Springer Theses
Recognizing Outstanding Ph.D. Research

Massimiliano Izzo

Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management

Springer

Springer Theses

Recognizing Outstanding Ph.D. Research

Aims and Scope

The series "Springer Theses" brings together a selection of the very best Ph.D. theses from around the world and across the physical sciences. Nominated and endorsed by two recognized specialists, each published volume has been selected for its scientific excellence and the high impact of its contents for the pertinent field of research. For greater accessibility to non-specialists, the published versions include an extended introduction, as well as a foreword by the student's supervisor explaining the special relevance of the work for the field. As a whole, the series will provide a valuable resource both for newcomers to the research fields described, and for other scientists seeking detailed background information on special questions. Finally, it provides an accredited documentation of the valuable contributions made by today's younger generation of scientists.

Theses are accepted into the series by invited nomination only and must fulfill all of the following criteria

- They must be written in good English.
- The topic should fall within the confines of Chemistry, Physics, Earth Sciences, Engineering and related interdisciplinary fields such as Materials, Nanoscience, Chemical Engineering, Complex Systems and Biophysics.
- The work reported in the thesis must represent a significant scientific advance.
- If the thesis includes previously published material, permission to reproduce this must be gained from the respective copyright holder.
- They must have been examined and passed during the 12 months prior to nomination.
- Each thesis should include a foreword by the supervisor outlining the significance of its content.
- The theses should have a clearly defined structure including an introduction accessible to scientists not expert in that particular field.

More information about this series at http://www.springer.com/series/8790

Massimiliano Izzo

Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management

Doctoral Thesis accepted by
the University of Genoa, Italy

Springer

Author
Dr. Massimiliano Izzo
Oxford e-Research Centre
University of Oxford
Oxford
UK

Supervisors
Prof. Marco M. Fato
Department of Informatics, Bioengineering, Robotics and System Engineering
University of Genoa
Genoa
Italy

Dr. Luigi Varesio
Giannina Gaslini Institute
Genoa
Italy

ISSN 2190-5053 ISSN 2190-5061 (electronic)
Springer Theses
ISBN 978-3-319-31240-8 ISBN 978-3-319-31241-5 (eBook)
DOI 10.1007/978-3-319-31241-5

Library of Congress Control Number: 2016934423

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Parts of this thesis have been published in the following articles:

Journals

M. Izzo, F. Mortola, G. Arnulfo, M. M. Fato, and L. Varesio. "A digital repository with an extensible data model for biobanking and genomic analysis management," *BMC Genomics*, 15(Suppl 3):S3, 2014.

International Conferences and Workshops

M. Izzo, G. Arnulfo, M. C. Piastra, V. Tedone, L. Varesio, and M. M. Fato. "XTENS-a JSON-based digital repository for biomedical data management," *Bioinformatics and Biomedical Engineering*, Springer International Publishing, pp. 123–130, 2015

Data is not information,
Information is not knowledge,
Knowledge is not understanding,
Understanding is not wisdom.

(attributed to) Clifford Stoll and Gary Schubert

This doctoral thesis is dedicated to
Vincenzo Tagliasco (1941–2008),
who taught me the real measure
of University grades.

Supervisor's Foreword

It is a great pleasure to introduce Dr. Massimiliano Izzo's thesis work, accepted for publication within Springer Theses and awarded with a prize for outstanding original work. Dr. Izzo joined my research group for Biomedical Instrumentation and Bioimaging after finishing his master's degree in Bioengineering at the University of Genoa (Italy), in March 2006, and after a limited experience as a clinical engineer at the Hospital IRCCS Neuromed Pozzilli (IS) Italy (www.neuromed.it). He started his Ph.D. programme in Bioengineering and Robotics at the University of Genoa with a three-year scholarship of the Hospital IRCCS Giannina Gaslini working in data management and integration for biomedical research, and completed his Ph.D. thesis with an oral defense on 20 April 2015.

The Ph.D. programme in Bioengineering and Robotics is rooted in the multiple ongoing collaborations between Università degli Studi di Genova/University of Genoa (www.unige.it), Fondazione Istituto Italiano di Tecnologia/Italian Institute of Technology Foundation (www.iit.it), and the Università degli Studi di Cagliari/University of Cagliari (www.unica.it). Research in Bioengineering and Robotics at the University of Genoa dates back to the 1980s and originates from the convergence of multidisciplinary activities in the fields of automation, electronics, ICT, and biophysics. Research activity in the field of Bioengineering has a long tradition of scientific excellence at the University of Genoa. Since 1982, the University of Genoa has been participating in the Ph.D. programme of Bioengineering, coordinated by the Polytechnic University of Milan. A local Ph.D. programme in Bioengineering was first established in 1999. A Ph.D. programme in Robotics was established in 1991, and was later merged with the Ph.D. programme in Computer Engineering.

The Hospital IRCCS Gaslini ("Istituto Giannina Gaslini", www.gaslini.org) was established in Genoa, in 1931, by an act of donation and solidarity of Senator Gerolamo Gaslini, who wished to honour his daughter Giannina who passed away in her infancy. Gerolamo Gaslini wanted to ensure that children receive only the best possible type of medical treatment and care based on the most innovative biomedical research. All paediatric and surgical services are practised at the

Institute, which also houses scientific laboratories, as well as affiliated Genoa University Departments, many of which also have their own related Graduate Schools and postgraduate Specialisation Courses. Gaslini Institute has always been a reference point at the national and international level in many disciplines, so much so that it receives thousands of children of 90 nationalities every year, as well as over twenty thousand children from all Italian Regions, which constitute around half of all its patients. Data management and integration has become a major issue in contemporary biomedical research.

Modern genomic profiling platforms, such as high-throughput gene sequencing platforms, now produce outputs of several hundreds of gigabases. The gathered genomic information must be integrated with all available data about patient clinical history and lifestyle. Such an integrated approach will be of paramount importance as healthcare paradigms move towards personalised medicine. Extensive metadata are required to improve the collection and analysis of such information. For these same reasons, life science research is evolving into international multidisciplinary collaborations based upon increasing data sharing among scientific laboratories and institutions. Each single lab implements different protocols and performs its analyses by using different instrumentation. Therefore, metadata are inconsistent, poorly defined, ambiguous, and do not use a common vocabulary or ontology. Research collaborations are evolving from local to global scales, the heterogeneity of the collected metadata grows and no single standardisation is possible. Moreover, as detailed in the thesis, recent studies in Neural and Social Sciences encourage the view of metadata as a fluid, dynamical process rather than a fixed product. The main goal of Massimiliano was therefore to build an innovative data repository able to provide adaptive metadata management and configuration tools to maximise information sharing and understanding in multidisciplinary and possibly international collaborations. Massimiliano has developed a novel data management platform that is a valid alternative to more established technological solutions, because it presents a number of advantages. As far as I know, this is the first JSON-based metadata repository designed for Biomedical Sciences. I think it can provide sensitive benefits if adopted for Research Collaborations where flexible and extensive metadata support is required. The platform has been used with success in different scenarios—ranging from Neuroscience to Biobanking and Functional Genomics. As a biobanking management tool, it has been used on a daily basis for more than one year at the Giannina Gaslini Children's Hospital. The adoption of XTENS as a collaborative platform promotes dynamical information sharing and heterogeneous data integration. The overall goal is to use this platform to drive the discovery of new biomarkers or the design of new predictors (i.e., classifiers) to better characterize and study diseases with negative or heterogeneous outcomes, such as rare diseases or developmental tumours.

Massimiliano is distinguished for his solid knowledge of engineering basics and his remarkable skill in problem solving both at the theoretical and practical levels in an efficient manner. He has proved to have a strong motivation to work in the bioengineering research field. His communication skills and ability to work in a team enabled him to achieve important results in his research and in his career.

Genoa
December 2015

Prof. Marco M. Fato

Acknowledgements

I would like to thank the molecular biology team at Gaslini—Federica, Cristina, Daniela, Martina, and Simone—and the engineering fire squad at Biolab, DIBRIS—Andrea, Maria Carla, Keywan, and so on and so forth. Thanks to Gigi (Dr. Luigi Varesio), my head at the Gaslini Institute, for providing feedback, suggestions, and hypoxia during the thesis development. Thanks to Marco (Prof. Marco M. Fato), my supervisor, for enduring support. Special thanks to Pamela who was the first one testing and using the XTENS 1.5 installation at Gaslini and patiently kept notice of all the little dirty bugs popping up! Thanks to Gabriele who proofread the thesis. Many thanks to Giulia and Alessia for many a lunch together at Gaslini. Special thanks to Stefania who read carefully through various drafts and revisions, and offered me precious suggestions. Very special thanks to Valentina who helped me running the performance tests, developing some XTENS 2 components (namely, the graph visualiser), and proofreading the final draft. Thanks to Filippo e Sara, *animus et anima*: your friendship and counsel kept me sane (to some extent) through these three years and counting (one hopes!!). Thanks to Marco M for many a beer, strong friendship through hardship, and tons of rock 'n' roll. Super thanks to Davide for the good times we had sharing the same office (a loo turned into an office, actually!). Thanks to Giovanni for needful quotes and folding bikes: the best transport ever to get to work. Thanks to my parents Claudia and Giancarlo (special mention to my mother who cooked my daily meals!), to my family, and to my godfather Giacomo, who was so kind to come and assist in the Ph.D. Viva Voce. Finally, a special thank you to all the friends, colleagues, and relatives I forgot to mention explicitly. All of you played a role in shaping me and the output of my work.

Genoa
December 2015

Massimiliano Izzo

Contents

Abbreviations

aCGH	Array-based Comparative Genomic Hybridisation
API	Application Programming Interface
BiolMol	Laboratory of Molecular Biology
CARMEN	Code Analysis Repository and Modelling for e-Neuroscience
COINS	Collaborative Informatics and Neuroimaging Suite
CRUD	Create—Retrieve—Update—Delete
DCMES	Dublin Core Metadata Element Set
DIBRIS	Department of Informatics, Bioengineering, Robotics, and Systems Engineering
EAV	Entity—Attribute—Value
GIN	Generalised Inverted Index
i2b2	Informatics for Integrating Biology and the Bedside
IGG	Giannina Gaslini Institute
iRODS	integrated Rule-Oriented Data System
JSON	JavaScript Object Notation
Jsonb	PostgreSQL JSON Binary Format
MIABIS	Minimum Information About BIobanking data Sharing
MIAME	Minimum Information About a Microarray Experiment
MRI	Magnetic Resonance Imaging
NGS	Next Generation Sequencing
ODM	Object-Document Mapper
openBIS	Open Biology Information System
ORM	Object-Relational Mapper
RDBMS	Relational Database Management System
RDF	Resource Description Framework
REST	REpresentational State Transfer
SEEG	Stereoelectroencephalography
SIMBioMS	System for Informative Management in BioMedical Studies
SNP	Single Nucleotide Polymorphism
WGS	Whole Genome Sequencing
XCEDE	XML-based Clinical and Experimental Data Exchange

XML	eXtensible Markup Language
XNAT	eXtensible Neuroimaging Archive Toolkit
XSD	XML Schema Definition
XTENS	eXTENsible platform for biomedical Science

Chapter 1
Background

In the last two decades, huge investments in basic science have produced a tremendous progress in the methods and technologies available to clinical medicine. Imaging platforms now detail the structure and the function of the human body and have been a driving force in our growing understanding of various organs and apparatuses, most notably the brain. High-throughput genomic[1] technologies have allowed researchers and clinicians to spot thousands of DNA mutations that play a role in the onset and evolution of a variety of human diseases. This shift towards the individual genomic profile has brought up the concept of personalised medicine, whose goal is to better predict the patient's responses to therapies on the basis of the genetic profile. The current challenge is how to deliver the benefits of these discoveries and advancements to patients. As stated by Hamburg in her guideline paper "the success of personalised medicine depends on having accurate diagnostic tests that identify patients who can benefit from targeted therapies" [2]. Careful attention is now given to rare and neglected diseases, that are used as models to study and identify molecularly different subtypes of more common disorders. To achieve this, a particular focus is on the development tissue biobanks, and in their workflow. Traditionally, biobanks were centres dedicated to the collection and storage of biological specimens. Nowadays, they are evolving towards institutions that gather a whole spectrum of mostly digital information, including social, clinical, and pathological records together with genomic profiles. As a consequence, modern biobanking is shifting its focus from sample-driven to data-driven strategies.

[1] The words genetics and genomics are used quite interchangeably in common media. However they bear a different meaning. Genetics is mostly concerned with how inherited traits are transmitted from one generation to the next through the genes and how mutations can produce new traits. Genomics addresses all the genetic material of an organism and its function in order to determine its combined influence on growth and development [1]. In nearly all my thesis I will focus on genomics.

M. Izzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5_1

1.1 Big Data

The cost of an entire genome sequencing has dropped from millions to thousands of dollars (or euros), given the improvements in sequencing technologies and computational power [3]. For instance, one of the world's largest biology and data repositories, the European Bioinformatics Institute (EMBL-EBI) of Hixton, UK, currently stores 40 petabytes (PB) of data and backups about genes, proteins and small molecules. Genomic data alone account for about 5 PB, a figure that more than doubles every year [4]. To put it in perspective, this is about one tenth the volume of data stored at CERN, the leading European's particle physics organisation. By any account, biology has joined the club of "Big Data". Big Data is an umbrella term—or a buzzword, depending on your opinion on the subject—that encompass a new generation of (mostly) software technologies and architectures developed to extract knowledge and, ultimately, value from large volumes of heterogeneous data by allowing high velocity capture, discovery and processing. Big data require a paradigm shift in both storage requirements and data analysis. In my thesis I will examine the state of the art on data management solution and propose a possible approach to store, manage and search large biomedical datasets of heterogeneous nature. When dealing with big data, the data are most frequently unstructured or semi-structured, and do not fit well in a traditional relational database (i.e. with a fixed schema) design.[2] Novel data store approaches, such as those based on NoSQL solutions [6], should be adopted alongside—and most often not in substitution of—relational databases. See Fig. 1.1 for further details on data structuredness.

As a consequence of the data explosion in biomedicine, the emphasis has moved from population-based health care to personalised medicine with the goal of developing targeted diagnostics and therapies based on patients' clinical history, ancestry and genomic profile. A major challenge to achieve more reliable patients' outcome is the integration of clinical data, omics data, administrative data and financial information on a single management system. These ongoing changes towards data-centric health care will affect—in a positive way, one hopes—the physicians, as computer-aided medicine, web-based solutions and big data analytics will support and guide the individuation of the optimal therapy.

[2] As stated by Abiteboul, "semi-structured data is data that is neither raw data, nor very strictly typed as in conventional database systems" [5]. Among the main aspects of semi-structured data there are an irregular structure (due to the heterogeneity of the data components), an implicit structure (the explicit structure must be parsed or extracted), and a large and rapidly evolving data schema. In the context of semi-structured data the distinction between schema and data may even blur, as some classification information (e.g. the sex of an individual) may be stored either as data or as a type. Fully unstructured data do not have a pre-defined data model or schema and are typically text-heavy though they also may contain numeric or date elements.

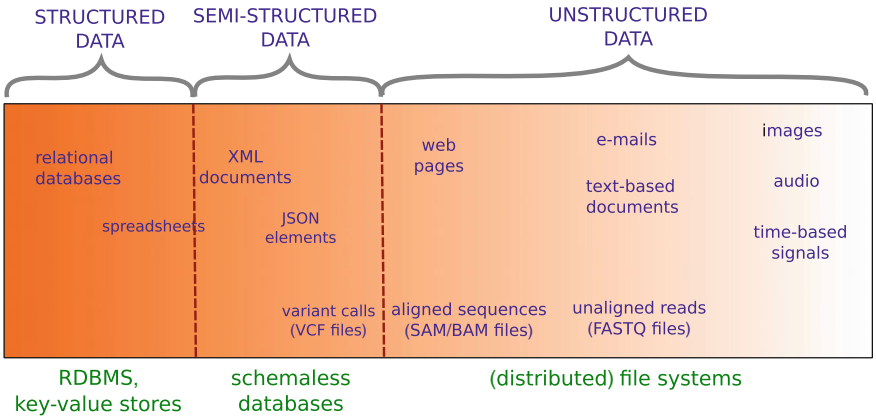

Fig. 1.1 Taxonomy of data based on their internal structure. Structured data are associated to a fixed data schema and usually presented in tabular form in database tables or spreadsheets. Semi-structured data can be stored with flexible schema notations like XML or, preferably, JSON (see Sects. 2.6 and 3.3). Unstructured data is usually text-based (email, word-processors documents, ……) or in binary form (images, audio, other signals). The figure highlights that the level of structure in NGS data formats increases as we move from raw data (e.g. FASTQ), to indexed alignments (SAM/BAM), and to variant calls (VCF). The scale is qualitative and by no means resolutive, but it is meant to provide a general overview

1.2 Omics

Deriving from the Sanskrit root "*om*" "completedness and fullness", the term *omics* refers to a systems-oriented approach to biological science that aims to characterise individual and population differences in living organisms. The postfix was first used to create the word *genome*, to describe the whole genetic markup of an individual, that in turn gave origin to the discipline called genomics. Similarly, the *exome* refers to the entire part of DNA that will be transcribed to RNA, while *transcriptome* refers to its RNA counterpart. The entire set of protein of an organism is called *proteome*, and its domain of study is now commonly name *proteomics*. Finally, the complete set of small-molecule metabolites (such as signalling molecules, hormones and molecular intermediates) of an organism constitute the so-called *metabolome*, whose branch of study is now called *metabolomics*. Genomics, proteomics, metabolomics, together with other disciplines focused on systems-wide exploration of biological organisms all belong to the omics field.

Among the analysis available to study the genome at DNA level there is the array-based Comparative Genomic Hybridisation (aCGH) technique, a molecular cytogenetic method to identify Copy Number Variations—loss (deletions) or gains/amplifications of DNA segments. It allows to study the entire genome with higher (100–1000 times) resolution compared to other karyotype investigations, and permits to identify the chromosomes regions and the genes involved in the rearrangements, helping the clinician to highlight the link between certain genetic aberrations and

the disease. Another karyotyping technique is the Single Nucleotide Polymorphism (SNP) array, which are used to detect SNP, a variation at a single site in DNA. SNPs are the most frequent type of variation in the genome. There are roughly 10 million SNPs per individual human genome (1 SNP in every 300 bases). SNP-arrays and aCGH and are also used to locate abnormalities in cancer. At the latest release, the comprehensive NCBI database dbSNP listed over 112 SNPs in humans [7]. There is a huge number of SNP-array chips available on the market, each with its own specifications. Currently the number of SNPs probed by a chip range from 600 thousand (e.g. Axiom Genome-Wide Human EU/ASI from Affimetrix) to 2.5 million (HumanOmni2.5-8 from Illumina) [8]. aCGH and SNP-arrays are also used to identify genomic abnormalities in cancer.

DNA microarray was the first high-throughput genomic technology to investigate gene expression at the level of RNA (i.e. transcriptome). The Human Genome Project has estimated that there are about 30,000 protein-coding genes in the human genome. A single microarray chip allows to measure the expression levels of a large number of genes at once. Currently microarray chip target about 50,000 probes, and usually more than one probe maps to a single gene. Next Generation Sequencing (NGS) technologies have made whole genome, exome, and targeted sequencing available to biology laboratories at an affordable price. NGS platforms adopt different strategies to decode the DNA sequence, but are all characterised by a massive parallelism that speeds up the analysis and allows a deep coverage (tens/hundreds of reads at any genome position). Low parallelism and scarce automation of the workflows were two main issues of Sanger sequencing [9], the method used before the outbreak of NGS. As of 2014, NGS is the fastest growing segment in the genomic field, and it is dominated by Illumina, that accounts for the 74 % of the entire market. The other competitors are Life Technologies (producers of the Ion Torrent sequencer), 454 Roche, and Pacific Bioscience.

Driven by this flood of new technologies and data produced, large scale projects dedicated to investigate omics are already well on the way. One of the most famous is the 1000 Genome Project [10], which aims at building the largest dataset of genomic mutation, integrated with genotype and expression data. Another notable big data effort is the ENCODE project [11, 12], devoted to map and characterise the behaviour of the entire genome. It has already individuated biological functions for about 80 % of the human genome, evidencing that the great majority (∼90 %) of the found mutations are located in regions with no protein-coding genes. Moreover, the ENCODE project found evidence to disproof the erroneous concept of "Junk DNA", pointing out that about the 75 % of the DNA is transcribed in at least one cell type within the organism. ENCODE studies have generated increasing interest on the role and function of the so-called non protein-coding genes and noncoding-RNA. None of these discoveries would have been possible without the huge amount of data brought on the field by NGS platforms.

However, challenging data management and processing issues have stated to rise on the surface. Presently, the DNA sequencing rate of the current technologies

exceeds Moore's Law[3] [3]. As a consequence, only distributed, grid- and cloud-based solutions can solve issues of data storage and computation. In the next years, cloud computing could bolster a new industry whose foundations are set in molecular biology and omics. Another option is the adoption of Graphical Processing Units (GPUs) for executing highly parallel operations. Whenever a task can be split in multiple parallel computations, GPUs will outperform cloud-based solutions.

Whichever will be the underlying technology, there are a number of issues that researchers will face:

1. **storage**: biological and medical data are more heterogeneous that information from any other research domain. No single comprehensive, cheap and secure solution exists to solve the problem of data storage, even though many private companies have started proposing solutions for the life science domain. This issue will most likely affect small labs and institutions, that until now did not have to consider this aspect.
2. **security and privacy**: the adoption of advanced encryption algorithms—like those available to financial institution—is an available option to secure personal information during storage and/or transmission. Distributed storage and computing environments can enforce security policies implementing the Grid Security Infrastructure (GSI), a dedicated specification for secret communication [13]. GSI adopts both symmetric and asymmetric encryption to enable secure and authenticatable communication. From the procedural point of view, proposals of broad consent forms to allow datasets availability for research project are under discussion under various institutions and organisations, including the European Union. In the end, whenever security becomes a major issue (i.e. data containing personal informations), internal/local solutions should be implemented instead of recurring to public clouds.
3. **transfer**: usually it is solved dispatching bulk data through external hard disks. A possible solution might be the adoption of efficient lossless compression algorithms to sensitively reduce the files' size. Other approaches include peer-to-peer file sharing, or distributed storage paradigms—such as the so-called "data grids"—that could take advantage of high-performance, secure and reliable data transfer protocols such as GridFTP [14]. GridFTP is an extension of the standard File Transfer Protocol (FTP) to distributed systems, and relies on GSI to handle authentication and authorisation. In this sense, Globus Online is a Dropbox like technology that implements GridFTP and provides "Software-as-a-Service" (SaaS) solutions for data storage [15].

In my thesis I will assess the issue of heterogeneous data storage for biomedical research. Obviously, this topic cannot be treated ignoring the other two issues: in particular, I will take into account security and privacy regarding the access to a data

[3]Moore's Law reflects the empirical observation that, over the history of computing hardware, the density of transistors on integrated circuits and the computational speed doubles about every two years.

storage platform. While high-performance transmission will not be tackled explicitly, I will expose how some solutions—such as GSI—can be implemented in the framework for data management that I have developed.

1.3 Human Neuroimaging

Neuroscience requires to probe the Central Nervous System (CNS) structure and functions at different scales to achieve a greater understanding of the brain. Usually the CNS is investigated and modelled at molecular, cellular, network, whole system and behavioural level, adopting a wide variety of technologies and approaches. One of the driving forces behind multi-scale structural and functional modelling has been the wide adoption of imaging technologies from Magnetic Resonance Imaging (MRI) [16]. In this sense, neuroscience is facing similar issues to those exposed for molecular biology and omics. Neuroinformatics tools worldwide now produce and collect more data in a few days than were generated over an entire year just a decade ago [17]. Currently, MRI based neuroimaging techniques are sources of information on:

- brain anatomy, with better tissue contrast and fewer artefacts than Computed Tomography (CT). Besides tradition MRI recordings, Diffusion Tensor Imaging (DTI) allows to reconstruct microscopic details about tissue architecture [18]. It is now routinely used to visualise the location orientation and morphology of the brain's fibre (i.e. axonal) tracts.
- brain activity in response to a cognitive stimulus. Functional MRI (fMRI) adopts MRI technology to detect variations in blood flow and estimate through it brain activity. A non-invasive method adopted to stimulate small regions of the brain is the Transcranial Magnetic Stimulation (TMS). Other imaging methods employed to study brain activity are the Positron Emission Tomography (PET) and the Single Photon Emission Computed Tomography (SPECT).
- complex patterns of inter-regional communication. To highlight neural interactions at a systemic level, the techniques mentioned above are further integrated with multi-channel Electroencephalography (EEG) and Magnetoencephalography (MEG) signals.

Multi-modal and multi-scale neuroscience has allowed to reveal the human brain structure and function with increasing level of detail. It has been estimated that, as of 2015, the amount of raw data associated to each published neuroscientifical study will exceed 20 GB, and this is most likely an underestimate [16]. These dataset, if taken individually, do not pose issues for analysis and post-processing, but storage issues may arise, when they are stored in large databases or digital repositories. As I will illustrate in Sect. 2.5, many data management solutions have already been developed with a specific focus on neuroscience and neuroimaging. However none of these are yet taking into account the next big step forward: the integration of neuroscience/neuroimaging datasets with omics information. This new (r)evolution is already under way, as there is an increasing interest in mapping the influence of

genome mutations and gene expression variations on the CNS. The size of the raw datasets available to neuroscientists and neuroinformatiscians are likely to grow of various orders of magnitude (up to the level of terabytes and petabytes). Besides the sheer size, heterogeneity will be the main issue to achieve data integration and ultimately extract knowledge that provides new insights on brain diseases' outcome and evolution. The bond between neuroscience and genomics is growing tighter and many issues already mentioned in the previous section will affect also the brain research. It must also be considered that the scientific efforts in the field become increasingly multi-disciplinary (as we have seen) and collaborative. Research collaborations involve groups with different expertise and technical background, located in different geographical areas, possibly in different countries. Distributed data management solutions for heterogeneous information now become a major requirement for biomedical research.

