and Finland: Learning Java over the Web. *Proceedings of ACM SIGCSE*, 38(11):217–221, 1999. 426
5. J. Johnson and M. Liu. Rough Sets for Informative Question Answering. In *Proceedings of the International Conference on Computing and Information (ICCI '98)*, pages 53–60, Winnipeg, Canada, June 17-20 1996. 434
6. J. A. Johnson and G. M. Johnson. Student Characteristics and Computer Programming Competency: A Correlational Analysis. *Journal of Studies in Technical Careers*, 14:23–92, 1992. 429
7. Z. Pawlak. *Rough Sets*, pages 3–8. Kluwer Academic Publishers, 1997. 428
8. Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko. Rough sets. *Communications of the ACM*, 38(11):89–95, 1995. 428, 434
9. N. Shan, W. Ziarko, H. J. Hamilton, and N. Cercone. Using rough sets as tools for knowledge discovery. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 263–268. AAAI Press, 1995. 429
10. S. Tsumoto. Automated Induction of Medical Expert System Rules from Clinical Databases based on Rough Set Theory. *Information Sciences*, 112:67–84, 1998. 429
11. S. Tsumoto. Discovery of Rules about Complications-A Rough Set Approach in Medical Knowledge Discovery. In *Proceedings* 7^{th} *International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing (RSFD-GrC'99)*, pages 29–37. Springer LNAI 1711, 1999. 429
12. S. K. M. Wong, W. Ziarko, and R. L. Ye. Comparison of rough-set and statistical methods in inductive learning. *International Journal of Man-Machine Studies*, 24:52–72, 1986. 429, 430, 434
13. Xindong Wu and David Urpani. Induction by Attribute Elimination. *IEEE Transaction on Knowledge and Data Engineering*, 11(5):805–812, 1999. 436
14. Y. Y. Yao, S. K. M. Wong, and T. Y. Lin. *A Review of Rough Set Models*, pages 47–76. Kluwer Academic Publishers, 1997. 429

MultiAgent Systems for Coalition Support

Zakaria Maamar[1,*] and Nader Troudi[2]

[1] Information System Technology Section, Defence Research Establishment Valcartier
Val-Bélair, QC G3J 1X5, Canada
`zakaria.maamar@drev.dnd.ca`

[2] Computer Science Department, Laval University
Ste-Foy, QC G1K 7P4, Canada
`troudi@iad.ift.ulaval.ca`

Abstract. The paper deals with the design and development f a coalition infrastructure. Generally, this infrastructure's nature is distributed, heterogeneous, and dynamic. Therefore, new advanced technologies are required. MultiAgent Systems (MASs) seem to be a good candidate and a promising approach. For instance, MASs could provide an insight into what aspects of the coalition should be delegated, how these aspects should evolve, and what should be the effect of specific changes. This research is part of the IC2MAS project that takes advantages of several research domains, including software agents, mobility, and ontology.

1 Introduction

DREV's Information System Technology Section is divided into three research groups, among these we concentrate on the Interoperability group. This group's interests are the design and development of collaborative environments for distributed and heterogeneous military applications that are called *Command & Control Information Systems* (CCISs). CCISs are increasingly important for land, naval, and air operations. Moreover, CCISs have civilian applications in multiple areas such as air traffic control, search & rescue, and emergency services. In a military context, a commander takes decisions concerning his troops deployment using the information supplied by the CCIS. It may occur that this commander aims at requesting other friendly CCISs before taking his decisions. For example, a Canadian commander has to take into account information on the positions of the enemy and friendly troops. Therefore, he has to involve other CCISs that may possess this information. It would be more appropriate if this commander could perform such an operation without being aware of each CCIS's characteristics. Take for instance a situation where different countries decide to set up a *coalition* for an international humanitarian-assistance. In fact, the CCIS of each country has its own functional and structural characteristics. Hence, it becomes impossible for a commander to be aware of all the CCISs' locations, languages, and information semantics. Therefore, it becomes urgent

[*] The different aspects presented in this article are those of the main author and should not be interpreted as representing the official policies of DREV.

to propose new support technologies that will free military users from worrying about the distributed, heterogeneous, and dynamic nature of the coalition, in general, and CCISs in particular. In this paper, we describe the IC2MAS (*Interoperable Command & Control based on MultiAgent Systems*) project that aims at managing the coalition infrastructure at the following levels: autonomy, adaptability, and scalability. Taking into account these three levels, the IC2MAS project has established an *interoperability* approach to provide effective support to the coalition infrastructure.

The motivation behind the support of a coalition infrastructure is to provide an integrated view of all the aspects that are relevant to this coalition. These aspects are multiple and include: the coalition structure; the roles played by and responsibilities held by people within the coalition; the flow of information within the coalition and with the external environment; the capabilities required by or available within the coalition; and the context (war, peace-making/keeping, from war to peace-making, and from peace-keeping to war) in which the coalition takes place. A number of these aspects could be handled by MASs. For instance, the coalition structure could be viewed as a collection of collaborative MASs; each MAS could correspond to a CCIS and each MAS could contain several agents of different types, with different roles, and with different responsibilities.

In the IC2MAS interoperability approach, MASs are the CCIS's front-ends to the coalition network and hence, have the capability to act on their behalf. Moreover, MASs encompass different *Software Agents* (SAs) [3] that handle and perform the functionalities required to coalition support, for example CCIS's autonomy management. However, given the distributed nature of a coalition and the network features in terms of reliability and bandwidth capacity (e.g. the coalition could occur in a country in which the network infrastructure is not well developed), some of the SAs are enhanced with *mobility* mechanisms [7]. A mobile agent can move from one system to another to perform specific operations, instead of continuously keeping the network "busy". Moreover, it often happens that SAs have to work together to perform common operations. For instance, in a coalition, the Canadian forces have to interact with non-government organizations as well as with armed forces of other countries. Therefore, SAs have to rely on *communication* [6] and *coordination* [2] mechanisms to avoid conflicts and collaborate effectively (i.e. doing the right things) and efficiently (i.e. doing the things right). When SAs communicate, they have to understand each other. By establishing an *ontology* [4], a common terminological and semantical basis for the various SAs is offered. Hence, the risks of getting inconsistent information is reduced.

The remainder of the paper is as follows. Section 2 presents an overview of the CCISs field. Section 3 describes the characteristics of the IC2MAS interoperability approach. Section 4, briefly, reviews the related work. Section 5 gives insights on topics that are currently, tackled. Finally, Section 6 consists of concluding remarks.

2 An Overview of CCISs

According to [9], a CCIS consists of a structure, functions, and tasks. The CCIS structure represents an assembly of facilities, arranged to meet the CCIS's objectives. To reach these objectives, the CCIS's functions are initiated in order to carry out the needed tasks. Tasks require the structure's facilities, in terms of personal, technical equipment, computing time, etc. Figure 1 presents a simplified architecture of a CCIS. Several types of functions exist within the CCIS, ranging from planning and weather forecast to data fusion. These functions are offered to users and are built on top of a support structure in terms of hardware and software resources. Furthermore, some of these functions receive messages from the external environment, e.g. remote sensors, through a communication module. Currently, multiple definition languages of messages, e.g. USMTF for United States Message Text Format, are available. These languages allow to format messages in order to be automatically parsed by appropriate engines of the different functions. Unfortunately, such languages cannot be used in the achievement of interoperable CCISs. These languages' structures are too rigid and do not have semantics.