References

1. World Health Organization et al. Genomics and world health: report of the advisory committee on health research (2002)
2. Hamburg, M.A., Collins, F.S.: The path to personalized medicine. New Engl. J. Med. **363**(4), 301–304 (2010)
3. Costa, F.F.: Big data in biomedicine. Drug Discov. Today **19**(4), 433–440 (2014)
4. Marx, V.: Biology: the big challenges of big data. Nature **498**(7453), 255–260 (2013)
5. Abiteboul, S.: Querying semi-structured data. In: Proceedings of the 6th International Conference on Database Theory, pp. 1–18. Springer, New York (1997)
6. Cattell, R.: Scalable SQL and NoSQL data stores. ACM SIGMOD Rec. **39**(4), 12–27 (2011)
7. dbSNP human build 142 (2014). http://www.ncbi.nlm.nih.gov/mailman/pipermail/dbsnp-announce/2014q4/000147.html. Accessed 16 Mar 2015
8. Ha, N.-T., Freytag, S., Bickeboeller, H.: Coverage and efficiency in current SNP chips. Eur. J. Hum. Genet. **22**, 1124–1130 (2014)
9. Sanger, F., Nicklen, S., Coulson, A.R.: DNA sequencing with chain-terminating inhibitors. Proc. Natl. Acad. Sci. **74**(12), 5463–5467 (1977)
10. 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. Nature **491**(7422), 56–65 (2012)
11. Maher, B.: ENCODE: the human encyclopaedia. Nature **489**(7414), 46 (2012)
12. ENCODE Project Consortium et al. An integrated encyclopedia of DNA elements in the human genome. Nature **489**(7414), 57–74 (2012)
13. Tuecke, S.: Grid security infrastructure (GSI) roadmap. Grid Forum Security Working Group Draft (2001)
14. Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L., Tuecke, S.: GridFTP: protocol extensions to FTP for the Grid. Glob. Grid ForumGFD-RP **20**, 1–21 (2003)
15. Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., Foster, I.: The globus striped gridFTP framework and server. In: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, p. 54. IEEE Computer Society (2005)
16. Martone, M.E., Gupta, A., Ellisman, M.H.: E-neuroscience: challenges and triumphs in integrating distributed data from molecules to brains. Nat. Neurosci. **7**(5), 467–472 (2004)
17. van Horn, J.D., Toga, A.W.: Human neuroimaging as a "big data" science. Brain Imaging Behav. **8**(2), 323–331 (2014)
18. Le Bihan, D., Mangin, J.-F., Poupon, C., Clark, C.A., Pappata, S., Molko, N., Chabriat, H.: Diffusion tensor imaging: concepts and applications. J. Magn. Reson. Imaging **13**(4), 534–546 (2001)

Chapter 2
Motivation and State of the Art

As we have seen in the previous chapter, data management and integration has become a major component of contemporary biomedical research. To be of any usefulness the flood of information produced by high-throughput genomic platforms (aCGH/SNP-arrays, DNA Microarrays, NGS technologies…) and high-resolution imaging platforms must not remain isolated in a sandbox, but must be integrated with all other available data about patient clinical history and lifestyle. This unified view is of paramount importance as healthcare paradigms move towards personalised medicine. A report published in *Nature* in 2013 highlighted that data storage and management requirements exceeded computational capabilities of one order of magnitude [1]. Management and integration of heterogeneous information has become an open biomedical research problem. The scientific community has proposed different solutions. So far, most of them are focused on specific subfields (i.e. functional genomics, mass spectrometry, computational neuroscience…). As multi-disciplinary collaborations become more and more pervasive in biomedical research, the limitations due to repositories focused on a single domain or a discipline must be overcome. In the next section I will introduce the concept of metadata, and discuss how they are used in information technology for data management. I will show that the concept of metadata as mere cataloguing tools has not changed as they were adopted in (biomedical) research, and that a new view on metadata must be attained as biomedicine is moving towards multi-disciplinary personalised medicine. Subsequently, I will illustrate the most common challenges a biomedical data repository must face, and then I will analyse some of the existing platforms. In the end I will address the missing aspects in the state of the art in order to envision a possible solution.

M. lzzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5_2

2.1 Data Management and Metadata

The two most used words in publications dealing with data management are not unsurprisingly "data" and "metadata". Before proceeding further, I will clarify—as much as possible—the meaning of the latter in the context of biomedical research. Metadata are usually defined as "data about data", or more extensively "data about data and all the processes that produce, streamline and output data".[1] In the most general definition, metadata are nothing else but description of 'things'. The described things can be physical (books, individuals, items stored in warehouse) or digital objects (files or other resources). In information technology and data management systems, physical objects will be mapped to digital entities (e.g. rows in a database). As a consequence metadata will refer to data that reside in the same physical (i.e. file system) or logical unit (database, digital repository, enterprise…). As stated by David Marco in his book *Building and Managing the Metadata Repository*, [3] "when we mention metadata we are talking about knowledge". Without metadata we might not be able to correctly interpret the data; they often contain essential information that could not be otherwise retrieved or reconstructed. For instance, if you take into consideration a three-dimensional MRI scan, the 3D image will be unravelled to a one-dimensional sequence of pixel in order to be written to a file. The information of width, height and depth of the image must be stored as metadata otherwise the image cannot be reconstructed univocally. Before becoming a hot topic in collaborative research, metadata were traditionally used to catalogue items such as books, articles, and magazines in the card catalogues of libraries. Metadata management in library catalogues was introduced through the adoption of Integrated Library Management Systems. The library paradigm later shifted to the information technology domain, with the institution of Digital Libraries, which are collections of digital objects having heterogeneous nature: books, images, videos, and so on. In these environments the main use of metadata is for the information retrieval. Some examples of digital libraries are online long-term archives like arXiv.org [4] and the Internet Archive [5]. In the last thirty years, science has become more and more data-driven and collaborative, both across scientific domains and geographical regions. Metadata have been more and more extensively adopted in research, and have gained a fundamental role, as the key to communicate data between groups in collaborations.

[1] This is now the most widely accepted definition in biomedical research and information science as well. However, it was not always so. In computer science, the term metadata was originally introduced by Philip Bagley [2] to encompass both "descriptive" and "structural" metadata. The latter category, defined as "data about the containers of data", specifies how the data is stored within a (digital) system, and has been the object of the ISO11179 standard specifications. The "validation" metadata category is a subcategory of structural metadata that provide validation constraints. "Guide" metadata are descriptive metadata that help the users to find and retrieve their data. In my thesis, whenever I will speak about "structural metadata" I will use the terms "metadata schema", "schema", or "metadata model", to distinguish it from the "descriptive metadata", that I will call simply metadata.

2.2 Data Sharing in Biomedical Research: An Open Issue

The constantly increasing usage of advanced imaging and high-throughput platforms, combined with the improvements of networking and fast computing technologies, have brought up an environment where global, multi-disciplinary collaborations between geographically distributed researchers are getting increasingly common. Data repositories offer an efficient way to share informations and analysis outputs of a study from different institutions in a multi-site collaborative effort, to provide larger datasets and common resources. However, imaging techniques and NGS platforms come in many flavours, are acquired in different modalities and process data using different analysis pipelines. This is a well known problem in genomics: for instance, Illumina technologies adopt a sequencing-by-synthesis approach [6] that employs fluorescently labelled reversible-terminator nucleotide, while Ion Torrent "harnesses the power of semiconductor technology" detecting the protons (i.e. H^+ ions) released as nucleotides are incorporated during synthesis [7]. As a result Illumina and Ion Torrent sequences are usually with different processing pipelines, and increasing efforts are required for comparison and integration. If follows that data originating from different sources or processing tools are heterogeneous in data format representation and metadata content. In principle, metadata allow scientist separated in time and space (different institution, country, regulation...) to interpret in the correct way the data they refer to. In this sense, metadata allow different individuals or research groups to find a "common ground" or understanding of the data. Usually the safest way to achieve this common ground is the definition of detailed and standardised metadata. Since the advent of Functional Genomics and sequencing technologies, the concept of metadata in biomedicine is usually accompanied with the concept of annotation. An annotation is metadata (usually in the form of a comment, explanation or tag) attached to some resource or data. It refers frequently to a specific portion of the original data. A genome annotation is therefore metadata containing biologically relevant information attached to a genomic sequence. Genome annotation usually describes the whole process of identifying the location of genes and other coding or non-coding regions in a genome, and determining what is the biological function of these genes or regions.

2.3 Metadata Standardisation

The adoption of standards, with extensive and highly structured metadata is usually invoked as the "holy grail", the best solution to optimise heterogeneous data sharing, to allow efficient and effective data analysis and to avoid misinterpretation and wrong data usage. According to this view, data repositories should enforce good management practices and standardised metadata. Shared repository should actually encourage—or somehow force—scientists adopting the aforementioned practices. Gray et al. [8] explicitly state that extensive metadata and metadata standards are one

the keys to achieve scientific success in the contemporary research scenario. He maintains that standardised metadata ease data discovery and understanding of datasets by scientists and processing tools alike. In the next subsections, I will show how standardisation efforts have been carried within and without the biomedical research field. Doing this, I hope to highlight the inherent limitations of the existing—and of any foreseeable—standardisation approach.

2.3.1 Approaches to Standardisation

The earliest metadata standardisation efforts occurred outside the domain of scientific research, to handle resource management in environments such as (physical or digital) libraries, stores, and warehouses. The first standards, such as the Machine Readable Cataloguing (MARC) were established in the 1960s to describe items catalogued in libraries. One of the first, simplest and most widely accepted standards for digital metadata management is the Dublin Core Metadata Element Set (DCMES), a set of 15 metadata terms that constitute the minimum requirement to describe any (web) resource [9]. Despite its original aim of achieving omni-comprehensiveness—that could be paraphrased with the motto "One metadata standard to manage them all"—the shortcomings of the limited 15-element DCMES have raised criticism for not offering the richness and specificity required for resource description outside the web [10]. In Fig. 2.1 is shown an example of resource description using DCMES, outlining the role of its elements.

A wide variety of more specialised metadata standards arose in the past years to describe text documents (Text Encoding Initiative, TEI) and heterogeneous metadata objects stored in digital archives, such as the Metadata Enconding Trasmission Standard (METS) and the Metadata Object Description Schema (MODS). In traditional data management scenarios, metadata are always seen as a fixed product describing a given physical or virtual object, to help cataloguing, search, and retrieval. There is no strong need for metadata to mutate, adapt and evolve. The approach to data and metadata management did not change when they started to be applied in the research field, even if the scientific domain had different requirements and a more flexible and dynamic nature. In fact, research paradigms, approaches and methods change quickly over time, as goals are constantly adjusted and corrected to take up with new discoveries and techniques. Despite this, in biomedical research metadata standardisation is usually achieved adopting the same two strategies used with success in digital catalogues. The first is the adoption of minimum information sets—the so called "metadata checklists"—while the second involves the use of shared controlled vocabularies. The two approaches are strongly related to each other and overlap to some extent. I will detail them in the following subsections.

```
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:identifier id="pub-id">
    urn:uuid:C42F6A69-5A7B-462B-B83E-CA6B90E3B484
  </dc:identifier>
  <dc:identifier id="isbn-id">urn:isbn:9780007322565</dc:identifier>
  <dc:title>The Silmarillion</dc:title>
  <dc:language>en-UK</dc:language>
  <dc:creator id="creator">J.R.R. Tolkien</dc:creator>
  <dc:contributor id="contributor01">Christopher Tolkien</dc:contributor>
  <dc:contributor id="contributor02">Guy Gavriel Kay</dc:contributor>
  <dc:date>2011-02-03T00:00:00Z</dc:date>
  <dc:description id="genre01">Mythopoeia</dc:description>
  <dc:description id="genre02">Fantasy</dc:description>
  <dc:description id="content-list">
    Ainulindalë, Valaquenta, Quenta Silmarillion,
    Akallabêth, Of the Rings of Power and the Third Age
  </dc:description>
  <dc:format>application/epub+zip</dc:format>
  <dc:publisher>HarperCollins Publishers</dc:publisher>
  <dc:rights>
    Copyright © 1977 The J.R.R.Tolkien Copyright Trust and C.R.Tolkien
  </dc:rights>
  <dc:source>urn:isbn:9780101010101</dc:source>
  <dc:subject>Morgoth</dc:subject>
  <dc:subject>Elves</dc:subject>
  <dc:subject>Silmarils</dc:subject>
  <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
  <meta property="dcterms:modified">2015-01-29T12:13:00Z</meta>
</metadata>
```

Fig. 2.1 The metadata element of an electronic version (EPUB 3.0 format [11]) of the book "The Silmarillion" [12], as specified using the DCMES. EPUB stores metadata in XML language and the Dublin Core Metadata Initiative (DCMI) elements are provided in the `dc:` namespace. For the EPUB 3.0 specification, the `identifier` (a universal unique ID (UUID), ISBN, ISSN or DOI), `title` and `language` elements are required together with the `modified` meta property. All the other elements are optional. Notice than more than one value can be provided for each field. The `source` element identifies the physical book from which the ebook was derived. The `type` element describes the nature of the document (i.e."text") using the DCMI Type Vocabulary. Two of the original 15 DCMES elements are missing in this example: `relation` and `coverage`. Further details about DCMES elements can be found at [13]

2.3.2 Minimum Information Requirements and Metadata Checklists

A minimum information standard is constituted by a set of guidelines for reporting experimental data derived by relevant method in biomedical science. These guidelines aim at ensuring that the data can be easily verified, analysed and interpreted by the scientific community. The overall goal of these efforts is to standardise the

annotation and curation processes of the experiments, providing specifications about which metadata is crucial together with the experiment's output data to make it comprehensive. One of the first efforts was the Minimum Information About a Microarray Experiment (MIAME), first outlined nearly fifteen years ago [14]. At that time, DNA microarray analysis was already widely adopted to generate gene expression data at a genomic scale. MIAME established a standard for recording and reporting microarray-based gene expression data, and it aimed at easing the development of databases and public repositories, together with standardised data processing tools. MIAME does not enforce a specific format, though formats that facilitate data querying such as spreadsheet (MAGE-TAB [15]) or XML (MAGE-ML [16]) were strongly encouraged. In its first specification, no terminology was adopted or specified to constrain the metadata values. MIAME compliant data are managed, stored and distributed by public repositories such as Array Express at EBI (UK), GEO at NCBI (US) and CIBEX at DDBJ (Japan). In the subsequent years, the scientific community specified other minimum information guidelines. Among others, noteworthy are the Minimum Information for publication of Quantitative real-time PCR Experiments (MIQE) [17] and the Minimum Information about a functional Magnetic Resonance Imaging study (MifMRI) [18]. In 2008 all these efforts were for the first time coordinated within the Minimum Information for Biological and Biomedical Investigation (MIBBI) project [19], which now provides a web-based, freely accessible resource for checklist projects, providing straightforward access to extant checklists, together with controlled data vocabularies, software tools and public databases. Not all the metadata checklist efforts are undertaken within the MIBBI consortium. The Biobanking and Biomolecular Resources Research Infrastructure (BBMRI) was instituted to harmonise biobanking management and procedures across Europe. BBMRI developed a minimum data set for biobanks and research studies using human biospecimens [20]. This data set—the Minimum Information About BIobanking data Sharing (MIABIS)—consists of 52 attributes that—according to the proponents—provides the minimal description of the biobank's content. The harmonisation of metadata elements referring to biobanks using the MIABIS standard should improve and facilitate samples' discovery, resulting in time and cost savings and faster emergence of new scientific results.

The tremendous momentum generated by NGS technologies has also prompted in the last few years the need for novel standardised metadata checklists. In January 2014 *OMICS: A Journal of Integrative Biology* has launched a coordinated initiative for a comprehensive and flexible multi-omics (thus spanning through genomics, proteomics, metabolomics, and so on...) metadata checklist [21]. The initiative aims at enabling a thorough use of single and multi-omics datasets thanks to data harmonisation and improved visibility and accessibility.

All these standardisation efforts through checklists show a wide range of success and acceptance by the research community, as one can see making a citation search for each one of the resources mentioned above. Even the most widely accepted standards, such as MIAME, have their shortcomings that are summarised in dedicated publications [22]. An issue considered by the authors is that implementing MIAME requirements has turned out to be challenging. Specifically, since MIAME does

not provide an explicit format for representing data, no standard computer-readable format has been adopted. The MAGE-ML standard did not gain wide popularity, because of its high complexity, caused by the intricate structure of XML schemas. The much simpler MAGE-TAB format, which does not require any particular tool for viewing/editing, has gained some popularity and novel more general formats (such as ISA-TAB) tend to steer away from the XML complexity and keep the simplicity of spreadsheet formats. As a general rule, the authors observe that developing and adopting computer-readable standards is more difficult than adopting general guidelines. Simplicity is the key to success to gain consensus within the research community but this is not an easy task to achieve. A minimum set of information cannot fit all needs of a community and is not conceived to tackle the requirements of the increasingly dynamical and multidiscplinary field that is biomedical science. I will proceed to illustrate an example on this in Sect. 2.4.

2.3.3 Controlled Vocabularies: Taxonomies and Ontologies

While several minimum information standards or metadata checklists do not enforce the usage of controlled vocabularies, they frequently encourage their adoption. A *controlled vocabulary* is an authoritative list of terms that is used in indexing. Controlled vocabularies are adopted for consistent indexing (i.e. when indexing multiple documents) and do not necessary specify a structure or relationship between the terms in the list. Controlled vocabularies are a broad category which include more specialised classes: thesauri, taxonomies, and ontologies. A *taxonomy* is a controlled vocabulary with a hierarchical structure. There are relations between terms within a taxonomy, and they usually represent parent-child relationship. A *thesaurus*, is defined in the literature retrieval and the information science as a controlled vocabulary where all terms have relations to each other. There are typically three kinds of relationships: hierarchical (parent/child, as for taxonomies), associative and equivalent. They are scarcely used in life science, and their main field of application is the indexing of periodical literature. An *ontology* is a category of taxonomies with structure and specific types of relationships between terms. An ontology supplies a greater variety of relationships than the simple hierarchical associations supported by a thesaurus. Each relationship is specified in its function. Ontologies define relationships and attributes that are specific to a particular business area. They have gained a significant popularity and are extensively adopted in biomedical science. Ideally, an ontology well-fitted for research should enjoy the following good properties. It should be:

- open, so that the ontology and the body of data described in its terms should be available to the whole research community at no cost whatsoever. Being open also mean a keen receptivity towards changes driven by community debate;
- orthogonal, to provide the benefits of modularity and ensure the additivity of annotation;

- instantiated in a well-specified syntax to support algorithmic processing;
- equipped with a common system of identifiers to ensure backwards compatibility with legacy annotations as the ontology is updated.

The Open Biomedical Ontology (OBO) consortium applies the principles outlined above to the development of life science ontologies [23]. All the ontologies validated by OBO are collected and published at the OBO Foundry website. Each OBO Foundry ontology satisfies these requirements: (i) it possesses a unique identifier space, (ii) it is, or can be, expressed in formal language, (iii) it includes textual definitions for all terms, and (iv) it adopts unambiguously defined relationships according to the specifications outlined in the OBO Relation Ontology [24]. The Stanford Center for Biomedical Informatics Research of the Stanford University has developed BioPortal [25, 26], a community-driven repository for Biomedical Ontologies. The portal is equipped with a REpresentational State Transfer (REST) interface, to access ontologies and their components. Ontologies provided by the OBO foundry constitute an important core of the BioPortal ontologies. As opposed to OBI guidelines, there are few constraints on the BioPortal ontologies, provided that the ontology has some level of relevance to the domain of biomedicine and it is written in a supported format.

2.3.4 Biomedical Metadata and the Semantic Web

Ontologies provide a systematised knowledge for a specific domain, and as a consequence provide semantic content for each element and relation they define. The connection between ontologies and the so-called Semantic Web—a (mostly theoretical) extension of the Web where all the resources are described in a way that machines can understand and process to achieve inter-domain data linkage [27]—becomes apparent exploring the ontologies contained in BioPortal. Most of the over 300 ontologies stored on the BioPortal repository are either in OBO format or in the Web Ontology Language (OWL), a World Wide Web Consortium (W3C) recommendation for representing ontologies in the Semantic Web. The BioPortal dataset contains also metadata related to each ontology and mappings among terms in different ontologies. Mappings are either submitted by users or generated automatically by internal procedures of the system. Users can access the ontologies in BioPortal using the SPARQL query language or retrieving de-referenceable terms and ontology Internationalised Resource Identifiers (IRIs) [28] in Resource Description Framework (RDF) [29] format. The RDF data model is a core element to describe resources in the Semantic Web. It structures any semantic expression as a triple, consisting of a subject, a predicate and an object. A set of triples composes an RDF graph, where subject and object are the nodes and the predicate is a link defining a relationship. Nodes and links are univocally identified by a URI (or more in general, by an IRI). The RDF specification is implemented in various formats: the two recommended by the W3C are RDF-XML (the first standard format, an XML-based syntax) and Turtle,

a compact and human-friendly format. The eXtensible Markup Language (XML) is used to formalise metadata and metadata schemas. XML permits automated parsing by software tools and the addition of semantic content. Addition of semantic content to metadata has achieved a limited consensus, despite strong efforts by the W3C consortium and the RDF Working Group. There is strong debate in the developer community on the reasons behind this poor success and various indications have pointed out some drawbacks of the semantic tools like RDF/XML, NTriples, Turtle, SPARQL and so on. These tools do no provide native support for lists and generalised graph structures and they have been accused of "creating esoteric solutions to non-problems" [30]. In practice, most of the developers of data-driven applications do not need to adopt semantic web solutions for managing their when there are more "natural" and simpler solutions to address the problem. If this is true for software developers, it will be even more true for biomedical scientists and researchers who have different goals than spending time and efforts to provide semantic content to their datasets. In conclusion, semantic support should be provided when the people involved in a project feel comfortable in using it, but not enforced or tightly coupled to the metadata schema of the digital repository.

2.3.5 Established Standards for Data Exchange in Clinical Research

Numerous data models equipped with extensive metadata support have been proposed and adopted for Clinical Data Exchange. The most notable is the Health Level 7 (HL7) Clinical Document Architecture (CDA), a document markup standard that specifies the structure and the semantic content of a clinical document for the purpose of exchange [31]. It employs an XML format and can contain any form of multimedia including text, images, videos, and so on. CDA tracks administrative workflow together with clinical reports. An important aspect of CDA is the focus on document exchange rather than data sharing, and the XML format is adopted precisely for this scope. I will return on this point later on.

The Clinical Data Interchange Standards Consortium (CDISC) has produced various standards to manage both patient health care and biomedical research activities. Their Operational Data Model (ODM) is yet another XML-based schema designed to facilitate the regulatory-compliant acquisition, archiving, and interchanging of metadata and data for clinical research studies, and it is mainly focused on questionnaire-based clinical trials [32]. Moving aside from pure clinical scenarios, the CCLRC Scientific Metadata Model provides a general paradigm for storing metadata of scientific provenance. Unfortunately it does not provide a subject-centric view, that is seen as quintessential by the great majority of clinicians and biologists.

2.3.6 The XCEDE Schema

The XML-based Clinical and Experimental Data Exchange (XCEDE) schema provides an extensive metadata hierarchy for storing, describing and documenting the data produced by scientific studies [33]. Despite its omni-comprehensive objective—somehow similar to DCMES in scope, if not in the applicability domain—it has been used mainly in biomedical sciences and especially in computational neuroscience. XCEDE hierarchical structure models scientific experiments using a set of hierarchy levels; each level can be annotated with specific metadata. Structured data, such as time event-based data or clinical assessments and questionnaires can be stored directly within the XML schema. XML has been adopted to provide a standardised way to transport and interchange scientific data easing import/export procedures between heterogeneous data sources, development of specialised web services, local storage of experimental information within data collections, and creation of human and machine readable descriptions of the actual data. XCEDE version 2 has adopted reusable abstract data types, novel components to model analyses and terminologies, and is built on a hierarchical structure for defining experiments. Data associations are mapped using lists of elements linked by identifiers, thus permitting an easier integration with relational databases.

XCEDE is built on eight main components:

1. an **experimental hierarchy** of multiple levels that allows the subdivision of an experiment at different granularities, meaning that the user can omit some levels. The *project* represents the top level element and collects multiple *subjects* or *subject groups*. A *visit* represents the subject's appearance at the clinical institution or experimental site, and it may consist of several *study* elements. Each study contains one or more data-collecting *episodes*. An *acquisition* is defined as the data produced during a single episode.
2. **Resources**. XCEDE elements can associate data to resources, (heterogeneous external entities). There can be *information resources*—usually described by the 15 fields of the DCMES—that point to documents, publications or web pages, and *data resources* that work as pointers to external files containing bulk data or additional unstructured metadata.
3. **Protocols** are defined as a (possibly hierarchical) sequence of experimental steps.
4. **Structured Data** can be stored directly within the XML schema using the specific <data> tag. XCEDE natively supports internal storage of clinical assessments and events; the latter are defined as time interval annotated with metadata
5. **Analysis** is a component used to document the result of data processing steps. It consists of inputs, a list of the employed software tools/methods, and the output value(s) and/or file(s).
6. **Catalogues** are used to index and collect together data of interest. Catalogues can be nested within another catalogue to build a multi-level hierarchy.
7. The data **provenance** module monitors the origins of data and the processing steps. It allows repetition of processing workflows to test the results' replication.

8. A **terminology** component adds semantic content to the elements' content, linking them to terms within a terminology. Given the main focus on neuroscience experiments, there is support for Brain Atlases and ontologies deputed to large-scale experimental annotation.

2.4 The Shortcomings of Standards in Research Collaborations

In any real-life scenario—and biomedical research is no exception—standards take time to develop. The first issue is reaching a consensus, or "common ground" among people with different expertise. Usually this process requires numerous corrective steps, when the actors with different vocabularies negotiate a common terminology. In a multi-disciplinary collaboration the same word(s) may convey a different meaning for scientists trained in different disciplines. The second issue is due to a diversity of objectives. Many times, metadata standards do not meet the investigation purposes. Moreover, formal metadata conformant to a standard are not employed on the daily routine or on a local basis. They are not felt as a priority but as a hindrance by the researchers. A third complication is related to the multi-disciplinary nature of many biomedical studies: research in the field can no more be bound within the limited borders of a single domain. In practice, in any scientific work, well-refined metadata products do not exist, even though they would be strongly desirable (especially by the scientists themselves!!). The concept of metadata as a product works well in the settings where they were originally adopted: physical/digital libraries and archives and businesses equipped with inventories. In collaborative research, uncodified and informal knowledge plays a key role for correct data understanding. How is this usually achieved? Using incomplete, loosely structured, *ad hoc*, and mutable descriptions. The pieces of information are generated on-the-fly during communications among researchers, as soon as a common terminology is reached or restated. As evidenced in a study by Edwards et al. focused on collaborative project on different scales [34], metadata in scientific research can be envisioned as ephemeral process, rather than a fixed, enduring product. Edwards observes that metadata seen as an evolving and dynamical process enjoys some properties that make them difficult to handle.

The metadata process:

- is **fragmented**, as it involves many contributors (i.e. individuals and research groups);
- is **divergent**: often, beside the standardised efforts (and despite of them), other metadata appear. The latter are usually simpler, more widely used, and are communicated in "unconventional" ways, like emails, phone calls and face-to-face conversation, rather than data management systems and repositories;
- is **iterative**: both the metadata fields and their content is constantly changed and repaired as the common terminology shifts in time;

- is characterised by a **local-centred** focus, as for any scientist the internal usage of data and the personal goals come before long-term sharing purposes.

Therefore, metadata management requires some level of paradigm shift to reach a level of adaptability able to satisfy the researchers' needs. As long as the focus remains on fixed and highly defined metadata, there will be a strong friction and misunderstanding in data sharing and communication among scientists. Biomedical science makes no exception to this rule: there are different situations where the inherent complexity of biological and medical data cannot be captured or synthesised only with the adoption of standards.

In particular, data in omics and neuroscience—the fields of life science that mostly fall in my area of interest and that are converging faster one towards the other—are subjected to a variety of local constraints, institutional practices and regulations and specification due to the manufacturer of the analysis platforms. In these fields, a great part of metadata is generated by scientific instruments—such as DNA sequencers, CT/MRI scanners, mass spectrometers, …—and their elements are specified by the manufacturer. Different vendors adopt different strategies even if they are using similar metadata fields. A data repository must find a way to manage and possibly integrate different metadata schemas—that is the way metadata is stored and contained within the database or file system—describing the same data type (i.e. an MRI image, or a whole genome sequencing assay). Ideally, a digital repository should provide automated mapping to translate the data content and its metadata from the data source to the repository internal model. In practice, given the divergence of formats, this approach is not feasible. Neu et al. [35] have shown the inherent difficulties of heterogeneous data management and metadata standards in neuroimaging collaborative studies. Radiological images come in a plethora of formats, devised in different periods by groups of people having different goals. Formats are different: the most used are DICOM, Nifti, Analyze 7.5. In the early 1980s, the DICOM standard was established to make images independent by the scanner manufacturer. Over the last thirty years DICOM has gradually changed to keep pace with the evolution of scanner technologies. Even though a single standard was provided, each manufacturer has applied with different approaches and it has added proprietary metadata to store values that were not supported in the standard specification. DICOM provides both public and private tags to store metadata. The latter should be used by the vendors for non-standardised metadata. In practice, private tags are used by manufacturers even when a public tag is provided, if the public terminology is not consistent with the manufacturer requirements. For instance, many of the imaging modalities used in current medical research, like Diffusion Tensor Imaging (DTI), functional MRI (fMRI) and Magnetic Resonance Angiography (MRA) are not recognised by the DICOM standards, that collects all of them under the term "MR". To distinguish the three different modalities, the scanners adopt private tags and these are different among the manufacturers. Even if we consider the relatively constrained domain of Neuroimaging global consistency cannot be achieved for a number of motivations:

- radiological images are used in the clinical and research domain, with different purposes. From the point of view of the manufacturer, the clinical domain has often a higher priority and frequently a more consistent terminology than the research counterpart. An issue to consider when transferring radiological images from the clinical field to research is a consistent anonymisation with removal of all the metadata that refer to the patient identity and other sensitive information;
- research is less regulated and more dynamical, so it is less prone to adopt fixed and standardised terminologies (and this is the same point highlighted by Edwards);
- when different groups collaborate, they try to find compromises to achieve a terminological consistency;

The dynamical nature of research and the rapid evolution of technologies, require metadata that vary over time. Variations in metadata are required on a continuous basis for novel experimental acquisitions and for innovative research protocols, like those that employ pharmaceuticals never tested before, and that are not supported by the current terminology.

Researchers are frequently forced to adopt non-standard solutions during their daily activity. There is therefore the need for innovative data repositories that can adapt data models to mutating requirements, to describe novel data types or to extend existing ones. Standardisation alone cannot achieve heterogeneous data integration and an optimal sharing of information when scientists of different disciplines are collaborating in a multi-disciplinary project. As research collaborations are moving constantly to geographically distributed efforts among groups with different background the repositories must also implement adaptive metadata management tools to share data on a national and global scale. As a note, the end user should be able to extend the metadata model without resorting to computer science specialists. The repository should provide a graphical interface for data type definition and modification, where metadata fields can be added or modified, vocabularies and terminologies can be extended, and when necessary semantic content can be retrieved from an ontology to annotate the metadata fields.

2.5 Data Repositories: The State of the Art

I will now proceed examining the mostly used digital repositories and data management system for biomedical science, with a particular interest on neuroscience, Functional Genomics, and Integrated Biobanking. In the end I will draw a comparison to highlight the strengths and the deficiencies of these platforms to support a dynamical and adaptive management of metadata in a multi-disciplinary collaborative scenario.

2.5.1 XNAT

The eXtensible Neuroimaging Archive Toolkit (XNAT) platform is one of the oldest and more established data repositories for Neuroscience [36]. XNAT is an open source software suite developed by the Neuroinformatics Research Group of St. Louis, Missouri, to address and facilitate data management challenges in Neuroimaging studies. While XNAT supports mainly DICOM images and reports, it can, at least in principle, store data of different types. XNAT has automated tools to capture data from multiple sources, keeps them in a secure repository and distributes the data to authorised users. XNAT relies heavily on XML and XML Schema [37] for data representation and for other repository functions such as security management and generation of user interface content. XML Schema was chosen as it was the W3C standard specification to extend XML data formats. XNAT uses XML Schema Definition (XSD) to define data types and to generate custom components, graphical and logical content for its Presentation, Application and Data tiers. Moreover, XML is employed for security, input validation and queries. Imaging data are stored in their native format on the platform file system: a link to the file URI is stored within the database. XNAT provides neuroscientist with an ad hoc workflow for neuroimaging data acquisition and sanitising, that consists of automated data and metadata capture directly from the scanners followed by a strict quality control procedure. Non-imaging data are first put in a virtual quarantine and must be verified by a qualified operator. This wide adoption of XML schemas poses a certain number of drawbacks. Inefficient metadata storage and querying is solved saving numeric and textual fields directly within the database. Another problem that is not so easy to solve, as the authors recognise, is the poor efficiency of the database tables generated from the XSD, due to sub-optimal mapping. As a consequence of the dichotomy between XML schema and database representation, whenever a change is made to the data model it must somehow propagate to the database. This operation usually requires manual changes by an administrator and cannot be independently executed by a user without some level of computer literacy. The core data model of XNAT bears some resemblances with XCEDE—the authors of XCEDE state on their publication [33] that they have been developed to complement each other—and consists of three main data types: *Project*, *Subject* and *Experiment*. A single project is the "owner" of a Subject and Experiment entities, but it is possible to share them across projects. Experiment represents the event when new data are acquired. Experiment is an abstract model. From it derives *Subject Assessment*, and from this in turn derives *Imaging Session*. Imaging Session has three specialisations according to the scanning modalities accepted by the DICOM standard: *MR Session*, *PET Session* and *CT Session*.

XNAT relies on a three-tiered software architecture made of a PostgreSQL database back-end, a Java-based middleware tier usually deployed on an Apache Tomcat servlet container, and a web-based user interface. The XSD-to-database mapping happens as follows: each one of the XML Schema global elements is mapped to a single database table, with its sub-elements mapped to the table columns. Foreign key and complex associations (one-to-many and many-to-may) can be constructed from

the XSD. If there is more than one XML Schema additional tables are created using different namespaces to avoid overwriting. To allow interaction with analysis tools and external platforms, XNAT exposes a comprehensive RESTful Application Programming Interface (API). Recently XNAT has been equipped with a data dictionary service that allows to define relationships between data elements and taxonomical structures across the XNAT installation [38].

2.5.2 COINS

A candidate competitor of XNAT is the Collaborative Informatics and Neuroimaging Suite (COINS) project, developed at the Mind Research Network (MRN) of Albuquerque, New Mexico, USA. COINS is a data repository focused on centralisation of neuroimaging datasets from multiple studies. The focus is in building a single centralised infrastructure for neuroimaging datasets from multiple studies, rather than developing a distributed network, but the overall goal remains to maximise data sharing and reuse [39]. The authors highlight the documented benefits of a centralised approach: reducing costs, increasing the citations' rate (due to multiple cross-referencing), and the possibility of novel discovery through datasets reuse. The COINS framework backbone is constituted by a well structured taxonomy for data and data sharing. The taxonomy is twofold: on one side, it classifies the data by plurality (singleton or collection), medium (document or digital file), confidentiality (sensitive or insensitive), sense (data or metadata), source (recording of observation or derivation) and mode of acquisition (by humans or by instrument). On the other side, sharing is classified by a source entity (i.e. institution), a target entity, the possible sharing operations (intersections or unions of datasets across studies), the delivery venue (in situ or ex situ), the transfer method (through computer network, by courier or *a manu*), and the security (encryption on source, transmission, or target).

In the first attempt to categorise in rigorous way the difficulties posed by data sharing in a biomedical context, COINS developers identify five main challenges:

1. Secure Personal Health Information (PHI) management.
2. In situ versus ex situ sharing. The centralised repository helps avoiding the latter, that is copy or transfer of data outside the repository domain. All the users can log into the COINS platform and access the information in situ. This is actually a requirement that all the biomedical data repositories should satisfy.
3. The adoption of standardised metadata without undermining the extensibility to novel data types. The authors recognise that this is a major issue. They address the support for non-standard DICOM metadata—such as those describing DTI or fMRI acquisitions—with customised methods for the extraction of vendor-specific metadata fields. They stress the point that these automated procedures require continued maintenance for updates to the new scanners and technologies. COINS adopts an EAV catalogue to allow some additional flexibility for data types not natively supported by the system.

4. Intuitive Ease of Use (IEU): it is emphasised by the authors, and rightly so, that new users should require a very small effort to be productive when using the repository. Otherwise, they will soon stop using it, and they will look for easier and more profitable ways to share their datasets.
5. Expose a uniform, friendly, powerful query interface
6. Check data provenance and, when required, modify/correct metadata.

Similarly to the previously exposed platforms, COINS is developed using the established Java-based software technologies and relies on PostgreSQL. It currently supports the following data types: MRI, EEG, MEG, genetic data, neuropsychological and clinical assessments. The first four types are collectively labelled NeuroImaging Data (NID), while the last two are labelled as Neuro-clinical Assessments (NA). NA data and ND metadata are stored on the database, while the bulk ND data are stored on a file system. The COINS platform is built of five main modules, that I briefly describe. Users access the systems via *web portal*. Different studies can have their own web portal, reducing security risks; moreover, portals do not store any PHI or identifier. The *assessment manager* allows dual-entry conflict resolutions for NA that are entered by humans, and might require manual checking and validation. Data is submitted via web form. Only free-text and drop-down options are allowed, which I personally feel as an over-exemplification, if researchers would like to store information different from NA. Data can also be collected from a *tablet-based client* and sent to the COINS server using a web service API. NID upload is handled by a customised DICOM receiver based on the dcm4che Java library. At the time of the referenced publication, no other formats than DICOM were explicitly supported for data upload. A graphical *query builder* module provides an intuitive interface, that is based on a query-by-example approach, without requiring knowledge of database structure or table associations. Authorised users can query data from multiple studies. Frequently used queries can be saved and reused later. The most important module of COINS is the *Medical Imaging Computer Information System* (MICIS), devoted to studies, subjects and scans management. MICIS supports definition of custom subject types, for instance to distinguish between patients and controls. It is also equipped with a mechanism to unlink PHI from an entire study when the study expires. As of February 2015, COINS installation at MNR managed nearly 31,000 subjects from 558 studies, with more than 38,000 scan sessions and over 4,90,000 neuropsychological assessments [40]. In the end, the system main characteristics are the adoption of a centralised infrastructure to avoid ex situ sharing and wide usage of EAV catalogues to handle data extensibility without using explicit schemas in XML or other formats.

2.5.3 CARMEN

The Code Analysis Repository and Modelling for e-Neuroscience (CARMEN) system has been developed in the UK to provide a web-based portal platform to share and exploit datasets, analyses, code, and expertise in neuroscience. It provides four

main types of assets: data, metadata, services and workflows. The authors state that "data and metadata are currently structured for neurophisiology, but the mechanisms for data and metadata management are generic and hence the platform is applicable for any science domain", a somewhat bold assertion that is neither proved nor disproved in the publication [41]. The CARMEN system employs a data format and schema that has been agreed by the project collaborators. This might prove a strong limiting factor for the data model extensibility, since it is not possible to modify the model directly without accessing and operating on the source code. In CARMEN, users can generate pre-populated templates to ease the metadata entry procedures. The templates will automatically populate the data entry forms, and the user will just have to update the fields that change during the experimental protocol. When the system was originally published in 2011, metadata upload from XML files was considered as a future possibility but was not currently supported by the system. Despite its lack of flexibility, a noteworthy feature of CARMEN is the adoption of Storage Resource Broker (SRB), a file virtualisation system for distributed file storage. None of the other repositories presented in this section is currently integrated with a distributed file management system. CARMEN also offers tools for data analysis and workflow configuration. Processing applications are made available in an interactive Software as a Service (SaaS) model for end users, thanks to a Java wrapper module that handles command-line tools written in a variety of languages.

2.5.4 XTENS

The eXTENsible platform for biomedical Science (XTENS) digital repository was originally developed by Corradi et al. at the Department of Informatics, Bioengineering, Robotics, and Systems Engineering (DIBRIS) of the University of Genoa to support integrated research in neuroscience, with a particular focus on Neuroimaging [42]. Its data management paradigm was designed to handle a various range of situations and environments in biomedical research, and already incorporates a basic sample management system, a feature not yet supported by any other repository examined in this survey. XTENS allows the generation of several different data types according to structured schemas. In this respect, XTENS shows some point of contact with the XCEDE data model: data types are described by an XML metadata schema associated to XSD and XSL files to respectively validate its structure and define its representation [43]. The repository can be configured to store the metadata totally or partially in the database. The metadata are stored as XML descriptions inside the data table, to display the data in a rapid and dynamic way using XSL Transformations [44], and as records of specific metadata tables, to perform complex queries in an easier way. The most striking difference with XCEDE, and all the platform I have reviewed so far, is that XTENS abstracts Experiments, Studies, Visits, Episodes, and Data Acquisitions using a taxonomic model, built of two entities: *Process* and *Event*. An event can be any 'atomic' operation that is performed on patients or samples, or any processing of data or everything else related to the XTENS repository

administration and management. A process is defined as a group of sequential events or sub-processes related to an activity, allowing the creation of a flexible and yet hierarchical structure. A data instance in XTENS is defined as the output of an Event. The main innovative point of XTENS is that the extension of the data model through the definition of new data types can be easily performed by users through an intuitive graphical interface.

The XTENS repository architecture consists of:

1. A web portal, that provides a client interface and allows users to access and to manage database requests. The XTENS portal is a Java Server Pages (JSP) and servlet application deployed on Tomcat. To better enhance user experience and interactivity, various components are designed using Asynchronous JavaScript and XML (AJAX) programming technique. Client and server exchange messages using JSON through JSON-RPC protocol whenever possible.
2. A MySQL relational database. Database access from the web application is managed with MyBatis, a persistence framework that automates mapping between SQL databases and Java objects. The MyBatis persistence layer permits to adopt, if required, a different SQL RDMS (PostgreSQL, Oracle,. . .) with moderate effort. The database contains all the information about projects, patients, data and everything related to the repository management (users, groups and accesses).
3. A data grid storage element, which contains all the files associated to registered data instances. The data grid of choice is the integrated Rule-Oriented Data System (iRODS) middleware. SRB, the distributed file system adopted by CARMEN, was a precursor of iRODS. iRODS possesses an internal metadata catalogue and the administrator can configure XTENS to store metadata both on the internal database and on the grid storage metadata catalogue, or only on one of the two systems.

The users access the system using an existing LDAP or database account available on the server. Each user is associated to Access Control Lists in order to guarantee security and auditing. The access is via web browser without any client installation and in a secure way through the HTTPS protocol. Authentication and access-control is managed using the Spring Security framework. XTENS addresses possible security and privacy policies regarding the access to proprietary data and sensitive clinical data. This is achieved by a thorough customisation of user permissions, defined by administrator-defined entities called *functions*. Authenticated users are allowed to view, insert, modify and retrieve data according to the set of functions enabled for their own group. Administrators can define groups of users associated with different access permission to the application pages and functions.

2.5.5 SIMBioMS

The System for Informative Management in BioMedical Studies (SIMBioMS) was probably the first open-source platform designed to integrate phenotype information with genomic data produced by high-throughput profiling technologies. SIMBioMS

authors state that the driving force behind their development effort was the lack of dedicated system for Collaborative Projects that could provide support both for subject and biological sample management. SIMBioMS was originally developed in the framework of a multi-site project. Afterwards it proved to be sufficiently customisable and scalable to be adapted for other research collaborations focused on population genomics [45]. SIMBioMS supports data-entry via graphical forms, and provides facilities for data import and export. The platform can be configured to satisfy the MIBBI requirements, while data can be exported according to MAGE-TAB, ISA-TAB and customised XML and tab-delimited format. A query-by-form interface provides content exploration and report building. SIMBioMS has a modular structure and consists of two main components, that can be installed separately:

- The **Sample Information Management System (SIMS)**, stores and manages phenotypic, environmental and technical information about the collected samples. It provides four main data types: Patient, Visit, Sample and Aliquot. A patient can undergo many visits and have many samples stored within the system. One or more aliquots may be extracted from each sample for analysis purpose.
- The **Assay Data Information Management System (AIMS)** handles the experimental output for a variety of high-throughput technologies. When the platform was originally published in 2009, no built-in support was yet available for NGS. AIMS provides a hierarchical structure where an Experiment contains multiple Studies consisting of many Assay. The Assay entity provide the link with the SIMS module: an aliquot is used for one or more assays.

From the technological point of view, SIMBioMS is a Java-based application running on Tomcat and supported by a PostgreSQL database. The communication between the application classes and the database entities is handled by Hibernate [46], a popular Object-Relational Mapper (ORM). As it can be seen from the SIMS configuration guide available on the internet [47], a modification to the data model—like adding new metadata fields, or changing the names of the existing ones—require modifications to the Hibernate mapping files and to various XML configuration files of the application. These operations require at least a programmer or software installer with some previous experience, and does not provide a full control of the data model to the end-user (i.e. the scientist). Metadata and controlled vocabularies can be imported in the system using once again XML files, but the SIMBioMS developers advise the reader that this is an error-prone procedure and must be undertaken with great care.

2.5.6 openBIS

The Open Biology Information System (openBIS) software suite [48] is a data repository tailored for long-term collaborative projects adopting cutting-edge technologies, where migrations of data and support for new data models are frequently needed. It has been developed to support data management in systems biology since the acquisition from a source—such as a microscope, a mass spectrometer, a sequencer…—to

the publication. Similarly to XNAT and SIMBioMS, openBIS provides a canonical hierarchy for Biological data management with four main entities: *Project*, *Experiment*, *Sample* and *Data*. There is no explicit support for patient personal data and electronic health records, and no issues of anonymisation are tackled in the publication. openBIS's area of competence is systems biology and a distinct integration system is needed to integrate clinical information with the high-throughput omics outputs. Another drawback is the tight coupling between the main data types, especially between Experiment and Sample. An experiment can contain one or more samples, but the reverse is not true. Therefore, there is no support many-to-many association between samples and experiments,that limits the applicability if compared to SIMBioMS, where a sample could be fragmented in many aliquots assigned to one or more assays. There is no separation between the sample management domain and the analysis domain, as it was the case with the SIMS and AIMS modules of SIMBioMS. Nonetheless, openBIS shows a lot of nice features that were missing in the other platforms. Dedicated mechanisms for data upload, metadata annotation, and flexible querying have been developed for the most important fields of System Biology research: NGS, quantitative imaging, and mass spectrometry for proteomics and metabolomics. The API has been designed using a "loose coupling" approach, to expose a unified façade to external programs. Metadata are managed separately from bulk data to ease scalability issue. Metadata are stored within the relational database located on the so-called *Application Server*, while the bulk data is stored in one or more file systems managed by *Data Store Servers*. In this way, bandwidth consuming operations like file uploads are not performed on the system that provides metadata management, and that might be used frequently for searches and retrieval. In openBIS, datasets are immutable: once uploaded, they cannot be modified any more. If a dataset must be modified or new data are derived from it, the novel or updated data will be saved as a child dataset for internal consistency and information traceability. The data content is separated from its representation. If a dataset possesses different representations, the user can create multiple datasets within a single dataset container: openBIS will show the different representations as a single entity, making the duplicates transparent for the user. Data upload for small and medium files is handled through a web-based drag and drop interface: for the supported data types metadata is automatically extracted and saved on the database using dedicated Extraction, Transform and Load (ETL) procedures. Metadata can be exported as spreadsheets, and users can download bulk data directly from the web interface of using the command line. All the services offered by openBIS are available to external applications through a REST API, to allow third-party data retrieval, analysis, and visualisation.

The underlying openBIS data model divides metadata in three different categories:

- **Structured metadata**: these represent custom properties and annotations and are stored within the database as single fields. Each structured metadata element belongs to a *property type*, that determines whether it is a textual field, a number, an email, a hyperlink, a date or a constrained value selected from some terminology or controlled vocabulary.

- **Semi-structured metadata** can be stored as an XML schema directly on the database. There is no mention of tools for building or formatting these schemas, so my guess it that you have to provide them already well structured and they will be validated against an XSD.
- **Unstructured metadata** such as free-text can be provided as a file attachment and associated directly to the Project, Experiment or Sample entity.

An outline of openBIS data model is shown in Fig. 2.2. Structured and semi-structured metadata are stored using the Entity–Attribute–Value (EAV) model, a paradigm that we will see often used to build up metadata catalogues. The user can extend the data model creating new Property Types and attaching them to any if the four hierarchical components of openBIS. More details about the EAV model can be found in Sect. 2.6.

The software stack of openBIS, built on Java technology, is mostly similar to the other platform exposed in this section. It relies on a PostgreSQL database for the Application Server domain, and a file system with support for segmented and distributed storage for the Data Storage Server domain. Queries that return a large number of rows are optimised to reduce latency during data retrieval. openBIS adopts a classical three-tier architecture with presentation (WebGUI/API), domain (business objects for internal data processing) and data (Data-Access-Object pattern) layers. While the first two layers are public and accessible respectively by humans and machines, the third is responsible for the Create–Retrieve–Update–Delete (CRUD) operations on the database and is kept private. The openBIS group has additionally developed two separate applications for large data transfer. The first one, the CISD File EXchanger (CIFEX), permits web-based data upload of files larger than 2 GiB (that is the current limit for the HTTP protocol), while the other, Datamover, manages secure transfer on unreliable connection and limited storage space. Web services are

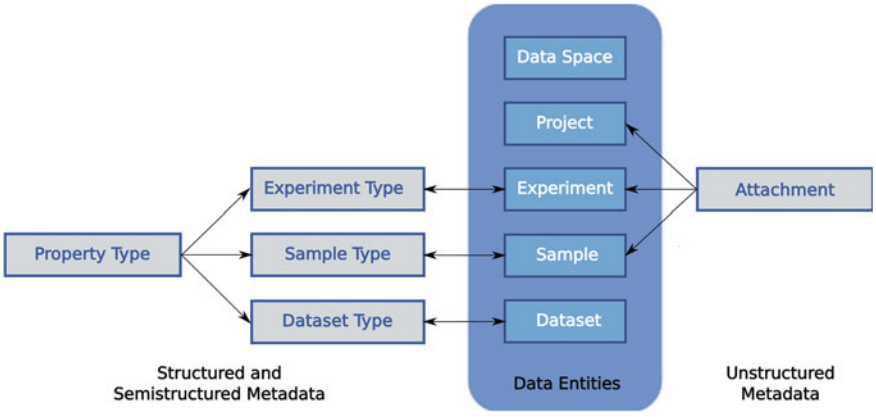

Fig. 2.2 openBIS data model. Data entities are built on a five-level taxonomy. Metadata can be either unstructured—i.e. file attachments—or (semi-)structured. The latter are described by a property type object

provided through a JSON-RPC interface, a lightweight protocol for data transmission across the internet.

2.5.7 i2b2

The Informatics for Integrating Biology and the Bedside (i2b2) platform was designed to provide clinical researchers with the tools required to integrate medical records and biomedical research data in the genomic age [49]. The i2b2 architecture is built on server-side modules, called cells, that communicate each other using web services. A set of connected cells constitute a i2b2 "hive", that can be thus extended according to custom requirements. i2b2 was designed to satisfy at least two requirements (i) find cohorts of patients that can be of interest for subsequent investigations, and (ii) use the information provided by medical records to mine the phenotype of the identified subjects in support of omics or environmental studies. The cohort of patient is selected directly from the institutional databases (e.g. clinical and biobank databases), preserving the privacy of personal information, and is copied in a project-specific data mart. A data mart is the access layer of a data warehouse that is used to expose data to the users. Data marts, like data warehouses, are read-only. i2b2 supports communication with commercial databases (Oracle and Microsoft SQL Server) to extract, transform, and load (ETL) clinical data to the data mart. The user accesses an i2b2 system from the *project management cell*, which handles authentication and authorisation. Using the i2b2 web client the authorised user can access the *data repository cell* and build up from the graphical interface ad hoc queries that are run throughout the institution (or enterprise) databases. The refined ad hoc queries will extract the data set to populate a new data mart. It is possible to select specifically which part of the data must be copied to the data mart, and specify dedicated privileges for other users. Personal data are either kept on a separate repository or encrypted within the data mart.

i2b2 data marts are based on the "star schema" design [50] of data warehousing. The schema consists of four tables:

- **observation_fact**: contains all the observations about a patient. It contains also all the value objects associated with the observation. Each value object must be a basic type (numeric, text, date,…) so composite values are stores in multiple rows. This table can be queried using and EAV approach. The observation_fact refers to the other four tables of the star schema.
- **patient_dimension**: contains the subjects' details, one patient per row.
- **visit_dimension**: describes the periods of time during which the observations were recorded.
- **concept_dimension**: contains controlled vocabulary terms that map the original codes that were used to specify the observation. The i2b2 internal vocabulary allows hierarchical grouping specifying names with a Unix-like directory approach. For instance the `/tumour` group may contain the subgroups

`/tumour/carcinoma` and `/tumour/sarcoma`. This approach allows efficient queries based on pattern matching.
- **observer_dimension**: describes the operator or the mechanical instrument that performed the recording of the observation

Overall, i2b2 collects the data of a biomedical enterprise or institution, usually located in multiple data sources, to be integrated in a small set of tables. The main implementation of i2b2 at Partners HealthCare, as of 2009, contained 1.2 billion observations from 4.6 million subjects, Observations included diagnoses, medications, procedures, and test results (including genomic test results).

i2b2 has been extended and used as a framework to integrate data in various collaborative clinical and research projects, such as the Onco-i2b2 platform [51] and the self-scaling chronic disease registry (i2b2-SSR) [52]. These implementations allow fine grained control over data integrated for sharing purposes.

2.6 Data Repositories Comparison

Confronting all the digital repositories and platforms that I have exposed in the previous section, it is possible to draw comparisons on at least three different grounds:

1. the underlying schema language/format;
2. the adaptability and ease of configurability of the metadata model;
3. the scaling properties of the system.