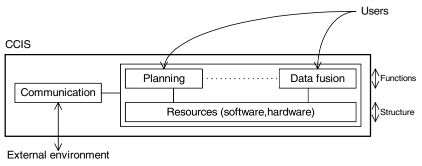


Fig. 1. CCIS Simplified Architecture

As CCISs are getting larger and more complex, their interoperability and hence the commandment-level collaboration, in a coalition context for example, are becoming a central concern for military users and CCISs' designers.

3 Presentation of the IC2MAS Approach

This section presents the IC2MAS interoperability approach. First, IC2MAS's architecture and types of SAs are presented. Then, IC2MAS's functioning is detailed.

3.1 IC2MAS's Architecture

In the literature, different approaches that deal with the problem of interoperable systems can be found [1,5,8]. All these approaches agree on the use of SAs as a means to develop such systems and have several elements in common, such as all the SAs are static. Therefore, these SAs do not have the opportunity to move to distant systems. Furthermore, all these approaches assume that the network infrastructure is fully reliable and has unlimited bandwidth for information transmission.

Based on these different approaches and the coalition's requirements, we proposed an IC2MAS's architecture to the coalition infrastructure (Fig. 2). Multiple MASs form the backbone of this architecture. These MASs interact about their respective CCISs by exchanging messages, either remotely or locally. In both cases, a facility called *Advertisement Infrastructure* is required. This infrastructure is managed by an agent and contains a *Bulletin Board* and a *Repository of Active-Agents*. We are aware that the Advertisement Infrastructure may be considered as a bottleneck. However and in the mid-term, this infrastructure will be duplicated and spread across networks.

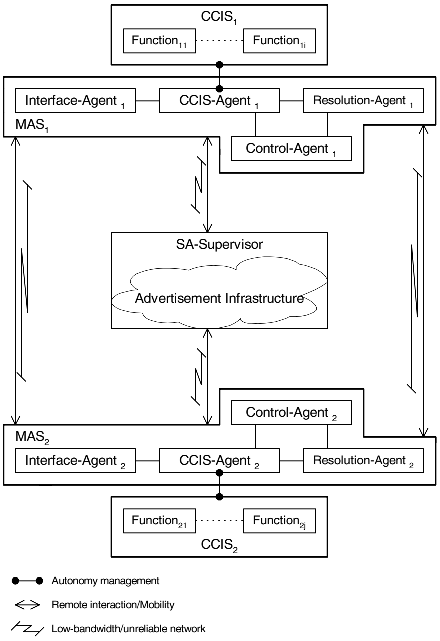


Fig. 2. IC2MAS's Architecture for coalition support

In the IC2MAS's architecture, MASs consist of different types of SAs: *Interface-Agents* assisting users, *CCIS-Agents* invoking CCISs' functions and satisfying users' needs, *Resolution-Agents*, also, satisfying users' needs, *Control-Agents* managing MASs, and finally, a *Supervisor-Agent* managing the Advertisement Infrastructure. Interface-Agents, Control-Agents, and Supervisor-Agents are static while CCIS-Agents and Resolution-Agents are mobile. They can "hop" to the Advertisement Infrastructure. Further, the Resolution-Agent can "hop" to other MASs.

3.2 Software Agents and Advertisement Infrastructure

Different types of SAs exist in the IC2MAS's architecture. These SAs belong to different MASs and collaborate through the Advertisement-Infrastructure facility.

Interface-Agent - By analogy to Interface-Agents of [8], the IC2MAS's Interface-Agent assists users in formulating their needs, maps these needs into requests, forwards these requests to the CCIS-Agent in order to be processed, and provides these users with answers obtained from the CCIS-Agent.

CCIS-Agent - By analogy to Resolution-Agents of [8], the IC2MAS's CCIS-Agent processes users' requests, only if these requests require the involvement of the CCIS of this particular CCIS-Agent. These requests are transmitted by the Interface-Agent. Moreover, and by analogy to Knowledge-Agents of [8], the IC2MAS's CCIS-Agent acts on CCIS's behalf and hence, maintains its autonomy and independence towards the coalition. For this end, the CCIS-Agent advertises, through its services (currently, the services do not have constraints, e.g. cost), the functions its CCIS performs. Here, the term service denotes a computing procedure, for example requesting the CCIS's weather-forecast function. In the IC2MAS environment, a CCIS-Agent has the ability to advertise its services, by *posting* notes on the Bulletin Board of the Advertisement Infrastructure. In order to post notes, the CCIS-Agent can either send a remote request to the Supervisor-Agent or migrate to this infrastructure. A decision about a remote request or mobility is based on the network status. In both cases, i.e. remote request or mobility, a *security level* is associated with the CCIS-Agent. This security level is used to identify the services this CCIS-Agent is authorized to advertise.

A CCIS offers different functions that vary from data fusion and weather forecast to planning (Sec. 2). Based on these functions and the complex nature of CCISs, for instance a planning function could be a distributed-objects client/server application that runs on top of an Object Request Broker middleware, new types of SAs, called *Function-Agents*, are introduced in the IC2MAS's architecture, and particularly at the MAS level. Each Function-Agent is associated with a CCIS's function. As a result, a CCIS-Agent manages a group of Function-Agents that evolve under its supervision. For instance, a request to the planning function of a CCIS is initially, sent to the CCIS-Agent that forwards this request to the appropriate Function-Agent. Hence, a Function-Agent knows the protocols through which a function of a CCIS accepts requests and provides back results.

Resolution-Agent - By analogy to Resolution-Agents of [8], the IC2MAS's Resolution-Agent processes users' requests, only if these requests are transmitted by the CCIS-Agent and need the involvement of several CCISs to be completed. In this situation, the resolution process requires that this Resolution-Agent collaborates with CCIS-Agents of other MASs. Thus, the Resolution-Agent moves to the Advertisement Infrastructure, consults the Bulletin Board, identifies appropriate CCIS-Agents through their offered services, goes back to its original MAS, and finally, designs the procedure needed to the performance of the user's request. Generally, this procedure is called a *route* or an *itinerary*. Therefore, the resolution process may require from this Resolution-Agent either to remotely interact with the CCIS-Agents of the other MASs or to migrate to the MASs and meet locally their CCIS-Agents. A decision about a remote request or mobility is based on the network status and the number of the CCISs required to satisfy users' requests. As the CCIS-Agent, a security level is also associated with the Resolution-Agent. This security level is used to check Resolution-Agents entering the Advertisement-Infrastructure as well as the different MASs.

Control-Agent - In an environment consisting of mobile agents, mobility operations consist of shipping the agents through the net to other distant systems, authenticating these agents as soon as they arrive, and finally installing these agents to resume their operations. In the IC2MAS environment, the Control-Agent of the MAS is in charge of all these operations. For instance, when the Resolution-Agent decides to move, it first interacts with the Control-Agent in order to be shipped to the desired MAS. Furthermore, Control-Agents maintain the coherence of their MASs by keeping track of the Resolution-Agents entering and leaving these MASs.

Supervisor-Agent - A Supervisor-Agent is in charge of several operations. It manages the Advertisement Infrastructure by receiving CCIS-Agents' advertisements, sets up a security policy in order to monitor the CCIS-Agents and Resolution-Agents accessing this infrastructure, and finally, installs CCIS-Agents and Resolution-Agents to resume their operations in this infrastructure. In the IC2MAS environment, the Supervisor-Agent uses the Repository of Active-Agents to register all the CCIS-Agents and Resolution-Agents that have got an agreement to enter and leave the Advertisement Infrastructure.

Advertisement Infrastructure - In a coalition context, CCISs are spread across networks and generally rely on low-bandwidth and/or unreliable channels for communications. Moreover, a military user may use his VHF Combat Net Radio to send and request information. In addition, this military may rely on mobile devices, such as portable computers, that are only intermittently connected to networks. In the IC2MAS environment, instead of overloading the network, CCIS-Agents and Resolution-Agents migrate to the Advertisement Infrastructure in which:

- CCIS-Agents advertise their services by posting notes on the Bulletin Board.
- Resolution-Agents consult the Bulletin Board to identify the CCISs that are required to satisfy users' needs.

3.3 IC2MAS's Functioning

Based on the characteristics of the IC2MAS's architecture and the types of SAs this architecture integrates, we proposed four stages to handle the IC2MAS's functioning: *Initialization*, *Advertisement*, *Operation*, and *Maintenance*. In what follows, the features of each stage are described. Note that Initialization and Advertisement stages are transparent to users.

Initialization Stage - This stage is characterized by the following operations:

1. The Advertisement Infrastructure and its components, i.e. Supervisor-Agent, Bulletin Board, and Repository of Active-Agents, are set up. Then, the Supervisor-Agent initializes the Bulletin Board and the Repository. Furthermore, this agent initiates the security policy that manages agents' accesses to the Advertisement Infrastructure.
2. MASs are set up and associated with their respective CCISs. For example, the CCIS-Agent and its group of Function-Agents constitute the front-end of the CCIS's functions to the coalition network. Furthermore, this CCIS-Agent is associated with the services to advertise and hence, to offer to other MASs.

In what follows, we assume that, before leaving and entering MASs, CCIS-Agents and Resolution-Agents interact with Control-Agents for security, shipping, and installment purposes.

Advertisement Stage - Once the initialization stage is done, CCIS-Agents advertise their services at the Advertisement-Infrastructure level. As stated in Section 3.2, a CCIS-Agent can either send a remote request to the Supervisor-Agent of the Advertisement Infrastructure or migrate to this infrastructure (Fig. 3, solid and dashed lines). This decision is based on the network status.

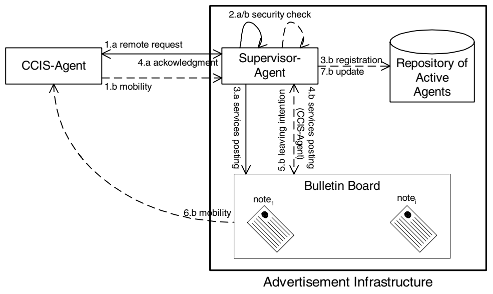


Fig. 3. Services advertisement in the Bulletin Board

Remote as well as mobility situations describe the behavior of a CCIS-Agent at the advertisement stage. In what follows, both situations are described. Numbers in parenthesis correspond to numbers in Figure 3.

– Situation a: to offer its services to other MASs, the CCIS-Agent sends a remote request to the Supervisor-Agent (1.a). Depending on the security level of this CCIS-Agent and the security policy of the Advertisement Infrastructure, the Supervisor-Agent decides if this CCIS-Agent is authorized to advertise and what services (2.a). In the positive case, the Supervisor-Agent processes the CCIS-Agent's request by posting the services it offers on the Bulletin Board (3.a). At the end, the Supervisor-Agent sends an acknowledgement message to the CCIS-Agent about the success of the operation (4.a). We assume that the CCIS-Agent sends only one request in order to advertise all the services it offers. Moreover, we assume that other requests, that update or withdraw the advertised services, will follow.

– Situation b: to offer its services to other MASs, the CCIS-Agent moves to the Advertisement Infrastructure (1.b). Before entering this infrastructure, the CCIS-Agent is checked and authenticated according to its security level and the infrastructure's security-policy. This operation is processed by the Supervisor-Agent (2.b). Before posting its services on the Bulletin Board (4.b), the CCIS-Agent is registered (3.b) in the Repository of Active-Agents. Once the CCIS-Agent has completed its operations, it notifies the Supervisor-Agent about its intention to leave the Advertisement Infrastructure (5.b) and goes back to its original MAS (6.b). Finally, the Supervisor-Agent updates the Repository of Active-Agents for coherence sake (7.b).

Operation Stage - Once the advertisement stage is done, the IC2MAS environment is ready to be operated. The operation stage of IC2MAS is summarized by two situations (Fig. 4):

1. Only the user's CCIS is required: the CCIS-Agent is in charge of handling this situation (Fig. 4, Situation a).
2. Several CCISs, including or not the user's CCIS, are required: the Resolution-Agent is in charge of handling this situation (Fig. 4, Situation b).

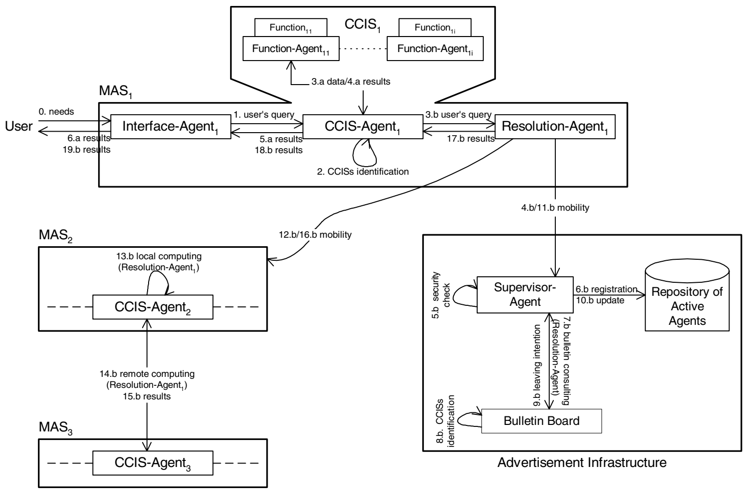


Fig. 4. User's request satisfaction

In what follows, numbers in parenthesis correspond to numbers in Figure 4 and illustrate operations chronology.
When a user wants to satisfy his needs (0), he interacts with the Interface-Agent of his MAS. Next, his needs are mapped into a request transmitted to the CCIS-Agent (1). This agent is in charge of deciding whether this user's CCIS contains the appropriate functions to process the user's request (2). Once this decision is obtained, two situations exist and are identified in Figure 4 as letters a and b.
In Situation a, the CCIS-Agent forwards the user's request to the appropriate Function-Agent (3.a) of the user's CCIS. This Function-Agent initiates the CCIS's function and provides the results it obtained to the CCIS-Agent (4.a). Next, results are sent to the user through the Interface-Agent (5.a, 6.a).
In Situation b, other CCISs, including or not the user's CCIS, are required to satisfy the user's request. These CCISs are identified using the Bulletin Board of the Advertisement Infrastructure. First, the CCIS-Agent forwards the user's request to the Resolution-Agent (3.b). Then, the Resolution-Agent moves to the Advertisement Infrastructure (4.b). Before entering this infrastructure, the Resolution-Agent is checked and authenticated according to its security level and the infrastructure's security policy. This operation is done by the Supervisor-Agent (5.b). Before the Resolution-Agent starts to consult the Bulletin Board (7.b), this agent is registered by the Supervisor-Agent (6.b) in the Repository of Active-Agents. Once the Resolution-Agent has completed its operations, i.e. once it has identified the CCIS-Agents with which it is going to interact (8.b), the Resolution-Agent informs the Supervisor-Agent about its intention to leave the Advertisement Infrastructure (9.b). Then, the Supervisor-Agent updates its Repository (10.b). Once the Resolution-Agent arrives to its original MAS (11.b), it starts to design its itinerary according to the number of the appropriate CCISs and the network status. To clarify things, hereafter is an example illustrating this itinerary. In Figure 4, the itinerary indicates that the Resolution-Agent first has to move to a MAS (12.b), for instance MAS_2. Next, the Resolution-Agent interacts locally with the CCIS-Agent of this MAS (13.b). Furthermore, to complete its operations, the itinerary mentions that the Resolution-Agent has to remotely interact with other CCIS-Agents, for instance CCIS-$Agent_3$ of MAS_3. Then, the Resolution-Agent sends a request (14.b) and waits for the results from CCIS-$Agent_3$ (15.b). At the end, the Resolution-Agent goes back to its original MAS (16.b) and sends the results it obtained to the user through the CCIS-Agent (17.b) and the Interface-Agent (18.b, 19.b).