Concerning point 1, nearly all the metadata schemas that we have seen are structured using XML. XCEDE is written in XML, and so are the data models of XNAT and XTENS; CARMEN and openBIS have elected XML as their format of choice to storage of semi-structured metadata. In clinical data management and biomedical research XML is definitively popular. One of the main reasons for the wide adoption of XML as data-exchange format is that HL7 version 3 messages are composed in XML, and accordingly, HL7 CDA version 2 adopts the same specification. But as explicitly stated in the Release 2 reference article [31], the objective of CDA is to specify "the structure and semantics of a clinical *document* (such as a discharge summary or a progress note) for the purpose of exchange" (emphasis mine). CDA is designed to exchange documents, not data. Various software developers—Douglas Crockford [53], most notably—have pointed out that XML has document-oriented syntax poorly suited for data-oriented objects. While the distinction between the two terms—data versus document—stretches very thin and is much open to debate, from the information technology point of view I can see two requirement that a data-oriented model should satisfy: (i) allow easy mapping to object-oriented languages and (ii) be written in a language that is native in commonly used databases. With the latter I mean that the format should be fast to search and that speed performances should scale-up well with the database dimensions. Both of these statements are not true for XML. Firstly, it was not conceived to be object-oriented and requires some

level of manipulation and mapping, named data binding, to obtain a business object from an XML document tree. Many binding tools have been developed for Java, which is the language adopted by many of the biomedical repositories: the most popular are JAXB, JiBX, and XMLBeans; C++ offers CodeSynthesis XSD. With XML, data must be put within some document structure and this can be complicated with elements that can be nested, attributes that cannot be nested and complex types with their own peculiarities. When trying to develop a data model compliant with modern object-oriented languages, a software engineer should evaluate other solutions besides XML. Secondly, XML is extremely slow to search. Being a text-based format, usually more information (i.e. bits) is required to store it rather than if it were a cell value. There exists some XML-based databases—most notably BaseX, eXist and MarkLogic Server—that are optimised to use languages for XML query and navigation like XQuery and XPath/EXPath [54]. BaseX is the only open-source solution that offers extended language features without recurring to proprietary extensions. In practice though, query performance and scalability is limited, and in many cases XML is stored as text field in a relational database. Such is the case in all the data repositories I have examined. Thirdly, XML is extremely verbose, with both opening and closing tags for each element, and not conceived to be read by humans. Overall, XML works well as an exchange format for transferring data across applications adopting the same data structure, but it is too rigid too allow the flexibility to model the fluid metadata process that could simplify data sharing in research collaborations.

The second point is to some degree related to the first: the more rigid is the adopted language/format, the less adaptable will be the data model. This is the case of XNAT. Existing data types can be extended adding new fields whose values will be stored in a EAV catalogue. The operation of creating a new data type—like a new observational or clinical assessment type—requires first the construction of a new XML document that is likely to daunt the great majority of clinical users. The procedure can be eased with the help of one the many available graphical editors like Liquid XML or Eclipse Vex. Even so, once you have made a new model you have to run an update script, update the database, redeploy the XNAT application and setup XNAT security to allow access to the newly defined data types. All these operations require an administrator of the repository with good computer literacy, and this will likely take the control of the data model away from the scientists. SIMBioMS suffers from similar limitations due to the Hibernate ORM mapping files. COINS and openBIS consent a greater level of extensibility resorting to the EAV paradigm. In openBIS new metadata fields are created a Property Types and be attached to existing entities, while COINS allows the creation of customised clinical assessments to complement the neuroimaging scans. However, it does not support user-configurable fields for all the neuroimaging data types (only for MR and MEG), and no explicit creation of new data types is available to integrate other data sources. While the EAV approach provides a useful tool for defining new metadata fields, it does not explicitly offer a method to construct new data schemas. The structure of the schema must be reconstructed from the associations declared in the database tables. Compared with the other systems, XTENS aims at finding a solution to generate new data types giving full control to the end user (i.e. the scientist). It provides

non-IT users with an experience friendlier than XNAT, removing all the burdensome compilation and re-deploy steps: the user builds up your schema using a graphical web form and once submitted the XML document is automatically generated and ready for use. The administrator has only to configure the permissions to the new data type for the authorised user groups, and this is done from graphical interface as well. If a the new data type has a large set of metadata fields, the construction from web form will be an overlong and tiresome procedure. In this circumstance, it would be necessary to resort to an XML graphical editor. A good approach here is keeping the driving principle of XTENS flexibility, while providing a different, more object-oriented language for composing the metadata schema.

The issue of scalability (third point) is twofold: on the one side we must consider the database and on the other the file system for bulk data storage. Concerning the database, here the main issue is the storage of metadata: besides the adoption of XML for semi-structured metadata, nearly all the systems I have presented either put all the structured metadata in an EAV catalogue or adopt a mixed model, where dedicated tables for widely used data types (e.g. subjects, samples and/or MRI scans) exist along with an EAV representation for all the remaining metadata. COINS, SIMBioMS, and openBIS fall in the first category, while XTENS and XNAT belong to the second. EAV is very attractive because it allows a higher flexibility and requires a small level of database modelling to get it to work: in the simplest implementation, just three tables are required: one for the Entities (that in our case are the data instances), one for the Attributes (the metadata fields), and one for the attribute Values (metadata fields' values and possibly measure units). EAV presents a number of challenges that could hinder with the scalability of the system, in particular a sensitive degradation of performance: if the catalogue grows beyond a certain size, it will reach a point where the efficiency of data retrieval and manipulation will hit a critical low. At that point, the database manager or the application developer has very few choices to solve the issue. It won't be possible to add table indexes, because the table has no specific columns for each attribute as it is the case in a standard relational representation. In general, EAV is less efficient in data retrieval than "conventional" relational schema. As noted in previous studies dedicated to Clinical Databases, attribute-centred searches, where the query criterion is based on the value of a particular attribute, are most likely to show impaired performance [55]. This performance degradation is especially noticeable when query criteria combine one or more simple conditions using boolean operators. The cause for the potential performance degradation is the conversion from the relatively fast AND, OR, and NOT operations that are used when operating on relational schema tables to the sensitively slower set-based equivalents (intersection, union, and difference) for EAV tables. This study, even though fairly ancient, provides us some interesting insight on the limitations of EAV. They found that EAV query performance was three to twelve times slower than the relational schema equivalent, with search speed decreasing as the query complexity increased. The EAV structure consumed approximately for times the size of a conventional schema. As stated by Nadkarni et al. [56] the EAV model is useful for specific scenarios:

1. When dealing with metadata fields that are both numerous and sparse. A metadata field is sparse if it is present only in a small number of data instances. The case for this could be a clinical data repository handling different specialities;
2. When, even if metadata fields are not sparse, the number of different data types is large, and the number of instances for the data types is reasonably small;
3. When dealing with so-called "hybrid" data types: where some fields are sparse and some are not. In this case, even if it might be suitable, the EAV model represents a sub-optimal solution.

As we can see, there are many scenarios that fall outside the three categories outline above. As high-throughput omics technologies become more affordable and extensively used, there will be a flood of metadata—such as the variant calls from a whole genome sequencing analysis—that are not handy to wield and fast to search if stored using EAV approaches. To assess file system scalability there are two main options: (i) a distributed file system managed by a resource broker or a data grid middleware, (ii) taking advantage of a cloud-based storage. While the second is an appealing solution, it not always feasible, especially when the files contain sensitive information and the Institution regulations do not allow storing data in third-party companies' servers.

In conclusion, there is the need for a data model, based on a more flexible, object-oriented and possibly with better performances than the structures that are currently adopted, mostly XML and EAV schemas. The data model should handle in an uniform yet flexible way the wide range of heterogeneous data that are found in the current biomedical research scenario: clinical records, omics and imaging data, and biological specimens. Ideally speaking, from the developer point of view, the model should adopt a format/language that is both natural to the database where it will be stored and, in perspective, to the applications that are going to use it. I have chosen the JavaScript Object Notation (JSON) [57] format as a serious candidate to build the model, because it satisfies all the conditions I have required, and it has become a valid alternative to XML as a data exchange model for many data-driven application on the web.

References

1. Marx, V.: Biology: the big challenges of big data. Nature **498**(7453), 255–260 (2013)
2. Bagley, P.R.: Extension of programming language concepts. Technical report, DTIC Document (1968)
3. Marco, D.: Building and managing the meta data repository. A full lifecycle guide. Wiley, New York (2000)
4. arXiv.org e-print archive. arXiv.org. Accessed 18 Jan 2015
5. Internet archive: digital library of free books, movies, music & wayback machine. https://www.archive.org/ (2015). Accessed 18 Jan 2015
6. Fuller, C.W., Middendorf, L.R., Benner, S.A., Church, G.M., Harris, T., Huang, X., Jovanovich, S.B., Nelson, J.R., Schloss, J.A., Schwartz, D.C., et al.: The challenges of sequencing by synthesis. Nat Biotechnol. **27**(11), 1013–1023 (2009)

7. Merriman, B., Torrent, I., Rothberg, J.M., R & D Team, et al.: Progress in Ion Torrent semiconductor chip based sequencing. Electrophoresis **33**(23), 3397–3417 (2012)
8. Gray, J., Liu, D.T., Nieto-Santisteban, M., Szalay, A., DeWitt, D.J., Heber, G.: Scientific data management in the coming decade. ACM SIGMOD Rec. **34**(4), 34–41 (2005)
9. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin core metadata for resource discovery. Internet Eng. Task Force RFC **2413**(222), 132 (1998)
10. Harper, C.: Dublin core metadata initiative: beyond the element set. Inf. Stand. Q. **22**(1), 20–28 (2010)
11. EPUB Publications 3.0.1. http://www.idpf.org/epub/301/spec/epub-publications.html (2014). Accessed 29 Jan 2015
12. Tolkien, J.R.R.: The Silmarillion. Random House LLC, New York (1979)
13. Dublin Core Metadata Element Set, Version 1.1. http://www.dublincore.org/documents/dces/ (2012). Accessed 29 Jan 2015
14. Brazma, A., Hingamp, P., Quackenbush, J., Sherlock, G., Spellman, P., Stoeckert, C., Aach, J., Ansorge, W., Ball, C.A., Causton, H.C., et al.: Minimum information about a microarray experiment (MIAME) toward standards for microarray data. Nat. Genet. **29**(4), 365–371 (2001)
15. Rayner, T.F., Rocca-Serra, P., Spellman, P.T., Causton, H.C., Farne, A., Holloway, E., Irizarry, R.A., Liu, J., Maier, D.S., Miller, M., et al.: A simple spreadsheet-based, MIAME-supportive format for microarray data: MAGE-TAB. BMC Bioinform. **7**(1), 489 (2006)
16. Spellman, P.T., Miller, M., Stewart, J., Troup, C., Sarkans, U., Chervitz, S., Bernhart, D., Sherlock, G., Ball, C., Lepage, M., et al.: Design and implementation of microarray gene expression markup language (MAGE-ML). Genome Biol. **3**(3), research0046 (2002)
17. Bustin, S.A., Benes, V., Garson, J.A., Hellemans, J., Huggett, J., Kubista, M., Mueller, R., Nolan, T., Pfaffl, M.W., Shipley, G.L., et al.: The MIQE guidelines: minimum information for publication of quantitative real-time PCR experiments. Clin. Chem. **55**(4), 611–622 (2009)
18. Poldrack, R.A., Fletcher, P.C., Henson, R.N., Worsley, K.J., Brett, M., Nichols, T.E.: Guidelines for reporting an fMRI study. Neuroimage **40**(2), 409–414 (2008)
19. Taylor, C.F., Field, D., Sansone, S.-A., Aerts, J., Apweiler, R., Ashburner, M., Ball, C.A., Binz, P.-A., Bogue, M., Booth, T., et al.: Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project. Nat. Biotechnol. **26**(8), 889–896 (2008)
20. Norlin, L., Fransson, M.N., Eriksson, M., Merino-Martinez, R., Anderberg, M., Kurtovic, S., Litton, J.-E.: A minimum data set for sharing biobank samples, information, and data: MIABIS. Biopreservation and biobanking **10**(4), 343–348 (2012)
21. Kolker, E., Özdemir, V., Martens, L., Hancock, W., Anderson, G., Anderson, N., Aynacioglu, S., Baranova, A., Campagna, S.R., Chen, R., et al.: Toward more transparent and reproducible omics studies through a common metadata checklist and data publications. OMICS: J. Integr. Biol. **18**(1), 10–14 (2014)
22. Brazma, A.: Minimum information about a microarray experiment (MIAME)-successes, failures, challenges. Sci. World J. **9**, 420–423 (2009)
23. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., et al.: The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. Nat. Biotechnol. **25**(11), 1251–1255 (2007)
24. Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A.L., Rosse, C.: Relations in biomedical ontologies. Genome biology **6**(5), R46 (2005)
25. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.-A., Chute, C.G., et al.: Bioportal: ontologies and integrated data resources at the click of a mouse. Nucleic acids Res. **37**(suppl 2), W170–W173 (2009)
26. Salvadores, M., Alexander, P.A., Musen, M.A., Noy, N.F.: Bioportal as a dataset of linked biomedical ontologies and terminologies in RDF. Semantic Web **4**(3), 277–284 (2013)
27. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Sci. Am. **284**(5), 28–37 (2001)

28. Dürst, M., Suignard, M.: Internationalized resource identifiers (IRIs). Technical report, RFC 3987 (2005)
29. Klyne, G., Carroll J.J.: Resource description framework (RDF): concepts and abstract syntax (2006)
30. Sporny, M.: JSON-LD and why I hate the semantic web. http://manu.sporny.org/2014/json-ld-origins-2/ (2014). Accessed 19 Jan 2015
31. Dolin, R.H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F.M., Biron, P.V., Shabo Shvo, A.: HL7 clinical document architecture, release 2. J. Am. Med. Inform. Assoc. **13**(1), 30–39 (2006)
32. Operational data model. http://www.cdisc.org/odm (2015). Accessed 20 Jan 2015
33. Gadde, S., Aucoin, N., Grethe, J.S., Keator, D.B., Marcus, D.S., Pieper, S.: XCEDE: an extensible schema for biomedical data. Neuroinformatics **10**(1), 19–32 (2012)
34. Edwards, P., Mayernik, M.S., Batcheller, A., Bowker, G., Borgman, C.: Science friction: data, metadata, and collaboration. Soc. Stud. Sci. **41**, 667–690 (2011). doi:10.1177/0306312711413314
35. Neu, S.C., Crawford, K.L., Toga, A.W.: Practical management of heterogeneous neuroimaging metadata by global neuroimaging data repositories. Front. Neuroinform. **6**, 8 (2012)
36. Marcus, D.S., Olsen, T.R., Ramaratnam, M., Buckner, R.L.: The extensible neuroimaging archive toolkit. Neuroinformatics **5**(1), 11–33 (2007)
37. XML schema. http://www.w3.org/XML/Schema (2014). Accessed 20 Jan 2015
38. Herrick, R., McKay, M., Olsen, T., Horton, W., Florida, M., Moore, C.J., Marcus, D.S.: Data dictionary services in XNAT and the human connectome project. Front. Neuroinform. **8**, 65 (2014)
39. Scott, A., Courtney, W., Wood, D., De la Garza, R., Lane, S., King, M., Wang, R., Roberts, J., Turner, J.A., Calhoun, J.D.: COINS: an innovative informatics and neuroimaging tool suite built for large heterogeneous datasets. Front. Neuroinform. **5**, 33 (2011)
40. COINS - central authentication system. https://www.coins.mrn.org/dx (2015). Accessed 05 Feb 2015
41. Austin, J., Jackson, T., Fletcher, M., Jessop, M., Liang, B., Weeks, M., Smith, L., Ingram, C., Watson, P.: CARMEN: code analysis, repository and modeling for e-neuroscience. Procedia Comput. Sci. **4**, 768–777 (2011)
42. Corradi, L., Arnulfo, G., Schenone, A., Porro, I., Fato, M.: XTENS - an extensible environment for neuroscience. Stud. Health Technol. Inform. **147**, 127 (2009)
43. Corradi, L., Porro, I., Schenone, A., Momeni, P., Ferrari, R., Nobili, F., Ferrara, M., Arnulfo, G., Fato, M.M.: A repository based on a dynamically extensible data model supporting multidisciplinary research in neuroscience. BMC Med. Inform. Decis. Mak. **12**(1), 115 (2012)
44. XSL Transformation. Version 1.0, W3C recommendation 16 November 1999
45. Krestyaninova, M., Zarins, A., Viksna, J., Kurbatova, N., Rucevskis, P., Neogi, S.G., Gostev, M., Perheentupa, T., Knuuttila, J., Barrett, A., et al.: A system for information management in biomedical studies SIMBioMS. Bioinform. **25**(20), 2768–2769 (2009)
46. Bauer, C., King, G.: Hibernate in Action. Manning, GreenWich (2005)
47. SIMS configuration guide. http://www.simbioms.org/wordpress/wp-content/uploads/2013/08/sims_configuration_guide_02.14.pdf (2013). Accessed 20 Jan 2015
48. Bauch, A., Adamczyk, I., Buczek, P., Elmer, F.-J., Enimanev, K., Glyzewski, P., Kohler, M., Pylak, T., Quandt, A., Ramakrishnan, C., et al.: openBIS: a flexible framework for managing and analyzing complex data in biology research. BMC Bioinform. **12**(1), 468 (2011)
49. Murphy, S.N., Weber, G., Mendis, M., Gainer, V., Chueh, H.C., Churchill, S., Kohane, I.: Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). J. Am. Med. Inform. Assoc. **17**(2), 124–130 (2010)
50. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. Wiley, New York (2011)
51. Segagni, D., Tibollo, V., Dagliati, A., Zambelli, A., Priori, S.G., Bellazzi, R.: An ICT infrastructure to integrate clinical and molecular data in oncology research. BMC Bioinform. **13**(Suppl 4), S5 (2012)

52. Natter, M.D., Quan, J., Ortiz, D.M., Bousvaros, A., Ilowite, N.T., Inman, C.J., Marsolo, K., McMurry, A.J., Sandborg, C.I., Schanberg, L.E., et al.: An i2b2-based, generalizable, open source, self-scaling chronic disease registry. J. Am. Med. Inform. Assoc. **20**(1), 172–179 (2013)
53. Crockford, D.: JSON: The fat-free alternative to XML. Proc. of XML **2006** (2006)
54. Iacob, E.: The extended XPath language (EXPath) for querying concurrent markup hierarchies. http://dblab.csr.uky.edu/~eiaco0/docs/expath (2005)
55. Chen, R.S., Nadkarni, P., Marenco, L., Levin, F., Erdos, J., Miller, P.L.: Exploring performance issues for a clinical database organized using an entity-attribute-value representation. J. Am. Med. Inf. Assoc. **7**(5), 475–487 (2000)
56. Dinu, V., Nadkarni, P.: Guidelines for the effective use of entity-attribute-value modeling for biomedical databases. Int. J. Med. Inform. **76**(11), 769–779 (2007)
57. JSON. http://json.org/ (2015). Accessed 23 Jan 2015

Chapter 3
The JSON-Based Data Model

3.1 Introduction

As stated in Chap. 2, the goal of the JSON-based data model is to overcome the issues and limitations evidenced by the literature. The model must provide a uniform interface and structure to handle heterogeneous information, such as patient health records, details about biological specimens, high throughput genomic outputs, and so on. However, some data instances require additional operations that must be dealt in a specific way. For instance, subjects may have sensitive information attached that the system has to manage taking into account privacy issues and regulations. The user must be able to associate samples (i.e. biological specimens) with a specific biobank and, if required, locate it within a freezer. Therefore, I decided to proceed as follows. First, I defined a generic (i.e. more abstracted) Data class, with a set of common operations that all the data instances must implement. Afterwards I defined two specialisations of the Data class, to describe subjects (such as human patients, but possibly also animals or cell lines) and samples. In the following section I will describe the generic Data entity.

3.2 Why JSON?

JSON is a text format for the serialisation of structured and semi-structured data. It is derived from JavaScript object literals, hence the name. In practice, it is a minimal and portable textual subset of JavaScript [1]. It has gained a huge popularity as the preferred data exchange format on the web, given the ubiquitous nature of JavaScript language, present and extensively employed in all the major web browsers.[1] JSON

[1]For the sake of simplicity from now on, whenever I speak about the "major browsers" I refer to these five guys: Internet Explorer, Mozilla Firefox, Google Chrome, Opera and Safari.

M. lzzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5_3

```
{
   "astring": "some text",
   "anumber": 10,
   "an object": {
      "obj property": "dunno",
      "an array": [1, "two", 3.33, null, true],
      "a float": 9.42
   },
   "human-friendly": true,
   "xml": false,
   "anullvalue": null,
   "georgian": "ფროდო ბეგინსი",
   "datetime-ISO8601": "2014-01-01T23:28:56.782Z",
   "email": "someemail@adomain.com",
   "uri": "http://www.tvtropes.org/"
}
```

Fig. 3.1 An example of a JSON data object. The three fundamental types are shown: strings (in *blue*), numbers (*orange*), booleans (*green*) and *null* (*grey*). Other types—such as date, timestamps, and emails are represented as formatted stings. Notice that array are allowed to contain heterogeneous values

provides a paradigm to store data and metadata in an organised yet flexible and easy-to access way. I find surprising that so far there have been no academic effort to use it as a format to store heterogeneous scientific metadata. An example of JSON is provided in Fig. 3.1. A JSON object is enclosed within curly brackets. The object is composed by a set of *properties* expressed as key—value pairs. Any recognised Unicode string is a valid key. JSON support four primitive values—string, number, boolean and *null*—and two structured subtypes: objects and arrays. It is therefore possible to nest objects within other object, building up a hierarchical structure. Arrays are ordered lists of objects or primitive types and are enclosed within square brackets. There exist various comparisons between JSON and XML available on the internet, so I am not going to repeat most of the arguments that have already been raised by more expert developers. I limit myself to two key considerations relevant to my design goals:

- JSON is an object-oriented model. Even though its syntax has been extrapolated from JavaScript its textual nature makes it language-independent. It does not required data binding libraries to be parsed in C-like languages like C++, Java and C#. In Python, another language widely used by the scientific community, support is provided by the standard libraries. In MATLAB there are various options available, the most popular being JSONlab.
- JSON natively supports ordered lists as arrays. Lists are what programmers use to manage collections of objects. Document-oriented formats like XML (and RDF) do not provide any internal construct to model lists. Arrays in XML have to be

expressed by conventions: for instance using an outer placeholder element that contains the arrays items as inner elements. The outer container is named by convention with the plural form of the inner element name.
- JSON does not provide an equivalent to the XML <[CDATA[]]> feature to store images, audio/video or binary payloads, but this is not an issue for my data model. JSON will be used to define the data schema and store metadata, while all the bulk files will be stored separately on a distributed file system. This approach will easy scaling-up the application to bigger data workloads.

There are other data exchange models that have proven to be very flexible and provide impressive performances on transmissions across computer networks. There are other data-exchange formats, noticeably MessagePack [2] and ProtocolBuffers [3], developed by Google. MessagePack is a binary format with a structure similar to JSON, but adopts strategies to store short integers and small strings in fewer bytes. The ProtocolBuffers format allows the user to define a data schema, then a compiler generates code for reading and writing the data. Binary data-interchange formats generally outperform self-describing (i.e. text-based) ones like JSON and XML in data serialisation [4] and transmission. On the downside they do not offer a human readable data representation and are not as widely supported. Emeakaroha *et al.* suggest a hybrid approach that combines the strengths of binary and text-based data format to enhance efficient communication and interoperability in Grid– and Cloud–based applications [5]. I have decided to adopt a text-based solution, keeping in mind that the end-user friendliness is one of the main objectives to provide the user with the better experience. Moreover, MessagePack and ProtocolBuffers are not as popular as JSON or XML, and I did not want to build up a stand-alone system who could not interoperate with the services and analysis platforms available online.

3.3 The JSON Metadata Schema

As previously stated in Sect. 2.1, the broadest definition of metadata is "description of physical or digital things". These "things", from the data management point of view are mapped to entities in a database—or, more generally, in a data store—and are what from now on I will call data instances, data entities, or simply data. This abstracted data entity lies at the core of the JSON data model. In biomedical science, a data instance may refer to a subject, a sample, or any kind of analysis and processing steps that may produced in a research project. Regardless of its nature, each data instance will have a set of metadata attributes that describes it. Different types of data will have a different metadata structure, that is a different set of metadata attributes (or fields[2]). It is the data "type" that uniquely specifies the metadata structure. The `DataType` class has only two properties: a name, an alphanumeric string that uniquely identifies the type, and a schema that is a JSON object that contains all the metadata structure

[2]The two words are used interchangeably in the text.

for the type. The schema contains the metadata structure or model. It is composed by two parts:

- a **header** containing the schema name, a brief description, a version number, a boolean field stating whether the data type has associated files, an optional reference to an ontology or terminology.
- a **body** containing all the metadata fields. Metadata fields are defined in a constrained hierarchical structure.

A metadata body contains one or more metadata groups. I introduced the concept of metadata groups to allow the division of metadata according to their nature or origin. This was inspired by the categorisation suggested by Nadkarni [6] in technical and descriptive metadata. The separation between the two categories is blurred, so it is to the user to divide its metadata in as many groups as she/he sees necessary. For instance, users may want to identify these metadata groups: system metadata, technical metadata, on the reproducibility of the experiment, metadata on the operator performing the task/analysis, and descriptive metadata retrieved from a file they are saving. Metadata groups are allowed to contain two different elements: single metadata fields and/or metadata loops. The metadata field represents the leaf of the model in its current implementation. A field cannot contain other subfields. The metadata field is a JSON object with a set of properties that constitute the so called structural and validation metadata. Here it follows the list of properties:

- **label**—it is the tag used by the software to recognise the object a metadata field. Its value is constrained to the string "METADATA_FIELD".
- **type**—the primitive type of the field. Currently the system supports five primitive types: Text, Float, Integer, Date and Boolean.
- **customValue** (optional)—a default value to populate the field when no value is provided by the user (optional).
- **iri** (optional)—an IRI that links the metadata field to some univocally identified resource on the web. It can be used to store a term from a controlled vocabulary or an ontology.
- **required**—boolean flag; if true the user must provide a value for the field.
- **sensitive**—boolean flag; if true the field will be removed before sending it to users without access rights to personal and/or sensitive information.
- **isList**—boolean flag; if true the user will select the value from a list of terms.
- **possibleValues**—the list of options that are allowed for the field. Used only if isList is set to true.
- **hasTableConnection**—boolean flag; if true, the value for the current field will be selected from a list or a controlled vocabulary provided from some external resource (a database table, a controlled vocabulary available online…).
- **tableConnection**—the Uniform Resource Identifier (URI) of the resource from where the external controlled vocabulary can be retrieved; used only if *hasTableConnection* is set to true.
- **hasUnit**—boolean flag; it states whether the field has a measure unit, such as it may be if it were a physical measure.

- **possibleUnits**—if *hasUnit* is true, this property provides a list of units. For consistency's sake a single unit value is advisable. However the model supports the adoption of multiple units for the same field. This might be useful, for instance, when a time lapse can be measured in seconds, days or years. In the current implementation there is no support for unit conversion, that could be achieved using an ontology such as the Ontology of Units of Measure and Related Concepts [7].
- **isRange**—boolean flag, currently used only for numerical fields. If it is set to true, the values allowed must fall within a range, that can be constrained by three properties: **min**, **max**, and **step**. The user can specify only one between *min* and *max*, while *step* is used only if *min* is provided. The allowed values will therefore be $val = min + N \times step$, with N belonging to the natural numbers, as long as $val <= max$.