Maintenance Stage - The IC2MAS environment is an open system. Indeed, a new CCIS can be integrated, another CCIS can be removed, etc. Therefore, the purpose of the maintenance stage is to take into account the situations that may have an impact on the architecture of the IC2MAS environment as well as on its functioning. Several situations have been identified. In this paper, we briefly present two of them:

- It happens that a CCIS adapts its structural as well as functional characteristics, for example by adding a new function, discarding a service,

or upgrading the version of a function's database management system. Therefore, the CCIS-Agent has to be adapted either by adding new services to its capabilities, canceling services, or updating its services. Further, the CCIS-Agent has to interact with the Advertisement Infrastructure.
– It happens that the Supervisor-Agent cleans up the Bulletin-Board of the Advertisement Infrastructure, because of a new security policy, for example. Hence, CCIS-Agents have to advertise their services from the beginning.

4 Related Work

This section summarizes the main characteristics of the IC2MAS environment with respect to other similar works. There exist different research projects in the field of systems interoperability [1,5,8]. All these projects have the same concerns, namely:

– Maintain the autonomy and independence of the systems to be integrated in an interoperable environment. In the IC2MAS environment, each CCIS has been associated with a CCIS-Agent that acts on its behalf.
– Reduce the informational disparities between the integrated systems. In the IC2MAS environment, the definition of an ontology is, currently, tackled (Sec. 5.1).
– Help users satisfy their needs. In the IC2MAS environment, each MAS integrates an Interface-Agent that assists users.

However, all the projects cited above assume that the network infrastructure is fully reliable and has unlimited bandwidth for information transmission. In a coalition, this is not the case. In the IC2MAS environment, network concern has been considered, for instance by enhancing certain agents with mobility mechanisms and giving these agents the ability to decide whether local computing after a move is preferable than remote computing. Furthermore, security issues have been considered in the IC2MAS environment, by suggesting a security policy to manage the Advertisement Infrastructure and a security level to identify agents. Additional security elements could be suggested, such as identifying services with authorization levels and users with use levels.

5 IC2MAS's Current Efforts

This section gives insights on topics that are currently, tackled, in the IC2MAS environment. Among these topics, we describe, briefly, the ontological disparities and the implementation strategy.

5.1 Ontology

Ontology is one of the main issues to be addressed in the design of a collaborative environment for heterogeneous systems. We consider an ontology as a means to represent and exchange information that are understood by all participants.

In a coalition context, each country has its own standards. Therefore, each military user specifies his needs, in term of *information requests*, and his CCIS's capabilities, in term of *functions*, using these standards. Therefore, the need to define two types of specification languages is raised in the IC2MAS interoperability approach. The first type is *a specification language for users' needs* while the second type is *a specification language for CCISs' functions*. Both of these languages have to be based on two different ontologies: a *user-oriented ontology* and a *CCIS-oriented ontology*. Furthermore, because of the coalition context, the user-oriented ontology has to be adapted in order to take into account the individual differences, for example diversity of cultures, that exist between the coalition's participants. To handle these characteristics, we intend to propose a user-oriented ontology that is "versioned" (certain authors talk about ontology sharing). Hence, only one user-oriented ontology is defined at the conceptual level but different versions of this ontology are defined at users level.

5.2 Implementation

Work is currently undertaken in order to demonstrate the viability of the IC2MAS environment and its multi-agent based approach to coalition support. Three issues related to this approach are addressed: How to implement IC2MAS's agents? How to allow IC2MAS's agents to move? And how to support IC2MAS's agents in their exchange of messages?

IC2MAS's agents would be implemented as Java classes. We plan to use the development environment Visual Café Pro from Symantec Inc. Moreover, because certain IC2MAS's agents have the ability to move, we plan to use the ORB Voyager from the ObjectSpace Company as a support middleware for these agents' moves. Finally, messages between agents could be based on the KQML language. We plan to use three interconnected PC Windows NT and 98 stations to simulate the IC2MAS environment. The NT station would contain the Advertisement Infrastructure and its components.

6 Conclusion

In this paper, we presented the major characteristics of the IC2MAS interoperability approach that uses MASs in the design of collaborative environments for distributed and heterogeneous CCISs. In the coalition context, MASs and their SAs are able to fulfill different operations, from users' needs specification to CCISs' functions initiation. Six types of SAs exist in the architecture proposed to coalition support (Interface-Agent, CCIS-Agent, Resolution-Agent, Control-Agent, Function-Agent, Supervisor-Agent) while four stages describe this architecture functioning (Initialization, Advertisement, Operation, Maintenance).

Whereas MASs appear to offer benefits to coalition support, we must be aware of their limitations. MASs must allow a large degree of human interaction. Therefore, it is unrealistic to expect to be able to provide a "fully" automated coalition support. A whole set of negotiations, dialogues, coordination and communication between participants, groups of participants, and systems are involved.

References

1. Bayardo, R. and Bohrer, W. and Brice, R. and Cichocki, A. and Fowler, G. and Helai, A. and Kashyap, V. and Ksiezyk, T. and Martin, G. and Nodine, M. and Rashid, M. and Rusinkiewicz, M. and Shea, R. and Unnikrishnan, C. and Unruh, A. and Woelk, D. Infosleuth: Semantic integration of information in open and dynamic environments. In *Proc. of the 1997 ACM International Conference on the Management of Data (SIGMOD)*, Tucson, Arizona, 1997. 439, 446
2. Ghenniwa, H. *Coordination in Cooperative Distributed Systems.* PhD thesis, Systems Design Engineering Department, University of Waterloo, Ontario, Canada, 1997. 438
3. Green, S. and Hurst, L. and Nangle, B. and Cunningham P. and Somers, F. and Evans, R. Software agents: A review. Technical report, Trinity College Dublin and Broadcom Éireann Research Ltd., May 1997. 438
4. Jones, D. and Bench-Capon, T. and Visser, P. Methodologies for ontology development. In *the proceedings of IT&KNOWS - Information Technology and Knowledge Systems, 15th IFIP World Computer Congress*. Vienna (Austria) and Budapest (Bulgaria), 1998. 438
5. Knoblock, C.-A. and Arens, Y. and Hsu. C.-N. Cooperating agents for information retrieveal. In *Second International Conference on Cooperative Information Systems*, Toronto, Canada, 1994. 439, 446
6. Labrou, Y. and Finin, T. and Peng, Y. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45–52, March/April 1999. 438
7. Lange, D. and Oshima, M. Dispatch your agents; shut off your machine. *Communication of the ACM*, 42(3):88–89, March 1999. 438
8. Maamar, Z. and Moulin, B. and B dard, Y. Software agent-oriented frameworks for the interoperability of georeferenced digital libraries on the world wide web: The sigal project. In R. Fegeas M. F. Goodchild, M. J. Egenhofer and C. A. Kottman, editors, *Interoperating Geographic Information Systems*, pages 335–354. Boston: Kluwer Academic Publishers, 1999. 439, 441, 446
9. Malerud, S. and Feet E. H. and Thorsen, U. A method for analysing command and control systems. Norwegian Defence Research Establishment (FFI) N-220 Kjeller, Norway. 439

A Fast Globally Supervised Learning Algorithm for Gaussian Mixture Models

Jiyong Ma and Wen Gao

Institute of Computing Technology, Chinese Academy of Sciences, China

Email: {jyma, wgao}@ict.ac.cn

Abstract. In this paper, a fast globally supervised learning algorithm for Gaussian Mixture Models based on the maximum relative entropy (MRE) is proposed. To reduce the computation complexity in Gaussian component probability densities, the concept of quasi-Gaussian probability density is used to compute the simplified probabilities. For four different learning algorithms such as the maximum mutual information algorithm (MMI), the maximum likelihood estimation (MLE), the generalized probabilistic descent (GPD) and the maximum relative entropy (MRE) algorithm, the random experiment approach is used to evaluate their performances. The experimental results show that the MRE is a better alternative algorithm in accuracy and training speed compared with GPD, MMI and MLE.

1 Introduction

The conventional Gaussian Mixture Models (GMM)[1] or EM algorithm [2,3], on-line EM algorithm [4,5] are powerful approaches for classification. One way to estimate parameters in the GMM is based on locally unsupervised learning. However, The locally unsupervised learning algorithm for estimating parameters doesn t take account of the sample distributions of other classes. The model discriminative capacity is not ideal when the training samples are insufficient, while these parameters can be used as predictors. Theoretically, if training samples are sufficient enough these parameters are optimal in the sense of maximum likelihood. The training samples are not infinite due to the constraints of experimental conditions in practical applications. Furthermore, training samples of different classes are often overlapped due to drawbacks of feature extraction methods, therefore, it is necessary to fine-tune the parameters in a globally supervised way to increase the discriminative capacities among different models. The related discriminant approaches include: Maximum Mutual Information (MMI)[6], the Minimum Discrimination Information (MDI)[7],the Generalized Probabilistic Descent(GPD)[8,9],the Probabilistic Decision-Based Neural Network (DBNN) [10]. In this paper we present a discriminative algorithm based on the Maximum Relative Entropy (MRE) for GMM.

The organization of this paper is as follows. To reduce the computation load in Gaussian probability density, the definition of quasi-Gaussian is introduced in Section

2. Section 3 addresses the supervised learning algorithm via using the maximum relative entropy as objective function. The performance evaluations of different supervised learning algorithms such as MMI, GPD, MRE and MLE are presented in Section 5. Section 6 contains the summary and discussion.

2 Quasi-Gaussian Density

The L-order quasi-Gaussian rational density in the multiple dimensions (L>0) is defined as

$$N_L(x,\mu,\Sigma)=c_L\frac{1}{\sqrt{|\Sigma|}}\frac{1}{1+d_m(x)+\frac{1}{2}d_h^2(x)+\cdots\frac{1}{L!}d_h^L(x)} \tag{1}$$

Where c_L is a constant subject to the equation $\int_{R^p} f_L(x)dx=1$,only if 2L>p, such kind of constant c_L exists, otherwise, Eq.1 is called as a pseudo-density and $d_h(x)=\frac{d^2(x,\mu,\Sigma)}{2}$, $d^2(x,\mu,\Sigma)=(x-\mu)'\Sigma^{-1}(x-\mu)$. $\Sigma=(\sigma_{ij})_{p\times p}$ is the covariance matrix, x is a vector with p dimension.

3 Global supervised learning FGMM

The EM algorithm [1] for GMM is an unsupervised learning algorithm. The parameters in the model are estimated with the training samples in one class via the maximum likelihood estimation method taking no account of the sample distributions of other classes. When the training samples are inadequate the discriminative capacity of the model is not ideal. To solve the problem, the following will discuss the supervised learning algorithm to fine tune the parameters estimated by the unsupervised algorithm.

Let $\Omega_1,\Omega_2,\ldots,\Omega_I$ be I different classes, the training sample set of i' th class denotes as $S(i)$ The class conditional density of sample x in the class Ω_i denotes

$$p(x|\Omega_i)=\sum_{m=1}^{M}\pi_{im}p_{im}(x) \tag{2}$$

Where, π_{im} is the mixing proportion subject to constraints $\pi_{im}\geq 0$, $\sum_{m=1}^{M}\pi_{im}=1$, $p_{im}(x)$ is a L-order quasi-Gaussian mixture density denoted as where

$$p_{im}(x)=N_L(x,\mu_{im},\Sigma_{im})$$

The relative entropy [11] of two class probability densities can be used as a measure to distinguish two classes. The larger the relative entropy is, the more discriminative the two classes are. Hence, the relative entropy can be used as the objective function to fine-tune the model parameters by maximizing the relative entropy. Other forms of discriminative functions can be found in [6,7].

The relative entropy of th' i class relative to other classes (Also named as information divergence or Kullback-Leibler distance) is defined as [11]

$$H_i = \sum_{x \in S(i)} p(\Omega_i | x) \log \frac{p(\Omega_i | x)}{p(\overline{\Omega}_i | x)} \tag{3}$$

Where

$$p(x) = p(x | \overline{\Omega}_i) p(\overline{\Omega}_i) + p(x | \Omega_i) p(\Omega_i) \tag{4}$$

$$p(\Omega_i |, x) = p(x | \Omega_i) p(\Omega_i) / p(x) \tag{5}$$

To get the formulae for estimating parameters in the FGMM, we summarize the following two lemmas.

Lemma 1. Let $p(x) \sim N_L(x, \mu, \Sigma)$, we have

$$\nabla_\mu p(x) = p(x) v(x) \Sigma^{-1} (x - \mu)$$

$$v(x) = \frac{\left(1 + d_h(x) + \frac{1}{2} d_h^2(x) + \cdots \frac{1}{(L-1)!} d_h^{L-1}(x)\right)}{\left(1 + d_h(x) + \frac{1}{2} d_h^2(x) + \cdots \frac{1}{L!} d_h^L(x)\right)} \tag{6}$$

Lemma 2. Let $p(x) \sim N_L(x, \mu, \Sigma)$, we have

$$\nabla_\Sigma p(x) = \frac{1}{2} p(x) v(x) \Sigma^{-1} (x - \mu)(x - \mu)' \Sigma^{-1} - \frac{1}{2} p(x) \Sigma^{-1}$$

To tune the model parameters by maximizing the total relative entropy, it needs computing the gradient information about parameters of the relative entropy, so the following theorem is obtained.