This is the list of properties currently supported by the model. Given the flexible nature of JSON—no validation is required against an XSD—it is easy to extend this list to provide further structural, validation and guide metadata. One that I plan to provide soon, is a **notes** field, to supply the user with a brief but essential explanation of each metadata field, to support the most correct value section. Given the dynamical approach of the model, guide metadata such as notes will be easy to edit by the end user, and will positively reduce the "metadata friction", as exposed in Sect. 2.4. I have included the concept of metadata loops—mutuated from the XTENS original data model of Corradi [8]—to describe recurring fields or ensembles of recurring fields. A loop represents a collection of recursive fields (i.e. attributes). It is useful to store recurrent metadata fields occurring an unforeseeable number of times. Two examples of relevance for clinical research are the field 'metastatic site' for a data type associated to a 'relapse' clinical event, or the 'overexpressed gene' field for a 'DNA microarray' data type. Both of them may occur more than once in a specific data instance, but the number of occurrences varies from time to time. The metadata loop object contains three properties: **name**, an optional **iri**, and **content**, an array containing its fields. The metadata group has the same set of properties of the metadata loop, but the *content* array can store both fields and loops in any order.

3.4 A First Implementation: XTENS 1.5

Given the need to test the applicability of the JSON data model, I first decided to incorporate it as the core of an existing platform. The reason was one of time: designing, developing, and testing an entirely new data repository was an effort too huge at the beginning, since I was still quite inexperienced in web programming and data management. Moreover it might have proved a worthless risk, if in the end the model did not work as I hoped. Therefore I decided to use the XTENS repository as the exoskeleton for a JSON-based data management platform. The choice of XTENS was driven by its capability to extend the data model (i.e. create new data types) online, without having to reconfigure, rebuild and restart the system. Furthermore, it already provided dedicated functionalities for sample management, a feature that

was required for tackling properly the integrated biobanking setup. I redesigned the XTENS object model in order to incorporate in it the JSON metadata schema. I have distinguished the object model in two separate domains, as it can be seen in Fig. 3.2:

- The *operational model*, a traditional object-oriented (OO) class model where all the classes that map operational entities of the biomedical domain are defined, together with their associations (see Fig. 3.3). A subset of these classes—those shown in Fig. 3.2, namely `Patient`, `Sample`, `Process`, `Event`, `Data` and `MdataValue` —are typed classes. All typed classes extend the abstract `TypedElement` class;
- The *meta model*, an OO class model that contains all the type definition classes. All the type definition classes implement the abstract `Type` class;

Therefore, each typed class instance is univocally defined by its type. The concepts of meta and operational models are borrowed, though applied in a different context, from the work of Bush and Wedemann [9]. In the operational model, the `Project` is—similarly to XCEDE—a macro group where all kind of patients' data and information are collected. Each project contains one or more `Patient` entities. In a generic biomedical scenario, when a patient is enrolled in a project, she/he may enter in a research study, composed by a (maybe flexible) set of analysis steps. The process-event paradigm of XTENS as described in Sect. 2.5.4 uses the two entities `Process` and `Event` to model these workflows. To briefly recap the concept, a process is a collection of sequential events and/or sub-processes related to an activity, allowing the design of a multi-layered hierarchical structure. The entity `PatientCollection` is an aggregate of `Patient`, employed to decouple patients from processes and manage possible situations where a process (e.g. study or visit) contains analyses that require merging data coming from more than one patient. `Process` and `Event` objects are fully characterised by the corresponding `ProcessType` and `EventType` instances as defined in the meta model. The same patient may be involved in more studies, such as a gene expression profiling and/or a clinical trial. In the process-event model, every time a patient enters a study a new process is created and activated. A study is composed by a sequence of events, and each of them may produce a `Data` instance, fully described by its `DataType`. In the XTENS repository authorised users can create and activate on the fly new process, event and data types, without the need to recompile the application. A form-based graphical interface (containing all the fields that map to properties in the metadata JSON schema) allows users/operators to manually define new data types, adding metadata groups, attributes and loops. When creating a novel data type, the users can select an ontology from a list, if they want to name the metadata fields using controlled and semantically associated terms. The ontology is downloaded directly online from the Bioportal repository. The selected ontology will be loaded and the application will suggest the terms to the operator using an auto-complete widget. All the parameters and properties specified in the schema as described in Sect. 3.3, can be set from the form-based interface. I have developed a client-side JavaScript routine that scans all its fields when the form is submitted, builds the JSON schema and sends it to the XTENS server. The server stores the newly created data type as

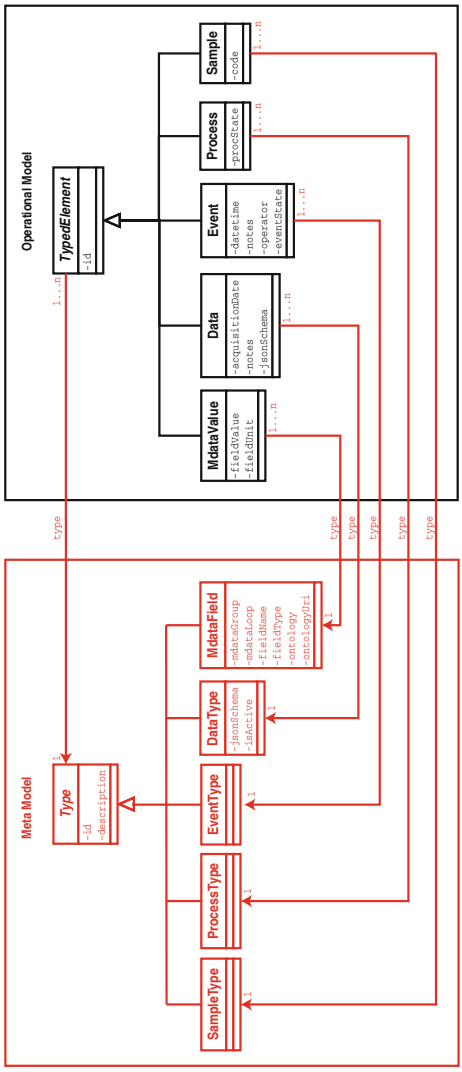

Fig. 3.2 Details of the class model, showing the operational model (*right*, in *black*), which contains all classes involved in the biomedical workflow and the meta model (*left*, in *red*) representing the types. All typed classes in the operational model inherit from `TypedElement`, while all types in the meta model inherit from Type. The property *jsonSchema* of `DataType` contains the metadata schema template as JSON for the specific data type. `MdataField` properties contain info about the metadata group and (optionally) loop that the attribute belongs to, its name and type, and an optional link to a terminology. The property *jsonSchema* of Data contains the metadata schema populated with values (and units) selected by the operator when the data instance was registered in the system. `MdataValues` represents a single metadata field instance, and has properties for value and unit. Event and Process both have a property (*eventState* and *procState*) to check whether they are still active, terminated or paused. The full list of Sample properties is not shown in the picture. Each class in the diagram realises an XTENS database table

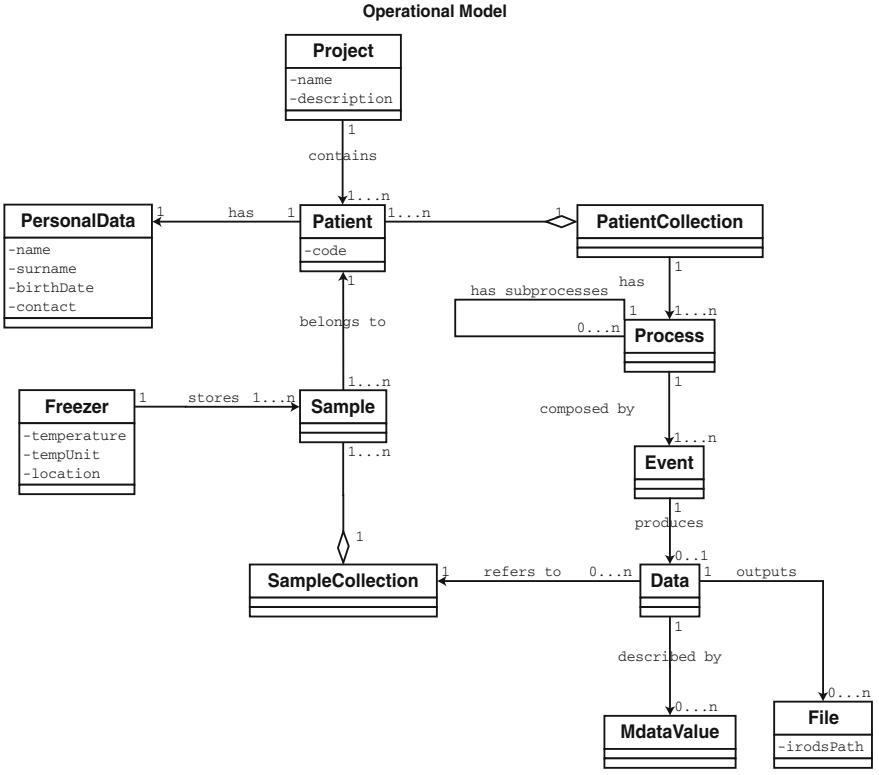

Fig. 3.3 The diagram shows the main entities of the biomedical operative model. The `Project` class represents the top-level component of our model. A `PersonalData` class contains all the sensitive information about a patient. Only authorised operators can access the identifier that maps `Patient` objects to `PersonalData`. The details of the freezer management system, that had already been implemented in XTENS by Corradi, are not shown. A process contains sub-processes and/or events, outlining a flexible hierarchical structure. Each Data object maps to a single Event and is described by a set of MdataValues objects. The property *irodsPath* of File contains the logical path of the document in the iRODS file system

a `DataType` instance in the XTENS database. The metadata schema is stored in the property *jsonSchema* of `DataType`. Additionally, XTENS stores each attribute definition as a type in the `MdataField` table, one of the classes outlined of the meta-model as shown in Fig. 3.2. Once the data type is activated for a user group, users belonging to the group can select it to save its data instances. The associated event is first created and inserted in the appropriate process when a user wants to register a new data instance in the repository. Then the metadata schema from the selected data type is retrieved from the database, is parsed, and dynamically converted to a web form using jQuery.dForm [10], a jQuery plugin. The metadata schema, populated with the values selected by the user, is stored in the property *jsonSchema* of the `Data` entity (see Fig. 3.2). On submission each metadata attribute is parsed, and

its value and measure unit are saved in the `MdataValue` table. The three tables `Data`, `MdataField` and `MdataValue` are respectively the Entity, Attribute and Value tables of the XTENS EAV catalogue. One might wonder why, after talking so much about the scalability issues of the EAV model, I decided to adopt it in this first implementation. It was necessary because, to incorporate the JSON schema within XTENS, I had to adopt a mixed model for metadata storage and management. In this mixed model, metadata from patients and samples are stored as columns in dedicated tables, while all the other metadata are stored both as a single JSON document. MySQL, the Relational Database Management System (RDBMS) used by XTENS, does not provide a primitive type to store JSON document, so the JSON schemas are stored as textual fields, with serious impairment for query speed. To overcome this issue, each metadata field value is also stored separately in an EAV catalogue.

If the *fileUpload* option is set to true in the *jsonSchema* header, one or more files can be uploaded by the user and registered in the in the XTENS data grid system managed by the iRODS middleware. A new file collection is created on iRODS, all the uploaded files are stored within it and the metadata are stored as attribute-value-unit (AVU) triples on the iRODS metadata catalogue (iCAT) and associated to each file in the collection. XTENS also saves the JSON schema as a text file in the same iRODS collection where all related data files are saved. This way, in a virtual community scenario involving many institutions, both files and the metadata description could be replicated and shared among all the centres deploying an iRODS server. The operations of data type creation and subsequent data management have been described in my publication on *BMC Genomics* [11]. In 2013 I recorded a video, that is available online, detailing the two procedures [12].

In selected cases, new data insertion may require additional operations and modifications inside the database. I have developed an abstract server side Java class loosely based on the Command design pattern [13] to handle these situations. The abstract class contains three methods: *check*, *retrieve* and *recovery*. *Check* is executed before inserting the new data, to verify whether all the required conditions (for data submission) are met and/or satisfied. The check method may also be used any time an event—and the related data entity—depends on previous events (as stated by a protocol workflow or pipeline), to verify that all previous required events have been registered in XTENS. *Execute* is called immediately after the new data insertion to apply all the additional modifications to database entities. It may also contain a procedure to automatically populate a metadata instance parsing a file header using, if necessary, an appropriate ETL procedure, without need for the user to load them manually using the web form. *Recovery* is run if the *execute* step fails for any reason, in order to fall back to the original configuration. This abstract Command class can be extended to handle different situations. I will show an implementation of it to manage sample aliquot deliveries in the next chapter. Finally, to support the metadata mixed-model, I newly designed a flexible search interface that allows users to compose queries based on the custom-defined metadata attributes and run them on the database and on the grid, to recover patient and sample information, and files. In the following chapter, I illustrate the first practical application of it, in an integrated biobanking scenario.

References

1. Crockford, D.: The Application/json Media type for Javascript Object Notation (JSON) (2006)
2. MessagePack. http://msgpack.org/ (2015). Accessed 23 Jan 2015
3. Kaur, G., Fuad, M.M.: An evaluation of protocol buffer. In: Proceedings of the IEEE SoutheastCon (SoutheastCon), pp. 459–462. IEEE (2010)
4. Maeda, K.: Performance evaluation of object serialization libraries in XML, JSON and binary formats. In: Proceedings of 2nd International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), pp. 177–182. IEEE (2012)
5. Emeakaroha, V.C., Healy, P., Fatema, K., Morrison, J.P.: Analysis of data interchange formats for interoperable and efficient data communication in clouds. In: Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, pp. 393–398. IEEE Computer Society (2013)
6. Nadkarni, P.M.: Metadata-driven Software Systems in Biomedicine: Designing Systems that Can Adapt to Changing Knowledge. Springer, Berlin (2011)
7. Rijgersberg, H., van Assem, M., Top, J.: Ontology of units of measure and related concepts. Sem Web **4**(1), 3–13 (2013)
8. Corradi, L., Porro, I., Schenone, A., Momeni, P., Ferrari, R., Nobili, F., Ferrara, M., Arnulfo, G., Fato, M.M.: A repository based on a dynamically extensible data model supporting multidisciplinary research in neuroscience. BMC Med. Inf. Decis. Mak. **12**(1), 115 (2012)
9. Busch, N., Wedemann, G.: Modeling genomic data with type attributes, balancing stability and maintainability. BMC Bioinf. **10**(1), 97 (2009)
10. JQuery.dform. https://github.com/daffl/jquery.dform (2015). Accessed 25 Jan 2015
11. Izzo, M., Mortola, F., Arnulfo, G., Fato, M.M., Varesio, L.: A digital repository with an extensible data model for biobanking and genomic analysis management. BMC genomics **15**(Suppl 3), S3 (2014)
12. Izzo, M.: XTENS-biobank: new data type creation. https://youtu.be/b9JXaCUnK5Y (2013). Accessed 31 March 2015
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education, Upper Saddle River (1994)

Chapter 4
Results: The Integrated Biobanking Use Case

The new version of XTENS equipped with the JSON metadata schema, named XTENS 1.5, has first been adopted at the BiolMol of the Giannina Gaslini Institute (IGG) of Genoa. BiolMol has an ongoing collaboration with DIBRIS for data management and analysis. IGG is a Paediatric Hospital with a long tradition and experience in developmental-related diseases. BiolMol, together with the Anatomical Pathology unit, manages Gaslini's main biobank, the Biobank Integrating Tissue-omics (BIT).

4.1 BIT Workflow

BIT collects tissue and blood samples of paediatric patients, with a main focus on neuroblastic tumours. BIT centralises samples of neuroblastic tumours—in particular neuroblastoma—from all over Italy. The biobank was founded in 2009 and, as of March 2015, more than 2400 different samples (see Table 4.1) have been collected and stored. Currently BIT has histopathological and genomic characterisation of the samples including structural alterations in DNA (CGH array) and gene expression profiles (Affymetrix DNA Microarray) of about 150 neuroblastoma tumours. Every six months, the patients' updated clinical history is provided from the Italian Neuroblastoma Registry [1]. All this information—samples, clinical, genomic and personal data—must be integrated and stored inside the biobank database. The use-case diagram of the activity related to BIT are shown in Fig. 4.1. Every time a new sample arrives—it is usually sent from the Anatomical Pathology facilities (if it is a tumour), from the Central Laboratories (if it is a whole blood sample) or from another hospital—the biologist stores it in the biobank and registers it in the system. Each clinical sample must be associated to a patient. By convention, the patient is univocally identified by name, surname and birth date, which are the only personal informations available to the biobank. For any banked sample, the pathologist provides the histopathological diagnosis and other pertinent information such as the

M. lzzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5_4

Table 4.1 Summary of patients, samples (by tipe), and clinical/molecular data stored in the BIT digital repository as of 25th March 2015

	Total	Neuroblastoma
Subjects	1241	758
Tissues	1881	1065
Fluids	519	320
DNA	916	819
RNA	475	400
Microarray	175	172
aCGH	155	155
ALK mutation (Sanger)	30	30
Clinical and molecular details	680	680

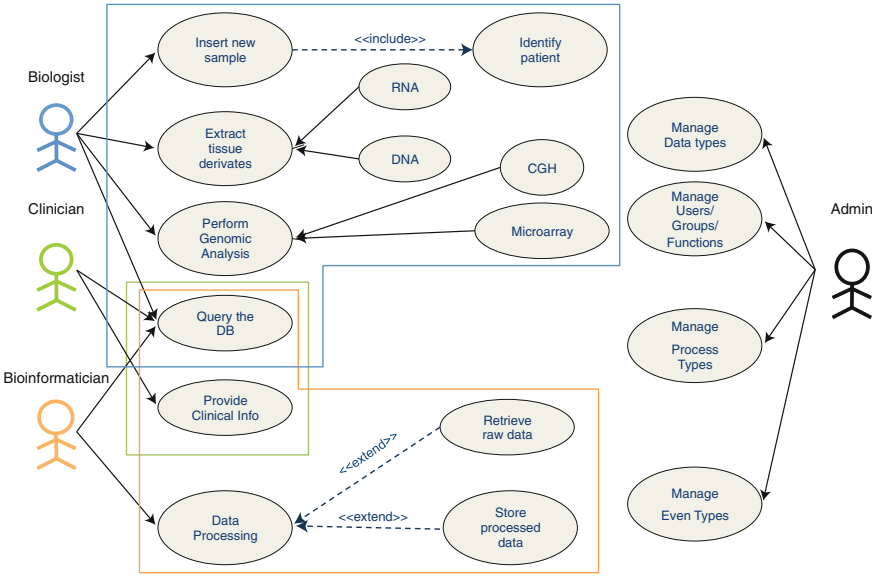

Fig. 4.1 Use case diagram for the biobanking activity at BIT-Gaslini. I have identified four actors in the biobank management system. The biologist banks tissue and blood samples, extracts RNA and DNA from them and performs structural and expression genomic analyses. The clinician periodically provides clinical data about patients. The bioinformatician retrieves the collected information and processes it using classifiers and machine learning tools. The repository administrator manages users, groups and functions. She/he creates and activates functions and data types for specific groups. In a small lab or group another actor (e.g. biologist) may also have administrative role

percentage of tumour cells. This information may arrive after some days (less than a week), depending on the pathologist's schedule. Afterwards, if the sample meets the requirements of specified by the clinical protocols requiring genomic investigations, the biologist extracts nucleic acid derivatives. Multiple extractions from the same

tissue can be performed, especially if the derivative quantity is not sufficient for all the genomic analyses, or the quality is poor. The tissue and its derivatives are stored and preserved in the biobank for further use.

4.2 XTENS for BIT

One limitation of XTENS 1.0 was that it did not support associations between samples, such as the hierarchical one-to-many association existing between a "parent" sample (e.g. a tissue) and its "children" samples (DNA and RNA, for instance, but also plasma and serum obtained from whole blood fit in this category). Therefore I have newly designed the sample management system as it follows. A patient can have one or more samples associated to it; I have introduced the typed class `Sample` in XTENS operational model and the associated type `SampleType` in the meta model, as it was shown in Fig. 3.2. I performed a further modification to the data management policy enabling data instances association to samples as well as patients. This is crucial in the BIT use case, where researchers perform the same functional genomic analysis—let's say a microarray or an NGS assay—on DNA or RNA extracted from two different samples (e.g. lymphocytes and tumour tissue) belonging to the same patient, to compare the genomic profile of sane and diseased cells. In such a situation, they must be able to associate each data instance to a specific sample. I introduced the entity `SampleCollection` as an aggregate of `Sample` to decouple the mapping from `Sample` to `Data` and handle also data instances that merge information coming from multiple samples (see Fig. 3.3). It has a role analogous to `PatientCollection` for `Patient`. In this way, patient has one or more samples, and in addition each tissue or blood sample may generate "children" samples as in the case of genomic derivatives. Use of a foreign key pointing to the "parent sample" in the Sample entity allows to track each final product or aliquot to the master sample. I have separated personal sensitive information from the remaining patient data into two different entities (i.e. database tables) to guarantee pseudononymisation. Only authorised users may access the unique identifier that allows retracing and retrieving personal information and link it to clinical data and samples. Each molecular analysis or bioinformatical processing step performed on biobank-related samples or data is associated to a new data instance registered in the repository. Using the process-event schema, I identified three main process types for biobanking management at BIT-Gaslini: *patient management*, *sample management* and *genomic analysis*. Through patient management it is possible to track patient creation, modification, and deletion, and any periodical insertion and update to the Clinical Data provided by Physicians. Sample Management comprises, besides sample insertion and update, also aliquot deliveries of tissue derivatives to other labs or institutions outside IGG for specific genomic analyses or research collaborations. When the system was first set up and installed, two genomic analyses were performed at IGG: CGH Array and cDNA Microarray. CGH stands for Comparative Genomic Hybridisation, and is a molecular cytogenetic technique for studying copy number variations (CNV)

relative to ploidy level (i.e. number of sets of chromosomes in the nucleus of a cell) in a test sample compared to a control [2]. In other words, it allows to detect losses and gains in DNA copy number across the entire genome, without prior information. The comparison is executed between a DNA sample of tumoural origin (our test) and a reference DNA. For the European Low and Intermediate Risk Neuroblastoma Protocol (LINES) and, to a lesser extent, the High Risk Neuroblastoma Treatment, there are three significant outcomes:

- Segmental Chromosome Abnormality (SCA): if the structure of one or more chromosomes is altered. SCA comprises: deletions, gains, amplifications, and more complicated mutations like translocations, inversions, and insertions. In neuroblastoma the presence of SCA is associated with a higher risk of relapse [3]. Since high-risk neuroblastoma demonstrates SCA in the great majority of cases, this abnormality is mostly of interest in low and intermediate risk cases [4].
- Numerical Chromosome Abnormality (NCA): if there is no SCA, and either a chromosome is missing from a pair, or there are more than two pairs of a chromosome (trisomy, tetrasomy, …);
- No Result: if none of the above mutations are found. No result must be due to a flat profile (no mutations), but also to the poor quality of the analysis (no sufficient material);

DNA Microarray is one of the oldest and the most established techniques for Gene expression estimation, measuring the quantity of RNA in the cell's cytoplasm (see [5] and Sect. 1.3). I created within the genomic analysis process type a dedicated sub-process type for each of the two analyses (CGH and Microarray). The sub-process types contain a set of (sequential) events to track the whole processing pipeline. Details of the Microarray analysis workflow are shown in Fig. 4.2. At BiolMol we store the raw data as .CEL files, and we performed two different microarray normalisations (MAS5 or RMA [6]), depending on the study requirements. We also store reports about a set of outcome and prognostic feature predictors, developed at IGG, and based on two machine learning classifiers: (i) a multilayer perceptron neural network developed by Cornero et al. [7], and a Logic Learning Machine (LLM) algorithm [8] implemented with the Rulex proprietary machine learning suite [9]. I have adopted the general purpose module inspired by the Command design pattern, as described in Sect. 3.4 to update stored material following deliveries of part of it to other labs or institutions. In this scenario, the *check* method controls the selected sample quantity and compares it with the quantity to be delivered. If the latter is greater an error message is returned and the procedure stops. Otherwise, a new sample delivery event and related data are registered in the database and the *execute* method updates the remaining quantity. If any error occurs during the new data registration, the *recovery* method restores the previous quantity value.