Theorem 1. For the FGMM defined by the Eq.6, the parameters $\pi_{im}, \mu_{im}, \Sigma_{im}$ satisfy Eq.7, Eq.8, and Eq.9 respectively.

$$\overline{\pi}_{im} = \sum_{x \in S(i)} \pi_{im} q_{im}(x) / \sum_{x \in S(i)} \sum_{j=1}^{M} \pi_{ij} q_{ij}(x) \tag{7}$$

$$\overline{\mu}_{im} = \sum_{x \in S(i)} \pi_{im} q_{im}(x) v_{im}(x) x / \sum_{x \in S(i)} \pi_{im} q_{im}(x) v_{im}(x) \tag{8}$$

$$\overline{\Sigma}_{im} = \sum_{x \in S(i)} \pi_{im} q_{im}(x) v_{im}(x)(x - \mu_{im})(x - \mu_{im})' / \sum_{x \in S(i)} \pi_{im} q_{im}(x) \tag{9}$$

Where

$$q_{im}(x) = \frac{p_{im}(x) p(\Omega_i)}{p(x)} \left(\frac{p(x | \overline{\Omega}_i) p(\overline{\Omega}_i)}{p(x)} \log \frac{p(x | \Omega_i) p(\Omega_i)}{p(x | \overline{\Omega}_i) p(\overline{\Omega}_i)} + 1 \right) \tag{10}$$

Note that values of $v_{im}(x)$ with respect to parameters μ_{im}, Σ_{im} are caculated according to the Eq.6. According to theorem 1, we obtain the following training procedure similar with the EM algorithm:

For locally unsupervised learning

(1) Set $p(\Omega_i)=1$ and $p(\overline{\Omega}_i)=0$.

(2) Classify training samples $s(i)$ into M classes via LBG algorithm.The initial values of μ_{im} are chosen as the codeword derived by LBG algorithm. The initial values of Σ_{im} are estimated via statistics. The initial values of π_{im} are set to $1/M$.

(3) Estimate new values of $\pi_{im}, \mu_{im}, \Sigma_{im}$ via Eq.7-9,and repeat the iteration procedure until the satisfactory accuracy is reached.

For globally supervised learning, the initial values $\pi_{im}, \mu_{im}, \Sigma_{im}$ are chosen as the values estimated by locally unsupervised learning algorithm mentioned above, but the priori probability $p(\Omega_i)$ of each classΩ_i are set to equal, the step 3 mentioned above is used for fine-tuning the parameters.

Finally, we discuss the convergence problem of the above algorithm, when training data are sufficient and $q_{im}(x)$ are positive, we can prove the iteration is convergent.

4 Experiment

To evaluate different methods for estimating parameters in the GMM, two different normal probability models are considered, one is the $N(0,1)$,the other is $N(3.6,1)$. 1000 training samples for each of the two models are generated by the *Matlab* tool. Four different methods for estimating parameters in GMM, the Maximum Mutual Information (MMI), the Maximum Relative Entropy (MRE), the Generalized Probabilistic Descent (GPD) and the Maximum Likelihood Estimation (MLE) are compared.

The estimated mean value, standard deviations and the thresholds with equal error are listed in the Table 1. The true threshold of the two models is *1.8*. From the table we know that the threshold estimated by the GPD is the nearest to *1.8*, hence GPD is the best alternative method, MRE takes the second place and so on. It needs thousands of iteration steps to estimate the parameters by MMI and GPD as shown in Fig.3, however, only tens of iteration steps are needed to estimate the parameters by MRE as shown in Fig.2. Therefore, the computation time of MRE is much less than that of MMI and GPD. Fig.1 demonstrates the comparison between the probability densities estimated by MLE and MRE. The drawing with '+' symbol represents the density estimated by MRE. From Fig.1, we know that MRE based supervised learning algorithm makes the model standard deviation become smaller and inter-class distance become larger.

When 1-order quasi-Gaussian mixture density is used in FGMM, the estimated mean values and standard deviations are listed in the Table 2. From the table we know that the threshold estimated by MRE is the nearest to 1.8,hence MRE is a better alternative method in training speed and accuracy. We found that the MMI method is not convergent, so no estimated parameters are listed in the Table.2.

Table 1. The estimated parameters by different methods

Methods	Model one	Model two	Threshold
MMI	$\hat{\mu}_1$ =-0.1397 $\hat{\sigma}_1$ = 0.9765	$\hat{\mu}_2$ =3.4691 $\hat{\sigma}_2$ = 0.8798	1.7587
MRE	$\hat{\mu}_1$ =-0.0887 $\hat{\sigma}_1$ = 0.8874	$\hat{\mu}_2$ =3.6139 $\hat{\sigma}_2$ = 0.8761	1.7745
GPD	$\hat{\mu}_1$ =-0.1841 $\hat{\sigma}_1$ = 0.5100	$\hat{\mu}_2$ =3.7350 $\hat{\sigma}_2$ = 0.5038	1.7766
MLE	$\hat{\mu}_1$ =-0.0431 $\hat{\sigma}_1$ = 0.9430	$\hat{\mu}_2$ =3.5569 $\hat{\sigma}_2$ = 0.9430	1.7569

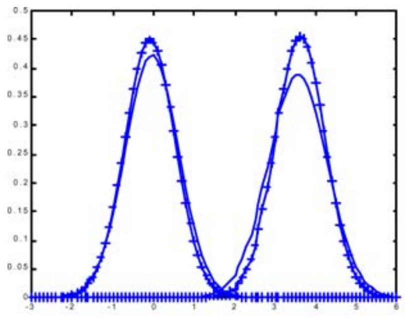


Fig. 1. Comparison between p.d.f

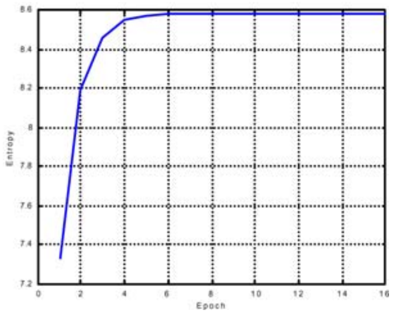


Fig. 2.The convergence of MRE

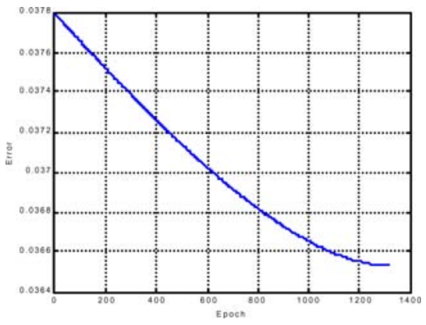


Fig. 3. The convergence of GPD

Table.2. The estimated parameters

Methods	Model one	Model two	Threshold
MRE	$\hat{\mu}_1$ =-0.0907 $\hat{\sigma}_1$ = 0.7587	$\hat{\mu}_2$ =3.6420 $\hat{\sigma}_2$ = 0.7551	1.7801
GPD	$\hat{\mu}_1$ =-0.0564 $\hat{\sigma}_1$ = 0.9605	$\hat{\mu}_2$ =3.6286 $\hat{\sigma}_2$ = 0.5105	2.3497
MLE	$\hat{\mu}_1$ =-0.1012 $\hat{\sigma}_1$ = 0.7582	$\hat{\mu}_2$ =3.6520 $\hat{\sigma}_2$ = 0.7549	1.7795

5 Discussion

In this paper we have described the globally supervised algorithm for Gaussian mixture models based on the maximum relative entropy (MRE). To speed up the computation of the parameters in the model, we introduce the concept of quasi-Gaussian density and apply it to the GMM. Experimental results have shown that FGMM is powerful in accuracy and speed compared with those of MMI, GDP and EM.

References

[1]A.P.Dempster,N.M.Laird,and D.B.Rubin. "Maximum likelihood from Incomplete Data via the EM Algorithm" J.of Royal Statistical Soc.,Ser.B,Vol.39,No.1,1977, pp.1-38

[2]A.L.Yuolle P.Stolorz and J.Utans,"Statisitical physics mixtures of distributions,and the EM algorithm",Neural Computations,Vol.6,1994,pp.334-340

[3]R.J.Hanthanway,"Another interpretation f the EM algorithm for mixture distributions",Statist. Prob.Lrtt.,Vol.4,1986,pp.53-56

[4]C.E.Priebe. "Adaptive mixtures". Journal of the American Statistical Association, 89(427),1994

[5]D.M.Titternington."Recursive parameter estimation using incomplete data," J.R. Statist.Soc.B,46:257-267,1984

[6]L.R.Bahl, P.F.Brown,P.V.de Souza,and R.L.Mercer,"Maximum mutual information estimation of hidden Markov model parameters for speech recognition,",in Proc.ICASSP' 86(Tokyo ,Japan),pp.49-52,1986

[7]Y.Ephrain,A.Dembo,and L.R.Rabiner,"A minimum discrimination information approach for hidden Markov modeling",in Proc.ICASSP' 87(Dallas,TX),1987.

[8]B.H.Juang and S.Katagiri,"Discriminative learning for minimum error classification",IEEE Trans. Signal processing,Vol.40,no.2,pp.3043-3054,1992.

[9]S.Katagiri,C.H.Lee and B.H.Juang ,"Discriminative multilayer feedforward networks," Proc.1991 IEEE Workshop Neural Networks for Signal Processing, Princeton,NJ,1991,pp.11-20.

[10]N.B.Karayiannis and G.W.Mi." Growing radial basis neural networks:Merging supervised and unsupervised learning with networks growth technique" IEEE Trans. On Neural Networks,Vol.8,No.6,1997,pp.1492-1506

[11]T.Cover,J.Thomas. Elements of Information Theory. John Wiley &Sons, Inc., NewYork. 1991

Optimizing Classifiers by Genetic Algorithm*

Wenyun Ji, Liang Zhang, and Wen Jin

Computer Science Department
Fudan University
220 Handan Road, Shanghai 200433, P.R. China
{wyji,zhangl,wjin}@fudan.edu.cn

Abstract. The paper focuses on methods of optimizing a single classifier and combining multiple classifiers by genetic algorithms (GAs). The method uses both the strategies of stacking and GAs to enhance the predictive precision of classifiers. experimetnal results show that it performs well on the task of optimization. Comparing with the single algorithm, it enhances the precision. In task of combining optimization, it can obtain more understandable result than constituent learners.

1 Introduction

Data Classification has been studied substantially in statistics, machine learning, neural networks, expert systems, and now becomes an important theme in data mining [2]. There have been many approaches on data classification, including machine learning approaches, statistics approaches, neural network approaches. Focusing on classification, we can choose an appropriate algorithm among them. When we obtain the information of a classifier after learning the training data with one algorithm, we can use GAs to optimize it and enhance the predictive precision of the classifier. Another case is that we use several algorithms. Applying an approach to a set of training data defines a unique learned model. A diverse collection of approaches may produce a set of learned models that vary considerably in generalization performance for a given prediction task. Some approaches perform well on certain prediction tasks and do poorly in others. Different approaches have different "perspectives" for the prediction task. Their errors are likely diverse when the various approaches have been used. We can also combine these approaches by using GAs to improve prediction. Hence, Optimizing one algorithm(referred as optimization) or optimizing several algorithms (referred as combining optimization) by GAs have the same algorithm structure.

There have been several combining approaches which obtain a more accurate prediction than that obtained from any single source alone. The naive approach is referred earlier as the plurality vote (PV). It can be applied to the situations that the errors are uncorrelated. SCANN [5] can be used in the situation that the errors are correlated. It is better than previous methods on being insensitive to

* This project is funded in part by 973 Projects Foundation of China, and the National Doctoral Subject Foundation of China

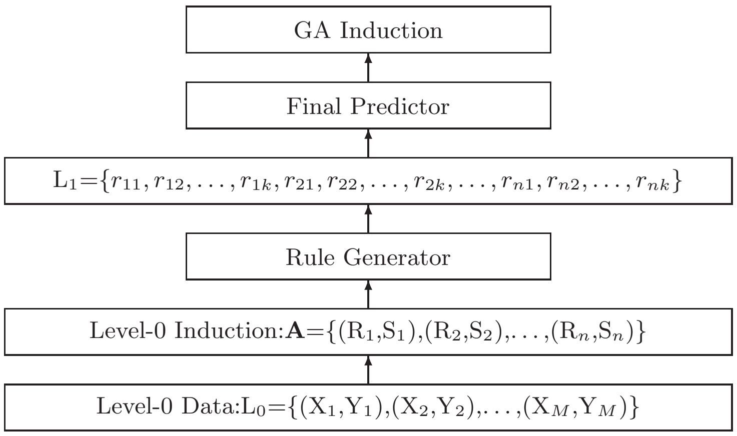

GA Induction
Final Predictor
$L_1=\{r_{11},r_{12},\ldots,r_{1k},r_{21},r_{22},\ldots,r_{2k},\ldots,r_{n1},r_{n2},\ldots,r_{nk}\}$
Rule Generator
Level-0 Induction:$\mathbf{A}=\{(R_1,S_1),(R_2,S_2),\ldots,(R_n,S_n)\}$
Level-0 Data:$L_0=\{(X_1,Y_1),(X_2,Y_2),\ldots,(X_M,Y_M)\}$

Fig. 1. Flowchart of Combining Classifiers

poor learned models and being highly accurate. Stacking [8] has been defined as a method for discovering and correcting for the systematic biases of the learning algorithms used.

GAs [3] are optimization techniques that work well on a broad range of problems. Because GAs do not require specific knowledge about the problem except the fitness function, it becomes very easy to hybridize them with other techniques. This paper will use GAs as the combining generator of various methods.

2 Our Work

2.1 Combining Method

The paper uses the framework of stacking [8], and uses GAs as Level-1 induction which is as the synthesized method of various models. The flowchart is as Figure 1.

The rudimental steps of combining method is as following:

1. A set of learning data, $\Gamma = \{(x_m, y_m), m = 1, \ldots, M\}$, where M is the number of examples.
2. A collection of learning algorithms, A_1 , A_2,..., A_N, where N stands for the number of learning algorithms. Let **A** be defined by a space of representations **R**, and a method **S** for searching through representation, i.e. **A**=(**R**, **S**).
3. Combine the results of above algorithms with GAs, and obtain a synthesized interpretation.

In this framework, a GA-based classifier is in the form of rule sets. It is coded in the light of Goldberg's method[3]. Each individual represents a solution to the problem. It is coded as a chromosome which consists of a sequence of gene segments. Each gene segment corresponds to a rule here. Thus each individual have a rule set which can express the total solution.

2.2 Algorithm Description

```
Input:
  R={RuleSet1, RuleSet2,...,RuleSetN};  //All GA-based rule sets.
  MatchMatrix[i][j][k].  //Corresponding relation among rule sets
Output:
  RuleSetFinal. //Set of rules after optimization
Procedure GaCombining()
  oldPop=Initialize_population(R);
  gen=0;
  Repeat
    gen=gen+1; // For the next generation
    While(ChildNumber<PopSize) DO
    Begin
      mate1=Select(oldPop);  // Use roulette wheels selection
      mate2=Select(oldPop);
      If(rand()>PXOVER)  // Is the crossover likely to be happen?
        Xover(parent1,parent2,child1,child2);  // Use MatchMatrix
      If(rand()>PMUTATION) // Is the mutation likely to be happen?
        Mutation(parent1,parent2,child1,child2);
      TestData(child1);  // Calculate the matchness
      TestData(child2);
    End
    NormalizedFitness(newPop)   // Calculate fitness
    For i:=0 To PopSize Do
      OldPop[i]=newPop[i];
  Until gen>MaxGEN OR
        the improvement of best individual<Threshold
  RuleSetFinal=inteprete(best_individual(OldPop));
```

3 Experiments and Results

3.1 Optimization for Single Algorithm

Since rules of decision tree algorithms (such as ID3, C4.5) are considered best understandable among the available choices, we choose them as algorithms to be optimized in the following examples.