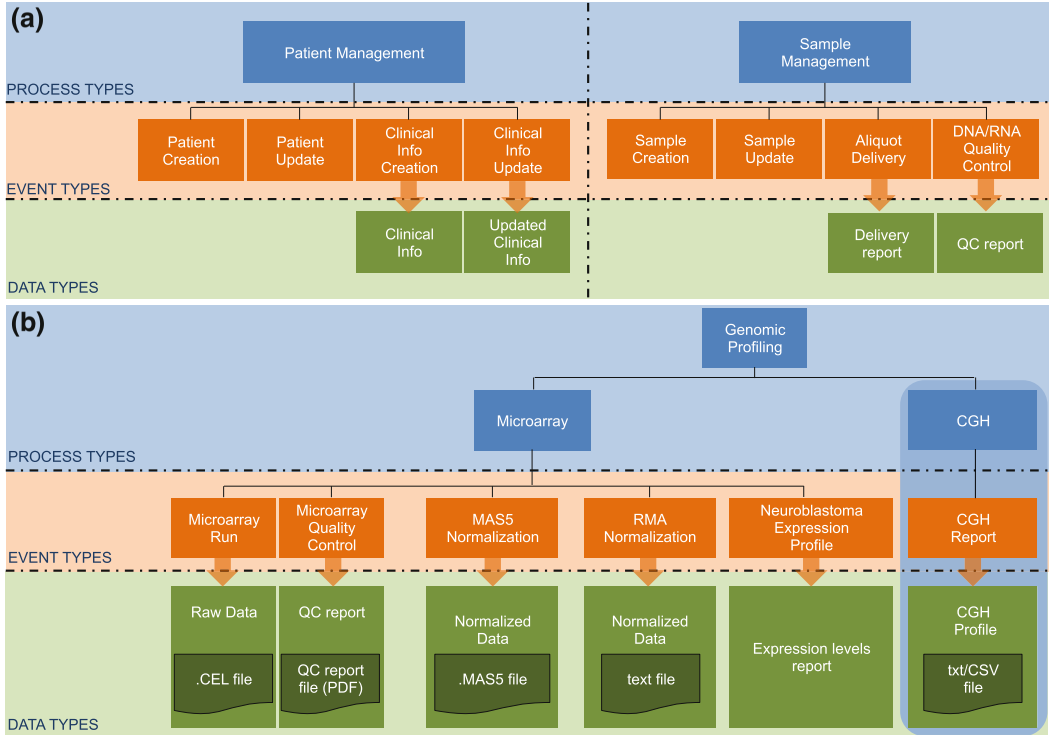

Fig. 4.2 Detailed process-event schema for Gaslini BIT data management. The biobank management system consists of the main process types: **a**-*left* 'Patient Management', **a**-*right* 'Sample Management', and **b** 'Genomic Analyses'. Through 'Patient Management' I track patient creation and update together and Clinical Information periodical update. 'Sample Management' provides events for sample creation and update, aliquot delivery and for storing quality control reports on DNA/RNA extractions. Each data instance is described by a set of metadata customised on the basis of SOP requirements and standards. 'Genomic Profiling' is made of two sub-processes to track each step of microarray and CGH analyses, and further processing steps. Only the microarray process type is fully detailed in the picture

4.2.1 Sample Management

Sample management has two principal activities: sample registration (i.e. creation of a new sample entity in the database) and search of samples, based on their metadata and on other data (such as molecular and genomic analyses) performed on the samples. In Fig. 4.3 is shown the form used to register a new sample. All the four main sample types (tissue, fluid, DNA, RNA) currently used in BIT expose a common interface with a two shared fields (*BIT code*, *Arrival Date*). The other sample metadata differ depending on the type. For primary samples, it is possible to specify a hospital and a unit of provenance, together with a possible external code, assigned before the sample's arrival at BIT. For tissue and fluid samples, Sample Codification and Pathology are values selected from controlled vocabulary, that can be extended by authorised users. I have personalised the original XTENS research page to handle complex searches within the BIT digital biobank. Searches can be performed either on patients or tissue/fluid samples, depending on what is the focus of the operator. For sake of brevity I will describe now only the sample search, which is the more detailed of the two, since it contains all patient data as well. The search page, shown in Fig. 4.4, is a multi-tab form, where search fields are divided on five panels, where the user can select the fields she/he is interested in querying and/or visualising. The first panel contains metadata fields related to the subject, the second (specific for patients enlisted in the neuroblastoma projects) contains a list of metadata describing clinical,

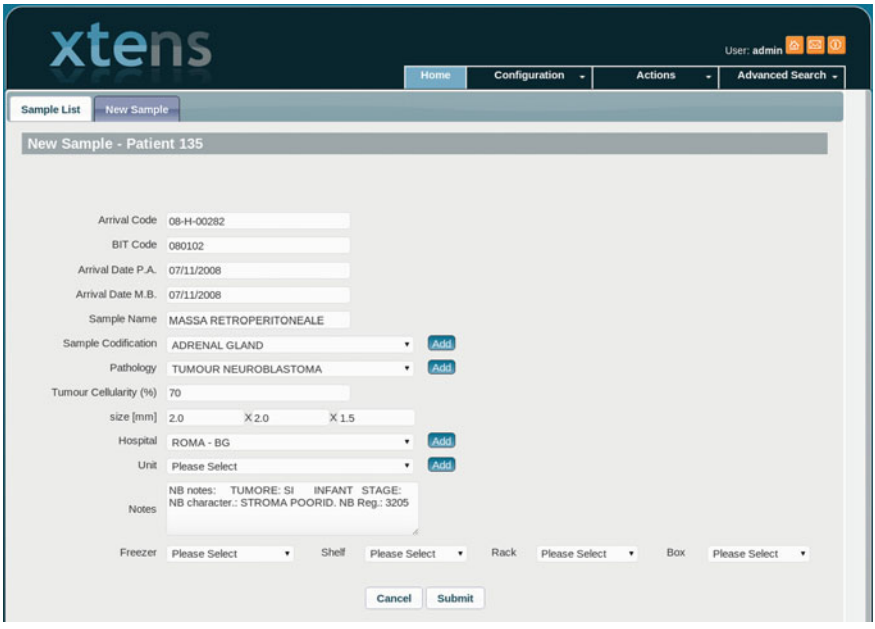

Fig. 4.3 Interface for a tissue sample creation and update

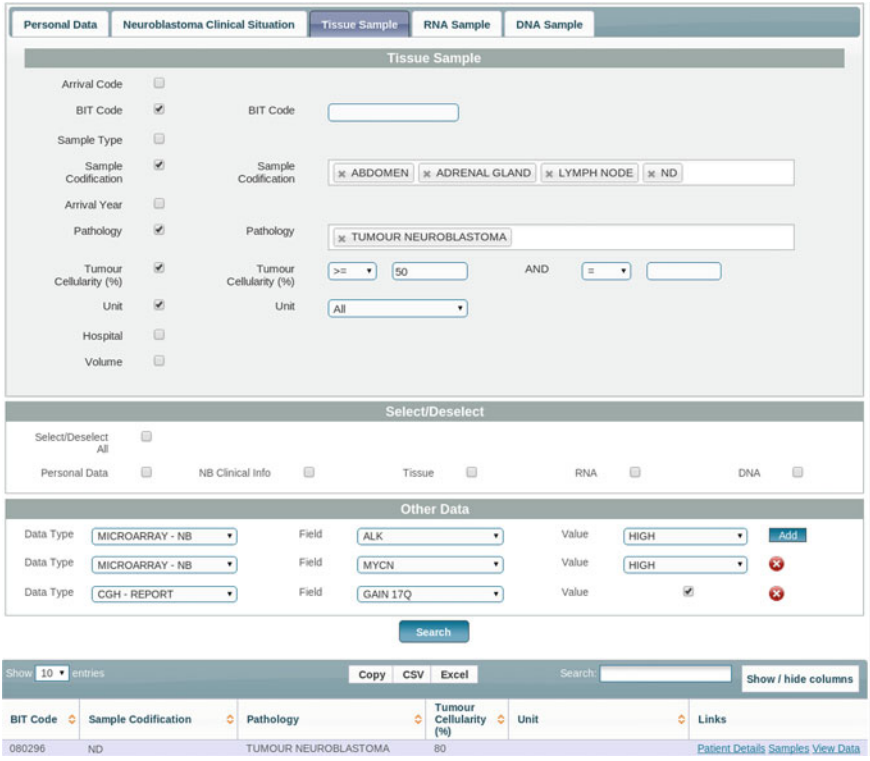

Fig. 4.4 XTENS graphical query builder

biological and molecular parameters significant for neuroblastoma prognosis. These parameters are stored in a dedicated table to allow *ad hoc* searches and separate management of sensitive information. The third table contains all the metadata fields used to describe tissue and fluid samples while the last two panels allow to add conditions on the derivative DNA and RNA samples extracted from the tissues and fluids. Each tab contains a full list of fields; using a checkbox list users select the fields they want to be shown in the result table. For each selected item they can specify one or more values for the query. Additional tuning can be performed using the custom-defined data types and metadata fields as they have been described in Sects. 3.3 and 3.4. The result provides an integrated view on all the requested information, as it is shown in Fig. 4.5. For each sample the full list of stored data instances can be visualised, and for any data instance the associated metadata and files. Authorised users can export the result as CSV or Excel file and download files stored in the data grid. External applications can access data and files using a RESTful web service interface.

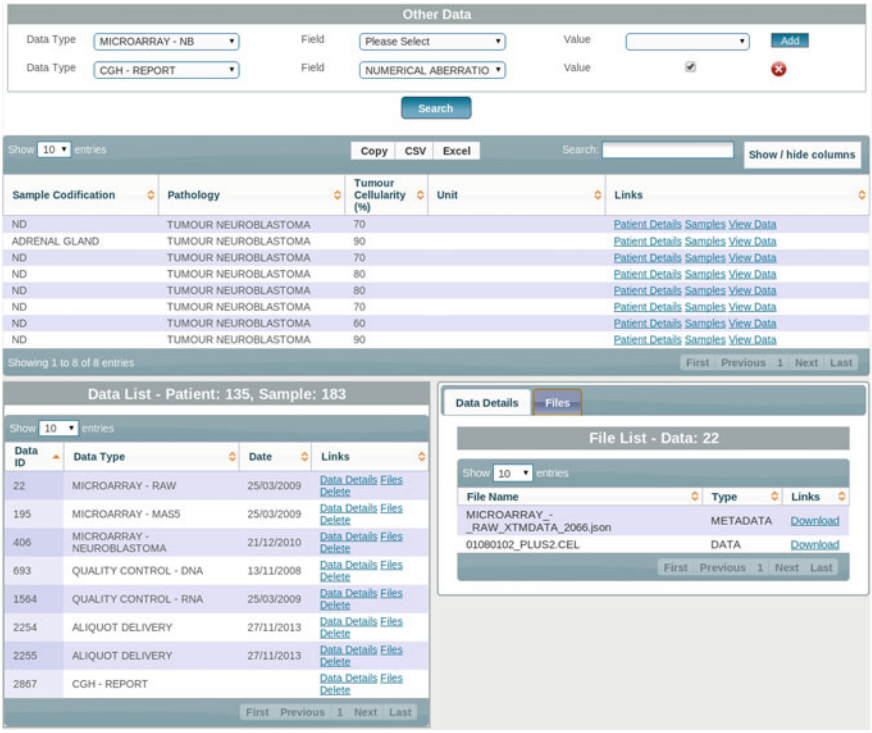

Fig. 4.5 XTENS graphical query builder

4.2.2 ALK Management

Since September 2014, the Biotherapy Department of the San Martino-IST Hospital, in Genoa, participates to the collaboration, providing the results of the mutation analyses of the gene ALK for a specific category of high-risk neuroblastoma patients. Germinal mutations of ALK are the main cause of familial neuroblastoma [10], while somatic alterations occur in ∼8 % of all neuroblastoma cases [11]. XTENS manages the ALK workflow as follows: the clinicians from the Oncology Clinical Department (OCD) of IGG enlist new candidates. A new "ALK analysis" process is activated on XTENS. The biologist checks whether there is DNA available on the biobank, and sends an aliquot to San Martino-IST. The analysis is currently executed using Sanger sequencing and by the end of the year will be executed with targeted NGS. Once the analysis is run, an ALK report is uploaded on XTENS and the associated metadata are populated. On the graphical web form, shown in Fig. 4.6, the operator can report the found mutations in the exons of interest for neuroblastoma (exons 20–25), the overall result of the analysis, and (optionally) whether a germinal mutation was found. The

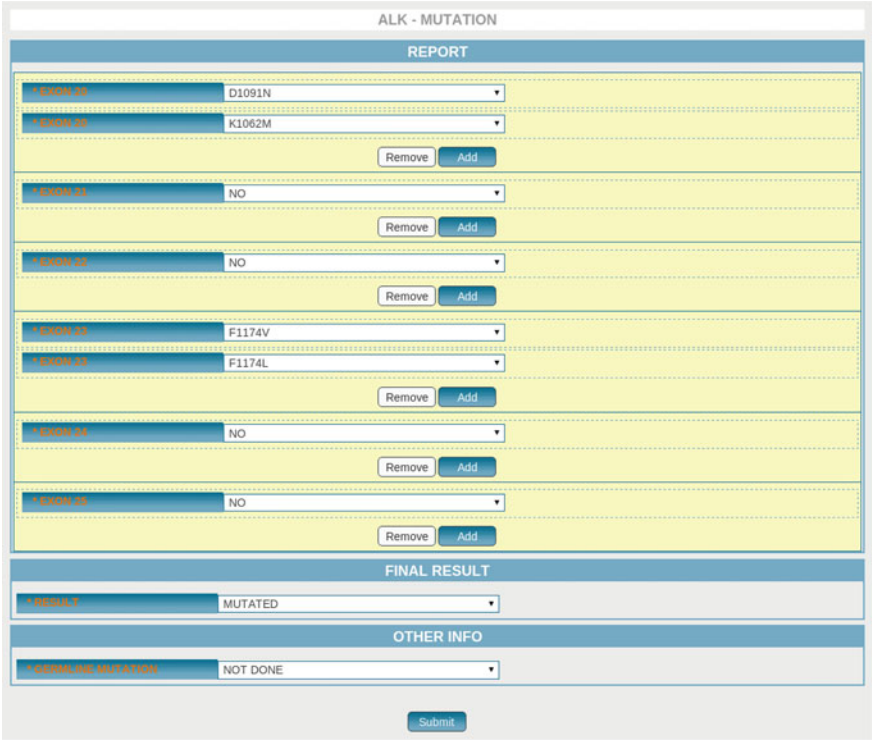

Fig. 4.6 The web form for reporting an ALK mutation analysis

"ALK analysis" process is closed and the results are available online to consult for the clinicians. As of March 2015, 30 ALK mutation reports are recorded on XTENS 1.5 at IGG.

4.3 BIT Installation Setup

All the Digital Biobank Platform is currently hosted on two servers with RAID 5 storage virtualisation located at IGG, equipped with Ubuntu Linux 12.04 LTS. I set up on the two machines an iRODS Zone (i.e. an autonomous iRODS system), named *iggZone*. iggZone consists of an ICAT-Enabled Server (IES), a secondary iRODS Server, and two storage resources: *mainResc*, located in an iRODS vault within the IES server, and *bakResc*, located in a vault within the secondary server. A single user—named *xtens*—is activated for users accessing iRODS from the Digital Biobank portal. All the files uploaded from XTENS are stored inside the collection */xtens/xtens-repo* and further divided depending on patient and data type identifiers. For instance, a microarray CEL file named *test_gene_expr.cel*,

belonging to patient *PAT001* will be stored at the irods path '*/iggZone/home/xtens/xtens-repo/microarray/PAT001/test_gene_expr.cel*'.

I have modified an iRODS system rule for file management after submission, called *acPostProcessForPut*, to check file integrity and automatically manage replication of data on the two resources for all the files saved in iRODS using the Digital Biobank portal. The code snippet is shown here as an example of iRODS rule syntax.

```
acPostProcForPut {
 ON($objPath like "/iggZone/home/xtens/xtens-repo/*")
    {
      msiSysChksumDataObj;
      msiSysReplDataObj("mainResc","bakResc");
    }
}
```

The Digital Biobank platform has currently been used in production for two years (since September 2013). We have inserted over 1800 primary tissue samples together with all extracted DNA and RNA derivatives form over 1400 patients. I have written a set of Java procedures to retrieve information from spreadsheets and automatically populate database entities, to automatise time-consuming operations such as the initial data import from CSV/Excel files and the periodical update of clinical data. I tested the database performances using a fixed set of queries on tissue samples with multiple table joins. These queries are built with 4–6 different data and metadata parameters. Then, I used a MySQL database, populated with about 10,000 data instances, 120,000 metadata, and hosted on a 64-bit computer equipped with 3.5 GB of RAM. I identified a set of table indexes to optimise the test searches, and I was able to reduce the query time of two orders of magnitude. At the end of the optimisation, all the tested queries took less than 0.9 s to be executed, transmission overhead included. As a consequence of indexing, new data insert speed is 31 % slower. The system's performance is more than suitable for BIT daily workload. In perspective, performances can be improved by more tailored indexing, query caching, and possibly by alternative tuning of MySQL configuration properties, and this strategies will be particularly useful if the dataset scales up. I have published the JSON-based data model and the results about the XTENS 1.5 setup at IGG on *BMC Genomics* [12].

References

1. Haupt, R., Garaventa, A., Gambini, C., Parodi, S., Cangemi, G., Casale, F., Viscardi, E., Bianchi, M., Prete, A., Jenkner, A., et al.: Improved survival of children with neuroblastoma between 1979 and 2005: a report of the Italian Neuroblastoma Registry. J. Clin. Oncol. **28**(14), 2331–2338 (2010)
2. Pinkel, D., Albertson, D.G.: Comparative genomic hybridization. Annu. Rev. Genomics Hum. Genet. **6**, 331–354 (2005)
3. Janoueix-Lerosey, I., Schleiermacher, G., Michels, E., Mosseri, V., Ribeiro, A., Lequin, D., Vermeulen, J., Couturier, J., Peuchmaur, M., Valent, A., et al.: Overall genomic pattern is a predictor of outcome in neuroblastoma. J. Clin. Oncol. **27**(7), 1026–1033 (2009)

4. Schleiermacher, G., Michon, J., Ribeiro, A., Pierron, G., Mosseri, V., Rubie, H., Munzer, C., Bénard, J., Auger, N., Combaret, V., et al.: Segmental chromosomal alterations lead to a higher risk of relapse in infants with MYCN-non-amplified localised unresectable/disseminated neuroblastoma (a siopen collaborative study). Br. J. Cancer **105**(12), 1940–1948 (2011)
5. Brown, P.O., Botstein, D.: Exploring the new world of the genome with DNA microarrays. Nat. Genet. **21**, 33–37 (1999)
6. Lim, W.K., Wang, K., Lefebvre, C., Califano, A.: Comparative analysis of microarray normalization procedures: effects on reverse engineering gene networks. Bioinformatics **23**(13), i282–i288 (2007)
7. Cornero, A., Acquaviva, M., Fardin, P., Versteeg, R., Schramm, A., Eva, A., Bosco, M.C., Blengio, F., Barzaghi, S., Varesio, L.: Design of a multi-signature ensemble classifier predicting neuroblastoma patients' outcome. BMC Bioinform. **13**(Suppl 4), S13 (2012)
8. Cangelosi, D., Blengio, F., Versteeg, R., Eggert, A., Garaventa, A., Gambini, C., Conte, M., Eva, A., Muselli, M., Varesio, L.: Logic learning machine creates explicit and stable rules stratifying neuroblastoma patients. BMC Bioinform. **14**(Suppl 7), S12 (2013)
9. Rulex (2015). http://www.rulexinc.com/site/. Accessed 26 Jan 2015
10. Mossé, Y.P., Laudenslager, M., Longo, L., Cole, K.A., Wood, A., Attiyeh, E.F., Laquaglia, M.J., Sennett, R., Lynch, J.E., Perri, P., et al.: Identification of ALK as a major familial neuroblastoma predisposition gene. Nature **455**(7215), 930–935 (2008)
11. Molenaar, J.J., Koster, J., Zwijnenburg, D.A., van Sluis, P., Valentijn, L.J., van der Ploeg, I., Hamdi, M., van Nes, J., Westerman, B.A., van Arkel, J., et al.: Sequencing of neuroblastoma identifies chromothripsis and defects in neuritogenesis genes. Nature **483**(7391), 589–593 (2012)
12. Izzo, M., Mortola, F., Arnulfo, G., Fato, M.M., Varesio, L.: A digital repository with an extensible data model for biobanking and genomic analysis management. BMC Genomics **15**(Suppl 3), S3 (2014)

Chapter 5
Results: XTENS 2, A JSON-Compliant Repository

In this chapter I will present the efforts done to develop a novel, fully JSON-compliant environment for the JSON data model outlined in Chap. 3. The original XTENS repository and XTENS 1.5—from now on collectively named "old XTENS"—were developed using a Java-based software stack and a MySQL database backend. As illustrated in Chap. 4, it proved to be successful in managing the workflow of a medium-sized integrated biobank. However, there were various limitations to flexibility yet. These were in part due to the fact that the XTENS platform was first developed with XML-based schemas. These strictures became apparent as I collected various feedbacks from the biobank operators. Here I will outline the main points of criticality, that convinced me to move towards a thorough refactoring of XTENS core architecture. First, as it was shown in Sect. 3.4 and in Fig. 3.3, old XTENS treated Patients, Samples, and other Data as separate entities, without exposing a common interface.[1] This pronounced separation forces the programmer to develop a different method for each of the three models (Patient, Sample, Data) even though the method should perform the same operation. More critically, Patient and Sample do not have an extensible metadata property, which is present only in Data. Previously I have defined the UML model of Fig. 3.3 as having a mixed metadata model. Patient and Sample entities store metadata in dedicated tables, where each property is mapped to a column. Data entities store the JSON schema in a table column, and all the metadata fields are individually stored in the EAV catalogue. Every time the scientist/operator needs to add a metadata field to either Sample or Patient the database schema must be modified. The procedure involves several steps, and in most cases, if the user wants to perform complex queries on the new field, also the source code requires modification. A possible way to overcome this limitation is to create a new Data Type to contain all the "extended" properties of the Patient (or Sample) type. There should be only one instance of this Data Type for each correspondent Patient and/or Sample entity. This mixed solution seems to be feebler than the one adopted,

[1] Here I speak of interface in the Software Engineering sense: a common set of public methods and properties.

M. Izzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5_5

for instance, by openBIS (see Sect. 2.5.6), where it is possible to attach metadata elements to patients, samples and datasets in a uniform way.

A second issue related to the platform architecture was the lack of a service-oriented interface. In its original implementation there were no RESTful web services for CRUD operations on Patient, Sample, and Data. I developed a RESTful API, using the Jersey Java library, but, due to various XTENS internal model shortcomings that became apparent as I delved more into the code base, they just allowed a very limited functionality. For instance, because of a limited abstraction in the design of the EAV module, extensive source code rewriting would have been necessary, to implement a general GET method on Data instances based on their metadata values. Given the inner fragility of other parts of the core modules, I thought a better approach redesigning the service-oriented API *de novo*. Furthermore, the old XTENS platform was designed using JSP and servlet with a tight coupling between the front-end and the back-end. It was not infrequent to find some *scriptlet*—native embedded Java code intended to run on the server—written directly on the JSP files. In practice, a JSP file, which should contain only the representational details (i.e. the "view"), incorporated also server-side code retrieval operations. This is a serious violation of the Model-View-Controller decoupling principle. The adoption of scriptlets within a Java Server Page is highly discouraged since the issue of the code conventions for JSP 1.1 [1]. I had already removed all the scriptlets during the development of XTENS 1.5, but I could not get rid of other subtler couplings that avoided the code being modular and with a clear separation of concerns between client- and server- side operations. The lack of modularity generates, among others, the following issues:

- little or null code reuse
- the effort to add a new software feature to the platform is considerably high
- there is a high risk of uncontrolled error propagation due to the lack of a test suite, with unit and integration tests at least for the most critical functions.

It appeared clear to me that in order to achieve a meaningful improvement on all these aspects I had to perform a more radical modification on the software architecture. I decided that, if I was to adopt a different technological stack, I must set as a first goal to adopt an environment where JSON is a first-class citizen. This approach has brought me to a somehow uncharted territory, at least as long as biomedical digital repositories are concerned. One can easily risk to step into some pitfall, such as adopting a language that does not provide extensive support for relevant functionalities (e.g. solid libraries for ontology management). Nonetheless, JSON has gained the status of *de facto*, for data exchange in the web, and I think that there is no reason to not try to adopt the same solutions that have obtained such a wide consensus for the same good properties—simplicity and flexibility—that are required in modern biomedical data management systems. Given these assumptions, I have resolved to adopt a so called *full-stack JavaScript* solution, that is a set of loosely connected JavaScript-based technologies that allow to develop all your web-based platform—in my case, the XTENS 2 repository—using JavaScript. I will now describe the technologies and the architecture I have adopted on the server, and how this approach allows a complete decoupling between back-end and the front-end.

5.1 XTENS 2 Back-End Architecture

The building-block of the XTENS 2 back-end is *Node.js*, a server-side JavaScript environment, designed for executing applications in real-time [2]. Node.js—usually simply called Node—has at its core V8, Google's open source JavaScript engine. While V8 was conceived to support JavaScript on the browser—notably on Chromium and Google Chrome—Node aims to support long-running server programs. When it first came out, Node had two innovative characteristics that set it apart from other existing server-side environments. It did not rely on multi-threading to handle concurrent execution of business logic, and it was based on an asynchronous, event-driven, non-blocking, I/O model. Running on a single thread, Node has a lighter memory footprint than Apache or Tomcat. It is conceived for data-intensive (i.e. I/O intensive) programs while a computing-intensive operation would block all other concurrent requests. In practice, every time the Node process executes an I/O request, it does not stop waiting for a response (from the database or the remote file system, for instance). The code executing I/O saves the callback function and returns the control to Node. The Node event loop[2] will execute the callback once the data is available or the I/O request is satisfied. Once the response arrives, the waiting code will be executed (as soon a the Node process ends executing the current code, and any other program with higher priority is executed as well). JavaScript supports event callbacks and is therefore an excellent language to use with in asynchronous environment. On top of Node I have adopted *Sails.js* [4], a server-side JavaScript framework for developing modular web applications. Sails.js provides the following functionalities:

- An data-oriented MVC architecture that automatically generates a REST interface with the five CRUD operations—find, findOne, create, update, delete—for the models you have defined. For HTTP request processing and routing, Sails (in versions 0.9–0.11.x) takes advantages of Express [5], a minimal framework for Node-based applications.
- a database-agnostic ORM/Object-Document Mapper (ODM) called *Waterline*. Waterline is defined as a "storage and retrieval engine", and provides a uniform API to access items stored in relational databases (MySQL and PostgreSQL are officially supported), NoSQL databases (MongoDB, Redis…), protocols (LDAP), third party API (Twitter), volatile memory (for testing and development) and disk. Waterline supports associations (one-to-many and many-to-many) that work also across different connections or adapters: for instance, you can join a table stored on a MySQL database with a collection living on a MongoDB installation.
- the capability of defining declarative and reusable security *policies* for authorisation and access control. Policies allow/deny access to Sails controllers and controller methods at different granularities.

[2] An event loop is "an entity that handles and processes external events and converts them into callback invocations" [3].

- built-in support for WebSockets [6], a protocol that enables two-way communication between client and server. WebSockets are particularly useful for real-time functionalities, such as online chats, news reports or medical device readings. In Sails, WebSockets work on the same routes specified by the REST API, without having to develop a dedicated code base.

The next step was to identify a database which allowed me to store data using a JSON or JSON-compatible format, while supporting associations and transactions. For this reason, I dropped out MySQL.[3] NoSQL solution, while attractive for their schemaless[4] design and their scalability properties, do not provide solutions that are both fast and reliable for handling associations, especially many-to-many associations. The standard "NoSQL way" for associations is to embed records within other records and/or to store data redundantly (replicate across the associated entities). In the latter case, ensuring that all the collections are correctly synchronised becomes a major issue for the developer. Moreover, if the relations between the various entities are on multiple levels and of complex nature, a NoSQL document-based storage system is a poor fit. On these ground, I have chosen to adopt PostgreSQL (also known as Postgres). It is an open-source RDBMS with a solid community and extensive functionalities. It has been already adopted in various biomedical scenarios, as testified by my survey in Sect. 2.5, with good success. Last but not least, it provides storage of semi-structured data in JSON format. PostgreSQL natively supports the text-based JSON format—called `json`—since version 9.2. In 9.3, a set of operators to access properties within `json` were introduced. In Postgres 9.4—the major version released in December 2014—Postgres developers team introduced a binary JSON format, named `jsonb`, designed to support schemaless data storage on a relational database [7]. The only differences besides the format itself are that `jsonb` (i) does not preserve semantically insignificant whitespaces, (ii) does not keep track of the order of the object keys, and (iii) does not maintain duplicates of the keys (if duplicates are provided, only the last value is stored). All the JSON-specific functions and operators of 9.3 work on `jsonb` as well; additionally, new operators specific for `jsonb` have been introduced. A full list can be found on Postgres official documentation (see [8] for Postgres 9.4). Finally, for the management of bulk data and unstructured metadata, I have kept on using iRODS, for its good properties on distributed storage scalability, data management through user-defined rules, and internal metadata support. However, I decided to decouple the storage system from the XTENS 2 core application. The decoupling is handled through a Strategy design pattern (originally formalised by the Gang of Four [9]) and will be described in the next section. Its main objective is allowing easier interchangeability of the storage system, if specific projects have different requirements.

[3] At the time I started developing the system MySQL had no native JSON support. Binary JSON support in MySQL was first introduced in 5.7.8 (July 2015).

[4] Here and afterwards, "schemaless" means that no fixed (tabular) schema is enforced or constrained by the RDBMS on write. The schema may be provided or enforced by an external application (in our case XTENS 2).

5.2 XTENS 2 UML Class Diagram

As a first step, I applied at least a single major modification to the UML model described in Sect. 3.4 and outlined in Fig. 3.3, to solve the issue of the separate management of patients and samples. Therefore in XTENS 2 I have modelled `Data` as a generalised class devised to describe all the "things" (in the sense introduced in Sect. 2.1) that are managed by the repository. `Data` is still a typed class: its *type* property refers to a `DataType` object. `DataType` has a *schema* property—equivalent to the *jsonSchema* property in XTENS 1.5—that contains the metadata schema in JSON format. `Data` instances only contain, for each metadata field, its value and unit (if requested): all metadata fields are stored as JSON in the *metadata* property. The `Subject` and `Sample` classes are specialisations of `Data`, and their metadata is handled exactly the same way. In Sails terminology, `Subject`, `Sample`, and `Data` are called *models*.[5] Accordingly, the *model* property of a `DataType` object determines which is its target model (i.e. in which table its data instances are stored) and allows three values: 'Subject', 'Sample', and 'Data'. `Subject` has two additional associations. The first association is with the `PersonalDetails` model, which is used to store in a separate entity all the sensitive informations relative to a subject. A subject is univocally identified in the system by the its first name, last name and birth date. This triple represents the minimum information to be submitted if the system keeps track of the personal information. A second

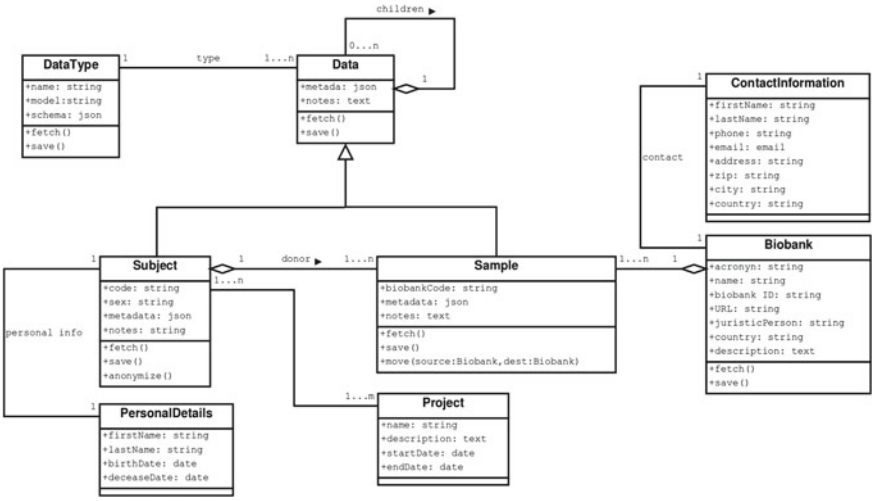

Fig. 5.1 XTENS 2 core UML class diagram

[5] I know, this is an unfortunate naming choice, because it may create ambiguity with the JSON metadata model. But I found all the other possible candidate names, namely *template*, *classTemplate*, or *class* even more inappropriate. Every time I refer to the JSON data model I call it explicitly *the JSON data model*.

association is with `Project`: I have modelled it as a many-to-many associations, thus allowing each subject to participate to multiple projects. A sample is associated to a subject through the *donor* attribute, and to a Biobank. In XTENS 2.0, a biobank is described according to the minimum information required by the MIABIS checklist (see Sect. 2.3.2 and [10]). This information is stored in two separate entities: `Biobank` and `ContactInformation`. The XTENS 2 data model allows to build a constrained hierarchical structure based on the entities `Data`, `Sample` and `Subject`. Each data instance can have one or more children data instances. Moreover, subject and sample can have as children both sample and data entities. The type of allowed children is specified by the relations that are defined among the various data types. Data types are related to each other with many-to-many associations: that is, a data type can have many progenitors and in turn it can have many descendants. The class diagram is shown in Fig. 5.1.

5.2.1 XTENS 2 Metadata Management

In a `Data` instance, each metadata field is stored within a JSON property having the same name of the field. Let's say for instance that we have a Data Type 'Clinical Report' having the following metadata fields: Diagnosis Age, Clinical status, and Diagnosis. For sake of simplicity they are stored in the *schema* of `DataType` with the field names converted to lower case and with whitespaces replaced by underscores. Therefore, the *metadata* attribute of a 'Clinical Report' `Data` instance contains the three property *diagnosis_age*, *clinical_status*, and *diagnosis*. Within each property there are two subproperties: *value*, where the actual metadata field value is stored, and an optional *unit* field.

This is the complete JSON *schema* of the 'Clinical Report' `DataType`:

```
{
    "header":{
        "schemaName":"Clinical Situation",
        "description":"an example of a patient clinical summary",
        "version":"0.1",
        "fileUpload":false,
        "model":"Data",
        "parents":["Patient"]
    },
    "body":[{
        "label":"METADATA GROUP",
        "name":"Clinical Report",
        "content":[{
            "label":"METADATA FIELD",
            "fieldType":"Text",
            "name":"disease",
            "iri":"http://purl.obolibrary.org/obo/OBI_1110055",
            "customValue":null,
            "required":true,
            "sensitive":false,
            "hasRange":false,
            "isList":true,
            "possibleValues":[
                "neuroblastoma","medulloblastoma",
```

```
                    "rhabdomyosarcoma","retinoblastoma","wilms tumour
                        ",
                    "osteosarcoma","leukemia","other tumour"
                ],
                "hasUnit":false,
                "possibleUnits":null
            },{
                "label":"METADATA FIELD",
                "fieldType":"Integer",
                "name":"diagnosis_age",
                "iri":null,
                "customValue":null,
                "required":false,
                "sensitive":false,
                "hasRange":false,
                "isList":false,
                "possibleValues":null,
                "hasUnit":true,
                "possibleUnits":["day","month","year"]
            },{
                "label":"METADATA FIELD",
                "fieldType":"Text",
                "name":"overall_status",
                "iri":null,
                "customValue":null,
                "required":true,
                "sensitive":false,
                "hasRange":false,
                "isList":true,
                "possibleValues":[
                    "complite remission","diseased",
                    "deceased","N.A."
                ],
                "hasUnit":false,
                "possibleUnits":null
            }]
        }]
}
```

In the header, it is specified that no files can be associated to this data type (*fileUpload* is set to `false`), that the data type is applicable to a generic `Data` object (no `Subject` or `Sample`), and that data instances of the type 'Clinical Situation' can be children of data instances of the type 'Patient' (i.e. the parent type). The schema body consists of a single Metadata Group containing in turn the three aforementioned metadata fields. The first one, *disease*, is a textual field whose name is associated to an IRI specified by the Ontology of Biomedical Investigation. A value is required and must be chosen from the controlled list specified in *possibleValues*. The second field, *diagnosis_age* is a numeric (integer) value that admits three possible measure units: days, months or years. The last field, *overall_status* is another required textual field with four admitted values. For the second and third metadata fields, no ontology identifier is provided. A data instance of 'Clinical Situation' contains a *metadata* field such as the following:

```
{
    "disease": {
        "value":"neuroblastoma"
    },
    "diagnosis_age": {
        "value":23,
        "unit":"month"
```

```
    },
    "overall_status": {
        "value":"complete remission"
    }
}
```

5.3 The Query Builder

My final aim with the new XTENS application was to provide a more flexible query tool for advanced data search on a heterogeneous data set. As a purpose of example, I will proceed to illustrate the query builder using an Integrated Biomedical scenario similar to the one outlined in Chap. 4. The query in the query builder can be constructed using any of the data types that a user is allowed to use. Let us consider an example where I have the following data types: Patient (model `Subject`), Tissue, Fluid, DNA, RNA (model `Sample`), Clinical Situation, Microarray Report, CGH Report, Genome Sequencing and Variant (model `Data`). Patient has three children: Tissue, Fluid and Clinical Situation. Tissue and Fluid share the same two children: DNA and RNA. RNA begets Microarray Report, while DNA originates CGH Report, Whole Genome Sequencing and Human Variant. The relationships among the various data types are described in Figs. 5.2, 5.3, 5.4, together with all the metadata fields specified in each data type schema. The query builder allows the user to configure queries using a graphical form on the web client interface. There are basically two main categories of queries that can be performed:

1. queries performed on a single data type, such as Microarray Report, CGH Report, Genome Sequencing, or Human Variant.
2. queries with conditions on multiple data types. In this case the user identifies a root data type, that is the type that she/he wants to search. Then, it is possible to add further conditions on the descendants of the root data type. For instance, if we choose to perform a query on Tissue samples, we can add conditions based on the fields of DNA, RNA, Microarray Report, CGH Report, Genome Sequencing and Variant Annotation. The query builder web form automatically suggests the correct data types to compose the subqueries. In general, if the query is performed using parameters from N different data types, at least $N - 1$ table joins will be required to obtain the result.

I have designed the query builder client interface with the aim of keeping it simple and minimal. When the user first enters in the page, there are only two select boxes with a limited set of options. The first one, labelled as “Search for”, contains the list of data types that the logged user is allowed to search. The other one, labelled as “matching”, contains two options: “all conditions” and “any of the conditions”. Their meaning will be clarified afterwards. Let us first consider the simpler case, a query on a single data type without nested conditions. As the user selects the data type of interest, a new select box and two buttons appear on the page. The select box, labelled “Field Name”, contains a list with the name of the metadata fields contained in the

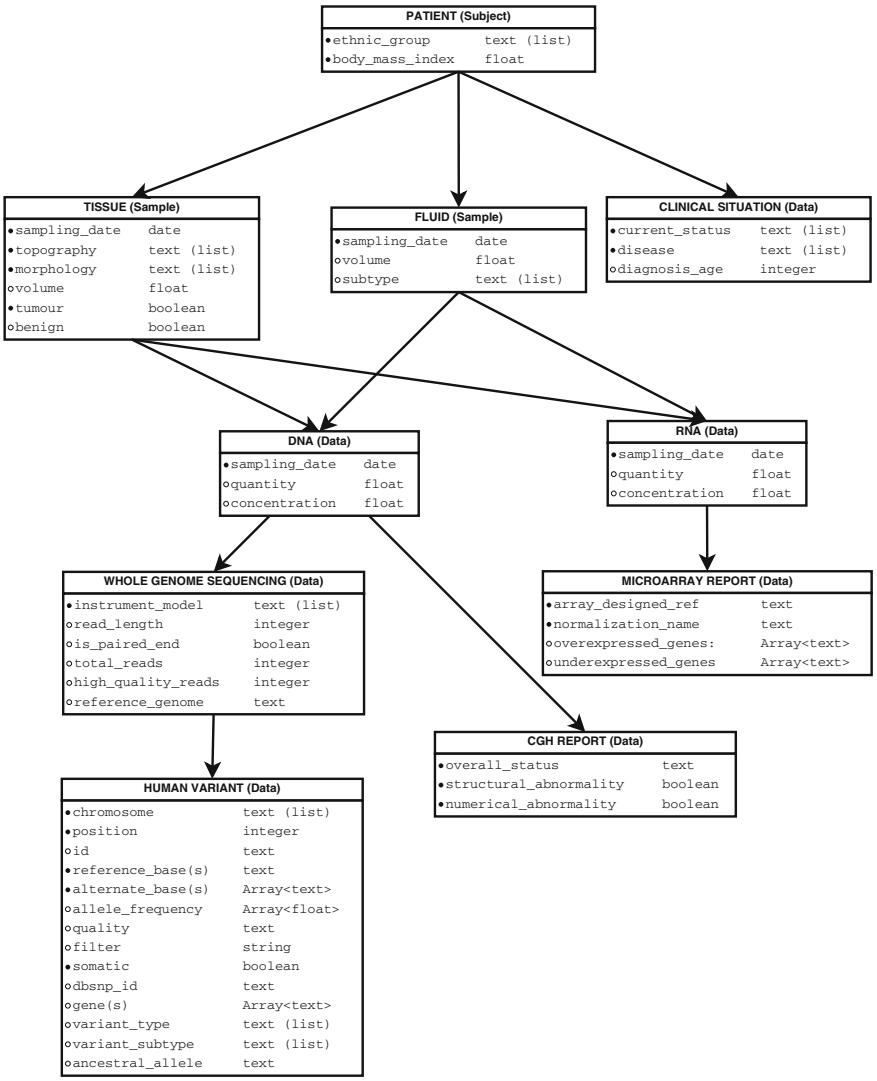

Fig. 5.2 The data structure designed and implemented to describe the query builder and test the query performance of the system

afore-selected data type schema. Once the user selects a metadata field, depending on its primitive type, additional filtering conditions can be enforced. If it is a numeric or date field the standard comparison operations can be applied ($=$, $<$, $>$, $\leq$, $\geq$). If it is a textual field, only equality and inequality comparisons are allowed; when the field values are limited to a controlled list, it is possible to select more than one option at once. The first of the two buttons, labelled "Add Field", allows to add new

ID	Name	Model	Parent	
5	Fluid	Sample	Patient	Edit
6	DNA	Sample	Tissue, Fluid	Edit
4	Tissue	Sample	Patient	Edit
7	RNA	Sample	Tissue, Fluid	Edit
9	CGH Array Report	Data	DNA	Edit
8	Clinical Situation	Data	Patient	Edit
3	Patient	Subject		Edit
11	Human Variant	Data	Whole Genome Sequencing	Edit
10	Whole Genome Sequencing	Data	DNA	Edit
12	Microarray Report	Data	RNA	Edit

New Data Type

Fig. 5.3 A table summary of all the data types defined in the schema of Fig. 5.2

conditions on the metadata fields. This should allow to build a sample query such as the following two:

1. retrieve all tissue samples of neuroblastoma extracted from the adrenal gland and liver whose mass is greater than 0.5 grams
2. retrieve all the somatic variants in the following region of interest (Chr16: 72,801,786–73,107,534)

When the user runs the search after composing the query, a client-side procedure parses all the fields and converts the web form to a JSON-based structure. The client sends a `POST/query/dataSearch` request to the XTENS server with the JSON parameters object in the request body. I have developed a dedicated Node module to compose the queries according to a specific strategy. In this way, the query composition procedure is decoupled from the core XTENS application, allowing easier interchangeability. Different strategies might be required for a number of reasons. For instance, in addition to the *metadata* attribute of Data (Subject, Sample), the single metadata values could be stored also on an EAV catalogue, or in dedicated table (for testing, but also for production purposes). Or, in another future scenario, it might be required to store the metadata in a different document store system, for instance a NoSQL database (e.g. the already mentioned MongoDB). As I have stated before most of the operations on the database are handled by the Sails ORM, waterline, that can be seamlessly used with different database solutions. Therefore, embedding the PostgreSQL-specific query composition in a strategy pattern will permit an easier switch to another database system or model paradigm (i.e. a different strategy), without the XTENS core system being aware of the change. The query

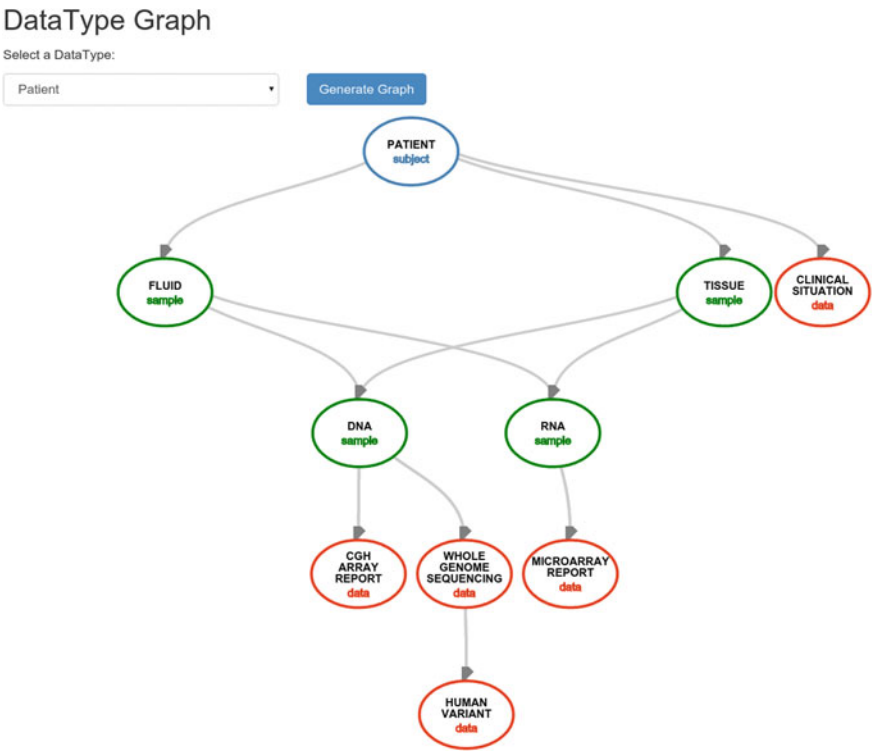

Fig. 5.4 The graph view of the data structure defined in Figs. 5.2 and 5.3, as shown in XTENS 2. The three different models (Subject, Sample and Data) are shown in different colours

composition model, which is simply called *xtens-query*,[6] is open source and available on GitHub [12]. The goal of xtens-query is to convert the JSON object contained in the POST request sent by the client into an SQL parametrised statement that can be executed by the RDBMS. The query is composed using a divide and conquer approach: the *dataType* and *model* terms are used to compose the SELECT clause, while each leaf—that is, each condition on a metadata field—is first parsed, converted to SQL, and stored within an array of strings. Then, all the elements in the array are joined using a junction operator, that was specified by the user, with the "matching" option (see above). If "matching" was set to "all conditions" the conjunction operator AND will be used to join all the terms. Otherwise, if the user selected "any of the operations" the OR operator will be adopted. The result string will be appended to the SELECT statement as its WHERE clause. In the current implementation is not possible to mix AND and OR conditions on the same WHERE clause. Figures 5.5,

[6]To futher decouple the Postgres database from the XTENS application, the xtens-query together with all the transactional CRUD operations handled in the *xtens-transact* module have been merged at a later time (November 2015) in a single "Postgres" module, named *xtens-pg* [11].

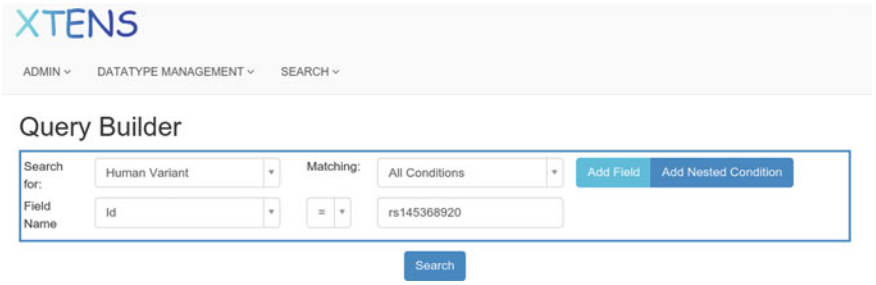

Fig. 5.5 Query a as composed from the XTENS 2 query builder interface

5.7, 5.9 and 5.11, in section show how the user composes query on the query builder client interface.

The only strategy currently implemented composes queries on the fields contained within the *metadata* JSON object. PostgreSQL provides various operators for accessing the properties of a `json`/`jsonb` element. Here I provide a summary of the most useful ones:

- `->` is used to access both JSON elements within an array (specifying the element index) and a JSON element field by key;
- `->>` works as `->`, but returns the array element or the object field as `text`. The returned element/field can then be further converted to other PostgreSQL native types using explicit type cast;
- `#>` is used to retrieve a JSON object at a specified path. The path is specified by a list of keys or array indexes;
- `#>>` works as `#>` but returns a text field instead of a JSON object;
- `@>` (only for `jsonb` fields) checks whether the right operand, which must be a valid JSON, is contained within the left operand. This is extremely useful for key-value searches and pattern matching, also because it can take advantage of the Generalised Inverted Index (GIN) of Postgres to perform the query.

5.4 Performance Tests

One of the aim of my study is to evaluate whether this new implementation is suitable to handle large datasets, such as those produced as output of a modern NGS analysis. I will not assess the management of the bulk data files, here—i.e. SAM/BAM files—because this will be delegated to iRODS, and the storage resources available for the tests were limited. I have chosen to focus specifically on the database performance in the case when a large number of variants, as those obtained after a variant calling analysis and stored in Variant Call Format (VCF) file, are saved as semi-structured metadata on the repository for query research. I have written a dedicated JavaScript

class to automatically generate and populate a dataset based on the data model shown in Fig. 5.2. I populated a tests datasets with 2500 subjects. These fictitious subjects are modelled as patients affected by a variety of tumours. For each patient, a tissue sample—for instance from a biopsy—and a whole blood sample have been collected in one among five biobanks afferent to the project. In this way, I can also test the multi-centric biobank management scenario. From each tissue sample, DNA and RNA have been extracted and used to perform aCGH and microarray analysis. Then, I have hypothesised that about the 50 % of the subjects (chosen randomly by the program) have undergone a Whole Genome Sequencing (WGS) analysis. In the generated dataset, 1242 WGS data instances were produced. Each WGS analysis was further annotated with a list of variants. The 1000 Genome Project estimated that there are approximately 10 million variants in each human genome, and of these about 15–20,000 variant are located in the exome (the whole set of protein-coding genes). I decided to consider only the exosome and annotated each WGS analysis with 20,000 variants each. I used two datasets of exosomal variants for populating the database: (i) the Exome Variant Server (EVS) dataset [13], containing only germinal mutations, and (ii) the dataset of protein-coding mutations from the COSMIC catalogue [14]. The purpose here was one of performance testing, not one of plausibility. Therefore I assigned randomly to each WGS analysis—that is, to each subject—19,500 germinal variants from EVS and 500 somatic variants from COSMIC. While 500 somatic variants is an over-estimation, as in average a tumours contains 33–60 somatically mutated genes, it is a reasonable number for colorectal and breast cancer [15]. In any case, an overestimation will not affect or falsify the results of the performance tests. I used the generated dataset to test the following test queries:

A. Retrieve (or count) the number of variants with a specific id (one condition);
B. Retrieve (or count) the number of SNP somatic variants from a specific chromosome (three conditions);
C. Retrieve (or count) the number of mutations in a specific chromosome and within a position interval [pos_min, pos_max];
D. Retrieve all the tissue samples of morphology “neuroblastoma” removed from a given anatomical position, “adrenal gland”. Of these samples I require that DNA and RNA be already available in sufficient quantity for further genomic analyses (e.g. at least 0.5 μg). Furthermore I require CGH Report to be already executed on the DNA, resulting in a bad prognosis (i.e. SCA profile). This query must be composed using four different (yet related) data types;

For each of the queries I compared the result obtained querying the `jsonb` metadata field, with the query performed against the EAV catalogue. The EAV was implemented in the database following the advices and guidelines exposed by Nadkarni ([16], see in particular Chaps. 11 and 15). All the attributes defined in the data types were stored in a dedicated table named `eav_attribute`. Then I implemented a separate EAV catalogue for each of the three models (`data`, `sample`, and `subject`) that contain the entities. For each model I created five separate tables, one for fundamental field type (text, integer, float, date,

boolean). The catalogue tables are named according to the following convention: `eav_value_{type_name}_{model_name}`. For the Data model the five EAV tables are: `eav_value_text_data`, `eav_value_integer_data`, `eav_value_float_data`, `eav_value_date_data`, and `eav_value_boolean_data`.

Each EAV catalogue table contains two columns "entity" and "attribute", respectively referring to the model table and `eav_attribute`, and a "value" column where the actual metadata value is stored. The database was installed on an Ubuntu 12.04 LTS virtual machine with 4 GB of RAM. The RDBMS parameters were tuned according to the guidelines of Postgres official documentation and the indications provided by Smith [17]. In details, `shared_buffers` was set to 950 MB (∼25 % of free memory), `effective_cache_size` was set to 2500 MB (∼65 % of free memory) and `work_mem` was increased up to 32 MB to encourage the building of large hash tables in memory (otherwise, the RDBMS switches to disk which is slower). The data tables were indexed on *type* (using BTREE) and *metadata* (using GIN), while the EAV catalogues were indexed (using BTREE) on *entity*, *attribute*, and *value*. Additionally, a multi-column index was created for the pair (*attribute*,*value*). The textual catalogue of data (`eav_value_date_data`) were indexed on the MD5 hash of *value*, because some cells exceeded the limit size for BTREE indexes. There is a caveat here: the MD5 index cannot be used for pattern searches using the `LIKE` operator (e.g. search all terms containing the pattern 'neuro' to find neuroblastoma, ganglioneuroblastoma, and ganglioneuroma all at once). However, this is not an issue in the queries I am testing. The whole database was analysed before running the tests, to allow Posgres to collect statistics about all tables. The `data` table contained over 25 million records and has a size of 22 GB. The largest EAV table is `eav_value_text_data`, counting over 247 million rows for 15 GB. The other four tables together account for 5438 MB and over 94 million records. The GIN index on the *metadata* field of the `data` table weighs 1.2 GB (∼5.4 % of the table size), while the BTREE index on the *value* column of `eav_value_text_data` weighs 14 GB (∼93 % of the table size), as a consequence of using the MD5 hash, which is 32 characters long. Nonetheless, also the indexes on *entity* and *attribute* are larger, weighing 5.3 GB (∼35 % of the table size). The other EAV tables show similar patterns. Therefore, the indexes' cost in term of storage space is higher for the EAV catalogue.

I have executed each of the three tests detailed in Sects. 5.4.1–5.4.2 according to the following protocol:

1. before running each query I stopped the PostgreSQL server, flushed the system cache and restarted the RDBMS;
2. the statement, as formulated by xtens-query, was run a first time and its execution time was saved as "cold cache" performance;
3. the statement was executed three more times. The three values were recorded and their average was saved as warm cache performance. This procedure was inspired by the approach adopted by Chen [18] and the considerations of Smith (see [17], Chap. 10);

4. two onerous queries—i.e. a row count on large EAV tables—were executed, to avoid cache-independent memory effects of Postgres. When I first ran the tests, I observed noticeable performance boosts if the same query was executed immediately after restart, even after flushing the cache as explained at point 1.
5. points 1–4 were repeated for the equivalent statement written against the EAV catalogues;

The procedure above was repeated five times to collect 5 samples for each query strategy on cold cache, and 5 on warm cache. Cold cache represents a worst-case scenario, when the RDBMS has to retrieve all the data from disk, at a considerably slower speed. This is realistic for queries on tables that are accessed, or when the table(s) size exceeds the RAM capability.