Iris Dataset [1] We use C5.0[2] to generate rule sets. A sample is in Table 1.

Table 1. C50-based rule set of Iris

Rule 1: (cover 35)	$Petal\text{-}length \leq 1.9 \rightarrow classIris\text{-}setosa$
Rule 2: (cover 32)	$Petal\text{-}length > 1.9 \wedge Petal\text{-}length \leq 5 \wedge Petal\text{-}width \leq 1.6$ $\rightarrow classIris\text{-}versicolor$
Rule 3: (cover 29)	$Petal\text{-}width > 1.6 \rightarrow classIris\text{-}virginica$
Rule 4: (cover 28)	$Petal\text{-}length > 5 \rightarrow classIris\text{-}virginica$
	Default class: *Iris-setosa*

Table 2. Accuracy of C50 rules and GA-optimizing rules on iris

	C5.0	rules	GA-optimizing	rules
	Training	Testing	Training	Testing
Setosa	25/25	25/25	25/25	25/25
Versicolor	24/25	23/25	24/25	23/25
Virginica	25/25	24/25	25/25	25/25
Overall	74/75	72/75	74/75	73/75
	98.67%	96%	98.67%	97.33%

After optimization by our method, the rule set remains the same form, but several parameters have been changed. Notice that, in rule 2, the right boundary value of *petal-length* changes to 4.91, and in rule 4 , the left boundary value of *petal-length* changes to 4.70.

Table 2 gives the accuracy comparison of C5.0 rules and GA-optimized rules on the training and testing data. From it, we can find out that the result after GA-optimization is better than that of previous single algorithm. In the test data, two examples of iris-versicolor are misclassified as iris-virginia. The parameters of two examples are similar to that of iris-virginia, so only through the combination of the parameters can classify them correctly[6]. But understandability is poor when you classify examples by inequality, and they are not easily translated into SQL queries.

Synthetic Dataset . We use classification function-2 which is defined in [1].

As reported in [1], ID3 generated a relatively large number of strings for function-2 when the decision tree is built. When C4.5 rules is used, it generated 18 rules. Among the 18 rules, 8 rules define the conditions for Group A. The rules that define tuples to be a member of Group A can be found in [4].

[1] please refer to: http://www.ics.uci.edu

[2] please refer to: http://www.cse.unsw.edu.au/ quninlan

According to the frequency histogram, we will choose attributes of age and salary to perform GAs. For the rule 6, we can obtain the new rule which is

$$(25100 < salary < 74800) \wedge (age > 60.8) \tag{1}$$

after 70 generations [3]. For the rule 20, we can obtain the new rule

$$(74800 < salary \leq 124000) \wedge (39 < age \leq 58.7), \tag{2}$$

for the rule 10, we can easily obtain the result of

$$(50100 < salary < 100200) \wedge (age < 39.8) \tag{3}$$

after a few generations. Other rules have the same result as above after optimization, and the accuracy is also high, there is one mismatch in 1000 tuples. The result is similar to the original function. GAs are robust and can search the global schema. When the boundary value deviates its right value, GAs can obtain it by evolution.

We choose neural network (NN, a connectionist approach) and C5.0(a symbolic approach) as two different algorithms. The dataset is iris.

According to [7], the NN-based rules set of iris is given as Table 3. The C5.0-based rule set is listed in Table 1.

Table 3. NN-based rule set of Iris

Rule 1:	*Petal-length* $\leq 1.9 \rightarrow$ *Iris-setosa*
Rule 2:	*Petal-length* $\leq 4.9 \wedge$ *Petal-width* $\leq 1.6 \rightarrow$ *Iris-versicolor*
	Default rule: *Iris-virginica*

Table 4. C50-based rule set after optimization

Rule 1: (cover 35)	*Petal-length* $\leq 1.9 \rightarrow$ *classIris-setosa*
Rule 2: (cover 32)	*Petal-length* $> 1.9 \wedge$ *Petal-length* $\leq 4.95 \wedge$ *Petal-width* ≤ 1.6 $\rightarrow$ *classIris-versicolor*
Rule 3: (cover 29)	*Petal-width* $> 1.6 \rightarrow$ *classIris-virginica*
Rule 4: (cover 28)	*Petal-length* $> 4.95 \rightarrow$ *classIris-virginica*
	Default class: $Iris - setosa$

After optimization, we can obtain two set of rules. One is NN-based, it remains unchanged. The other is C50-based that is showed in Table 4, which changes its value at boundary. Now they have the same precision. Before optimization, NN rules have higher accuracy, but less understandable. From the

[3] in rule 6, the right boundary of salary deviates its right position very much

NN rule, we don't know the property of iris-virginica, only know it's the default type. In contrast, the C50-based one is understandable, but has less accuracy. After optimization, we can obtain the rule set of high accuracy and being understandable.

4 Summary and Conclusions

GAs are optimizing techniques that work well on the single and multiple classifiers. When it is used on a single algorithm, it acts as optimization, and when it is used on multiple algorithms, it acts as combining method. GAs can improve the predictive precision through varying the value of boundary. A GA searches the most promising areas of the space irrespective of local optima. When evolution ends, the value of boundary reach its best value to reflect the property of classifier. GAs are robust and easy to apply. Although values at boundary deviate its right position very much at first, GAs can also search for it in global areas. As a combining method, GAs can generate rule sets. The result is more understandable, hence it can easily be translated into SQL queries for efficient interfacing with relational databases. Other combining algorithms such as PV, SCANN result in a loss of interpretability. Limitation of GAs are that the parameters to be evolved simultaneously cannot be numerous. When parameters become numerous, A GA will not converge into its best solution. The method to overcome it is to evolve many parameters gradually. Another limitation is its training time. We can reduce its time of scanning test data by the parameters of threshold which can interrupt the test procedure when the parameters of individual is out of specified interval and the individual will be eliminated. The ideal algorithms to be optimized at level-0 should be fast enough (such as C4.5) but their accuracy could be comparatively lower.

References

1. Agrawal, R. , Imielinski, T., Swami, A.: Database Mining: A Performance Perspective. IEEE Trans. on Knowledge and Data Engineering, **5** (1993) 914–925 458
2. Fayyad,U. M., Piatetsky-Shapiro,G., Smyth,P., Uthurusamy,R.: Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996) 455
3. Goldberg,D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York (1989) 456, 457
4. Lu,H., Setiono,R., Liu,H.: NeuroRule: A Connectionist Approach to Data Mining. Proceedings of the 21st VLDB Conference, Zurich, Switzerland (1995) 478–489 458
5. Merz,C. J.: Using Correspondence Analysis to Combine Classifiers. Machine Learning **36** (1999) 33–58 455
6. Setiono,R.: Techniques for Extracting Rules From Artificial Neural Networks Plenary Lecture presented at the 5th International Conference on Soft Computing and Information Systems, Iizuka, Japan, (1998) 458
7. Setiono,R., Liu,H.: Understanding Neural Networks via Rule Extraction. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (1995) 480–485, Montreal, Canada 459
8. Wolpert,D. H.: Stacked Generalization. Neural Networks, **5** (1992) 241–259 456

Author Index