5.4.1 1-Parameter Query

Query A, when composed on the query builder as shown in Fig. 5.5, will produce the following JSON message to be sent to the server:

```
{
   "dataType":11,
   "model":"Data",
   "content":[{
     "fieldName":"id",
     "fieldType":"text",
     "isList":false,
     "comparator":"=",
     "fieldValue":"rs145368920"
     }]
}
```

The class *PostgreSQLJSONStrategy* of the xtens-query module converts the JSON query message into a parametrised SQL statement. The conversion procedure can be described in pseudocode as follows:

```
SET $select = 'SELECT d.id, d.metadata';
SET $queryConditions = [], $parameters = [];
IF "junction" THEN:
  SET $junction = 'junction' value;
ELSE
  SET $junction = 'AND';
determine the target table of the query from 'model';
determine additional columns to retrieve and add to $select;
add the FROM clause (with target table) to $select;
FOR EACH element IN "content" array DO:
  IF (comparator IS equality OR inequality comparator) THEN:
    compose a query condition using the containment operator (@>)
  against the metadata column;
```

```
14        push '{element.fieldName:
15             {"fieldValue":element.fieldValue,
16              "fieldUnit":element.fieldUnit}
17           }' into $parameters;
18      ELSE
19        compose a range query condition using JSON
              accessor operators
20      (-> & ->>);
21        push element.fieldName, element.fieldValue,
              element.fieldUnit
22             into $parameters;
23      ENDIF
24      push resulting query condition in $queryConditions
          ;
25    END FOR
26    SET $where = join all elements $queryConditions with
          $junction;
27    SET $statement = join $select with $where using a
        WHERE statement;
28    RETURN $statement, $parameters
```

If the model property is set to "sample" or "subject" the query will be executed against one of those two tables. For the JSON message above, the resulting parametrised SQL query will be:

```
1     SELECT d.id, d.metadata FROM data d
2     WHERE d.type = $1
3      AND d.metadata @> $2;
```

where the parameters are:

```
1   [11, '"id":{"value":"rs145368920"}']
```

Currently there is no strategy implemented in xtens-query to convert the JSON message to a query against the EAV catalogue. I composed the queries manually and tried different formulations to identify the most performant, For the one-parameter search on variant IDs, the only value table to consider is `eav_value_text_data`, the table containing textual fields. The query can be written in one statement, as shown below. To obtain the best performance, I hypothesised that the primary key of the `eav_attribute` row corresponding to the attribute *id* of the data type Human Variant is already available to the server. This is reasonable when the query is composed graphically from an online tools.

```
1   SELECT d.id, d.metadata FROM data d
2   INNER JOIN (  -- subquery on variant ID value
3     SELECT v1.entity FROM eav_value_text_data v1
4     WHERE v1.attribute = 35  -- variant.id
5       AND md5(v1.value) = md5('rs145368920')
6   ) as varid ON d.id = varid.entity;
```

SQL allows different ways of writing this statement—for instance using Common Table Expressions (CTE, also called WITH statements) or wrapping the subquery inside an `IN` predicate—but the query written above gives the best performance

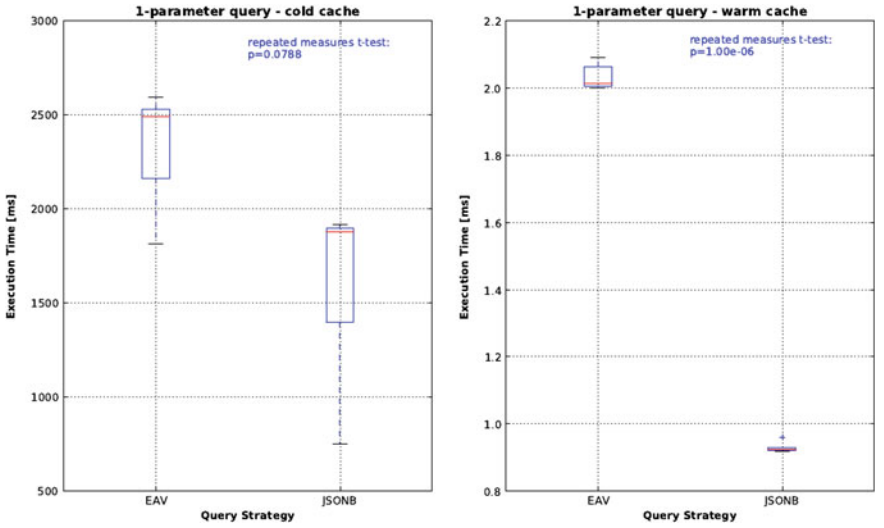

Fig. 5.6 One-parameter query performance on cold (*left*) and warm (*right*) cache condition

as the number of metadata parameters increases. The statement is highly selective, returning 15 rows out of the whole table. The results, as illustrated in Fig. 5.6, show that the two approaches are not significantly different when executed on cold cache, but that JSONB strategy is significantly faster once the data are loaded on RAM. The JSONB strategy has an average execution time of 1.6 ± 0.2 s on cold cache, which jumps down to 0.93 ± 0.02 ms on warm cache. The submillisecond execution time is consistent with the performances computed by `jsonb` developers [19]. The EAV approach takes 2.3 ± 0.15 s on cold cache and 2.03 ± 0.04 ms on warm cache. The JSONB strategy is 2.2 times faster than EAV on warm cache, even though after the EAV tables were clustered and their statistics were improved (see next paragraph). All the queries perform index scans on the tables and have optimised plans.

5.4.2 3-Parameter Query

The user composes query B from the query form as shown in Fig. 5.7. Here I omit the JSON message sent to the server, since its structure is analogous to the message produced by query A. The JSONB strategy of xtens-query will output the following parametrised statement:

```
1  SELECT d.id, d.metadata FROM data d
2  WHERE d.type = $1 AND d.metadata @> $2
3   AND d.metadata @> $3 AND d.metadata @> $4;
```

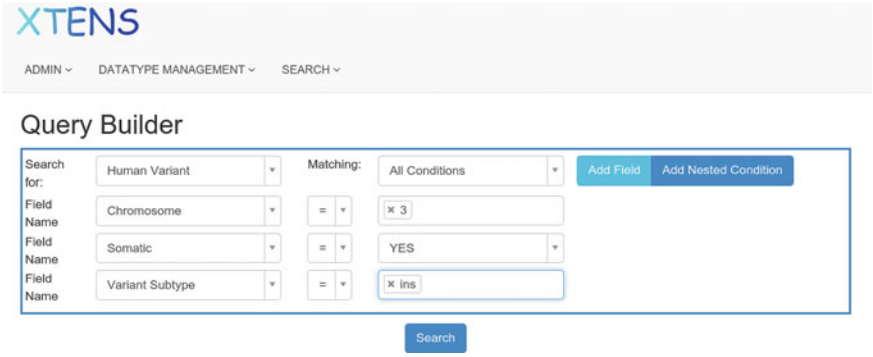

Fig. 5.7 3-parameter query as built from the XTENS interface

with the following parameters array:

```
[11, '{"chromosome":{"value":"3"}}',
'{"somatic":{"value":true}}', '{"variant_subtype":{"value":"ins"}}']
```

This statement returns 451 rows from the data table (i.e. ∼30 times the rows returned by query A), executing on in 10.5 ± 0.6 s on cold cache and 64.4 ± 0.1 ms on warm cache. Checking the internal strategy adopted by PostgreSQL query planner, I noticed that the query planner performs a single index scan checking all the three conditions on *metadata* at once. This means that writing the query in a more compact form (i.e. merging the three conditions on `metadata` in a single clause) does not improve the performance in any sensitive way. I did try the compact query, and the execution doubles, sticking at 128 ms. The worse performance is likely due to an overestimation of the returned rows by the query planner. Therefore, I have not considered the compact `jsonb` statement it in these tests. It is more challenging now to help PostgreSQL query planner in determining the best strategy for the EAV search. The query is:

```
SELECT d.id, d.metadata FROM data d
INNER JOIN (
    SELECT v1.entity FROM eav_value_text_data v1
    WHERE v1.attribute = 30  -- attribute id for 'chromosome'
      AND md5(v1.value) = md5('3')
) AS ch ON ch.entity = d.id
INNER JOIN (
    SELECT v2.entity FROM eav_value_boolean_data v2
    WHERE v2.attribute = 43  -- attribute id for 'somatic'
      AND v2.value = true
) AS so ON so.entity = d.id
INNER JOIN (
    SELECT v3.entity FROM eav_value_text_data v3
```

```
14      WHERE v3.attribute = 48 -- attribute id for '
            variant_subtype'
15        AND md5(v3.value) = md5('ins')
16  ) AS st ON st.entity = d.id;
```

When I executed it the first time over five trials on cold cache, its execution time was 995 ± 5 s, which is on average over 17 min. The execution time did not improve when run on hot cache, so the bottleneck was not the disk, but a wrong planning. PostgreSQL collects statistics of all the tables in a database to guess the number of rows returned by each query. Due to the high variability of the "value" field in the EAV catalogues, the query planner underestimated the number of returned rows by an order of magnitude, and executed the three joins using three nested loops. As a result the EAV execution time was several orders of magnitude greater than using the default xtens-query `jsonb` strategy. To improve the EAV performance, I performed two operations:

- increased the statistics target (i.e. the size of the data sample[7] used to compute statistics on a table column) for `eav_value_text_data.value` and for its index. The statistics target was first increased up from the default value of 100 to 500, and subsequently to 5000.
- clustered the rows in `eav_value_text_data` on the index that is adopted by the query planner. Clustering is a *una tantum* operation, which locks the table to reads and writes, and must be carried offline. New rows inserted subsequently are not clustered. The operation took over 32 h consuming over 99 % of the CPU resources.

Afterwards, the planner picked up more correct rows estimates, and the execution time dropped to 83.0 ± 2.2 s on cold cache and 4.15 ± 0.1 s on warm cache. Both cold- and warm-cache mean execution times are significantly better for JSONB strategy, when tested with a paired t-test (see Fig. 5.8). On warm cache, JSONB strategy is ~64.4 times faster than EAV. To put it in perspective, if a database server for a large biomedical project had to satisfy one such request every few minutes on peak workloads, it would be a necessary improvement. There are other approaches to further improve the EAV query performance. However, they require a careful tuning of the costs[8] and their modification could impair other most used queries. Query C, as shown in Fig. 5.9, takes once again three parameters. However, the *position* parameter is numeric and clause requires that a range of values be satisfied. For this clause the query builder cannot adopt the containment operator, and must resort to accessing the numeric values within the 'position' property of *metadata*. The JSON message posted to the server in this case is:

[7] As defined in statistics, "a data sample is a set of data collected and/or selected from a statistical population by a defined procedure" [20].

[8] The cost is an arbitrary unit used by the RDBMS to estimate the best query plan.

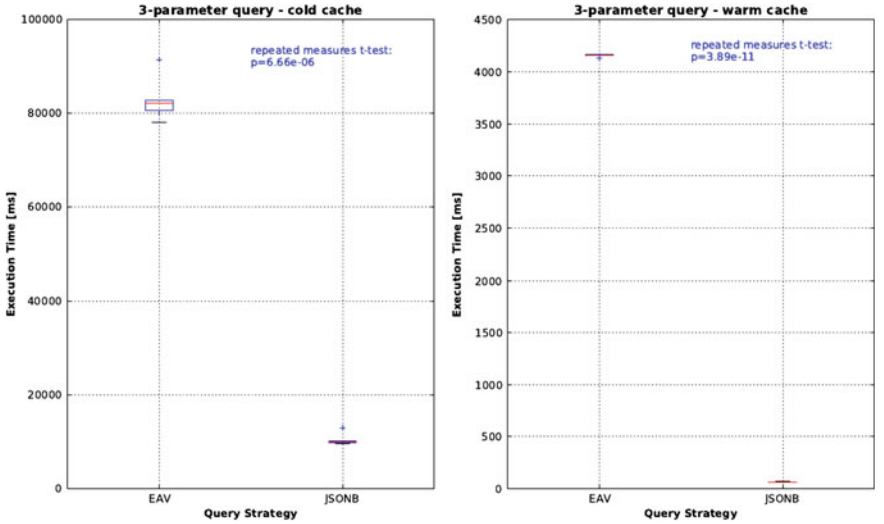

Fig. 5.8 Three-parameter query performance on cold (*left*) and warm (*right*) cache condition

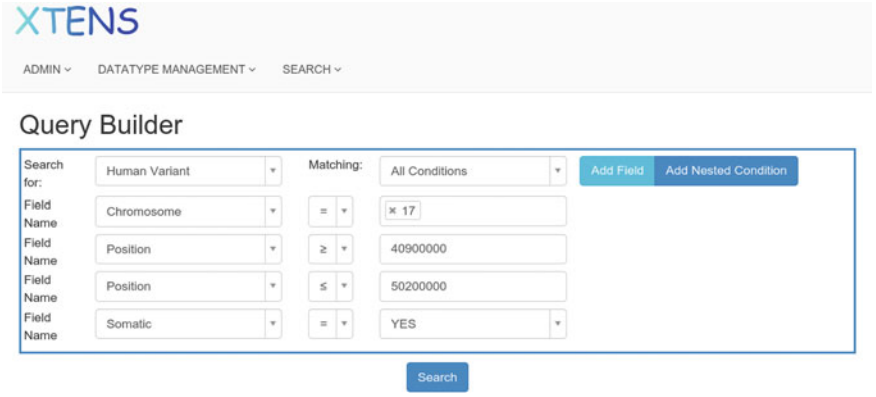

Fig. 5.9 3-parameter query with range condition as built from the XTENS interface

```
{
    "dataType":11,
    "model":"Data",
    "content":[{
      "fieldName":"chromosome",
      "fieldType":"text",
      "isList":true,
      "comparator":"IN",
      "fieldValue":["17"]
},{
      "fieldName":"position",
      "fieldType":"integer",
      "isList":false,
      "comparator":">=",
```

```
      "fieldValue":40900000
 },{
      "fieldName":"position",
      "fieldType":"integer",
      "isList":false,
      "comparator":"<=",
      "fieldValue":50200000
 },{
      "fieldName":"somatic",
      "fieldType":"boolean",
      "isList":false,
      "comparator":"=",
      "fieldValue":true
 }]
}
```

And the parametrised query, as computed from xtens-query is:

```
SELECT d.id, d.metadata FROM data d
WHERE d.type = $1
 AND d.metadata @> $2 AND d.metadata @> $3
 AND (d.metadata->'position'->>'value')::integer >=
     $4
 AND (d.metadata->'position'->>'value')::integer >=
     $5;
```

with the following parameters array:

```
[11, '{"chromosome":{"value":"3"}}',
'{"somatic":{"value":true}}',
40900000, 50200000]
```

The range condition and, to a lesser extent, the larger number of returned rows (4121) impair the query speed. The execution time on cold cache is 78.6 ± 1.2 s and 292.7 ± 1.7 ms on warm cache. The result does not improve if a `BETWEEN` operator is used in place of the two range clauses. Currently xtens-query does not support `BETWEEN` statements, but I plan to add them in the next future. The query against EAV, whose structure is similar to the previous one and is not reported here, takes 176.8 ± 0.8 s on cold cache and 5.27 ± 0.02 s on warm cache, respectively. Both conditions are significantly—2.25 times for cold cache and 18 times for warm cache—faster when executed by JSONB strategy (see Fig. 5.10). However, further optimisation are required if range-based queries are executed with a high frequency against the `jsonb` metadata. Dedicated functional (BTREE) indexes on numerical metadata values can help improving range-based search.

In conclusion, multi-parameter queries on a single data type are consistently executed faster using the schemaless `jsonb` strategy currently adopted by XTENS 2 rather than EAV, except on cold cache condition for 1-parameter query. The performance gap increases as the number of exact matching conditions increases, while tends to be less pronounced for range-based queries. More sophisticated EAV architectures and database tunings may further improve the performances and reduce this gap. Nonetheless, they require a deep knowledge of optimising the database, which is more common among professional database administrators than among academic researchers and software developers. The number of table joins in the EAV query increases proportionally to the number parameters in the query. Query optimisation

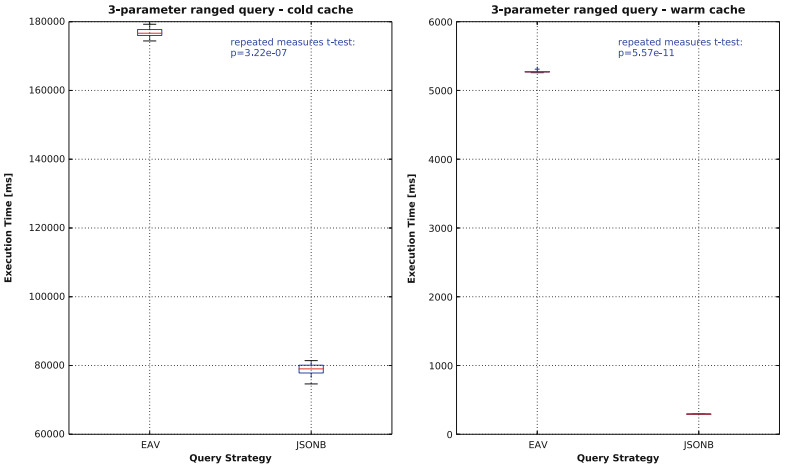

Fig. 5.10 Performance of a three-parameter query whith range condition on cold (*left*) and warm (*right*) cache

is a factorial ($n! = \prod_{k=1}^{n} k$) problem with respect to the number of tables involved in the join. The query planner cannot explore all the combinations and has to adopt some sub-optimal heuristics. Above few parameters, the developer has to explore alternative approaches—such as using materialised views or temporary tables—otherwise the execution times diverge. These database tunings and query rewritings are not required—at least for small–medium size projects and collaborations—to achieve query speeds below 100 ms for multi-parameter queries, using xtens-query default strategy.

5.4.3 *Multi-datatype Query*

Using the query builder, the user can also compose queries based on multiple data types if these are hierarchically related. Query D, for instance, can be composed as shown in Fig. 5.11, which is sent to the server as the following JSON object:

```
{"dataType":4,"model":"Sample","content":[
   {"specializedQuery":"Sample"},
   {"fieldName":"topography","fieldType":"text","isList":false,
    "comparator":"=","fieldValue":"Adrenal gland"},
   {"fieldName":"morphology","fieldType":"text","isList":false,
    "comparator":"=","fieldValue":"Neuroblastoma"},
   {"dataType":6,"model":"Sample","content":[
   {"specializedQuery":"Sample"},
   {"fieldName":"quantity","fieldType":"float","isList":false,
    "comparator":">=","fieldValue":"0.5","fieldUnit":"µg"},
   {"dataType":9,"model":"Data","content":[
       {"fieldName":"prognostic_profile","fieldType":"text","
           isList":true,
        "comparator":"IN","fieldValue":["SCA profile"]}
```

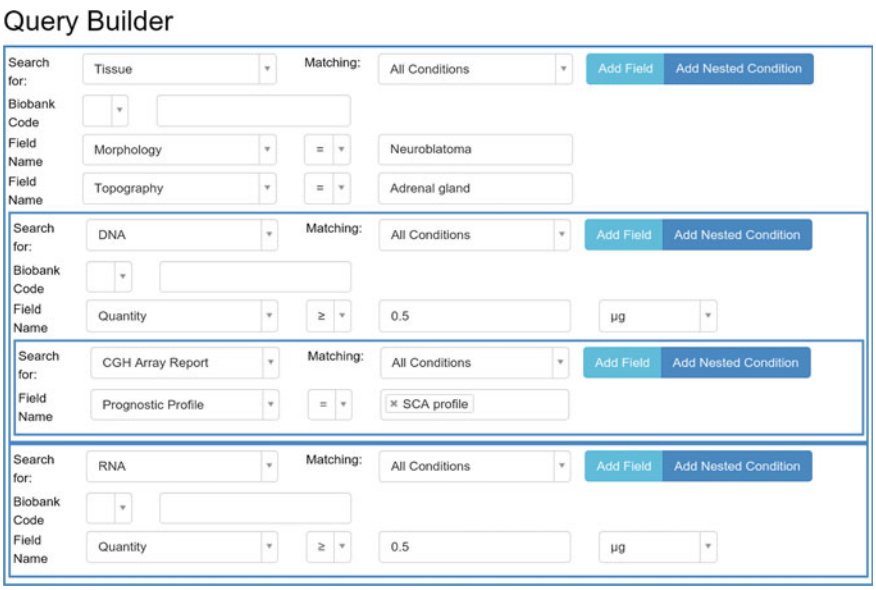

Fig. 5.11 A three-level hierarchical query con tissue samples

```
    ]}
    ]},
      {"dataType":7,"model":"Sample","content":[
      {"specializedQuery":"Sample"},
      {"fieldName":"quantity","fieldType":"float","isList":false,
       "comparator":">=","fieldValue":"0.5","fieldUnit":"µg"}
    ]}
]}
```

Now I describe how *PostgreSQLJSONStrategy* works on multi-level queries. A select statement is computed for each level of the query, that is for each data type specified in the "dataType" property. For `sample` and `subject` models the user may specify additional query parameters besides the ones that are contained in metadata, These parameters—such as the biobank and the code for a sample, or the personal details for a subject—are handled by dedicated functions. In our example there are no additional specialised parameters. All the single level queries are recursively built (using a depth-first tree traversal) and a single parameter array is generated. Then all the subqueries—in our example those on DNA, RNA, and CGH Report—are put into a WITH statement (i.e. a CTE). The main query is converted into a `SELECT DISTINCT` statement to ensure that each entity is returned only once from the table against which the main query is run (in our example the `sample` table). The views returned by the queries in the CTE are joined according to the data hierarchy (as shown in Fig. 5.11).

The resulting composite statement is:

```
-- WITH queries (Common Table Expressions)
WITH nested_1 AS ( -- subquery on "DNA" data_type
 SELECT id, parent_subject, parent_sample FROM sample
 WHERE type = $4
  AND (((metadata->$5->>'value')::float >= $6 AND
     metadata @> $7))
), nested_2 AS ( -- subquery on "CGH Report"
    data_type
 SELECT id, parent_subject, parent_sample,
     parent_data FROM data
 WHERE type = $8 AND metadata @> $9
), nested_3 AS ( -- subquery on "RNA" data_type
 SELECT id, parent_subject, parent_sample FROM sample
 WHERE type = $10
  AND (((metadata->$11->>'value')::float >= $12 AND
     metadata @> $13))
)
-- main query on "Tissue" data_type
SELECT DISTINCT d.id, d.metadata FROM sample d
 INNER JOIN nested_1 ON nested_1.parent_sample = d.id
 INNER JOIN nested_2 ON nested_2.parent_sample =
     nested_1.id
 INNER JOIN nested_3 ON nested_3.parent_sample = d.id
WHERE d.type = $1
 AND d.metadata @> $2
 AND d.metadata @> $3;
```

with the following parameters array.

```
[
 4, '{"topography":{"value":"Adrenal Gland"}}', '
 {"morphology":{"value":"Neuroblastoma"}}',
 6, 'quantity', 0.5, '{"quantity":{"unit":"µg"}}',
 '{"prognostic_profile":{"value":"SCA profile"}}',
 7, 'quantity', 0.5, '{"quantity":{"unit":"µg"}}'
]
```

This statement is much more time-consuming, due to the three JOIN operations and (if many duplicates are returned) the SELECT DISTINCT statement. These queries are intended for reporting and cohort identification, and not optimised to be executed in real-time.

5.5 Collaborative SEEG Project Use Case

At DIBRIS, I have set up a first XTENS 2 prototype installation dedicated to heterogeneous data management for a neuroscientific collaborative project focused on Stereoelectroencephalography (SEEG). The participants to the project are (i) Niguarda Hospital of Milan, providing data, (ii) the Neuroscience Centre (NC) at the University of Helsinki, sharing analysis methods and tools, and (iii) DIBRIS at the University

of Genoa, which took charge of data storage and management. The aim of the project is to exploit recent advancements in functional and effective connectomics to tentatively define biomarkers for focal epilepsy. Functional (and effective) connectomics studies describe how different brain areas interact with each other and how modification of such functional (or effective) couplings is directly linked to neurological diseases. In this context, it is quintessential to have access to high quality tools to store, analyse, and retrieve multimodal datasets, also in order to comply with national and international (in our case, of the European Union) regulations that govern the sharing of medical information and patients details.

Details about the methods adopted in data preprocessing and analyses can be found in dedicated publications [21, 22]. Here I briefly summarise the peculiar steps that interact with the XTENS 2 repository. SEEG is a highly invasive techniques to record neural activity that is routinely used in clinical application aimed at localising seizure onset zones in patients with drug-resistant focal epilepsy undergoing presurgical evaluation [23, 24]. Despite the sparsity of SEEG implants, recently the project consortium showed that SEEG can successfully be used in the context of functional connectomics studies fully exploiting its potential. The project participants estimated that ∼100 patients are required to obtain a 85 % coverage of all the possible interactions in a 250-parcel anatomical atlas.

In the outlined scenario the analysis workflow can be divided in two domains: structural and functional. Each domain is characterised by different data, methods and analyses outputs. The structural domain deals with anatomical data and is composed of a post-implant CT (postCT) scans that show the electrode in their final locations and pre-implant MRI (preMRI) that contain the information about individual brain anatomy. DIBRIS and NC provided the physicians with a set of medical image processing tools that are specifically designed to deal with SEEG implants (i) to localise each contact in both individual and common geometrical spaces and (ii) to assign to each contact its neuronal source on a probabilistic reference atlas (e.g. Destrieux atlas, see [25, 26]). The functional domain deals with signal processing techniques aimed at quantifying the degree of synchrony between brain regions and at characterising the so called functional connectome. XTENS successfully manages data describing both domains and provides client-side services for physicians to submit data and retrieve analyses results. I have installed an XTENS setup on a Linux Server (Ubuntu 12.04 LTS) at DIBRIS. Details of the installation are shown in Fig. 5.12. Together with the clinical collaborators, we have defined the following data types: `Patient`, `Preimplant_MRI`, `Postimplant_CT`, `Fiducial_List`, `SEEG_Implant`, `SEEG_Data`, and `Adjacency_Matrix`. `SEEG_Implant` data instances are the output of the segmentation process operated on `Postimplant_CT` using `Fiducial_List` metadata as reference. On the other hand, `Adjacency_Matrix` is the data describing brain region phase couplings in individual patient geometry estimated using `SEEG_Data`.

`Postimplant_CT`, `Fiducial_List`, and `SEEG_Data` are directly uploaded by the physician on the XTENS repository. I have developed a Node.js package, called *xpr-seeg* (source code available online, see [27]), to provide a web interface with the segmentation tool running on a separate server. The user triggers

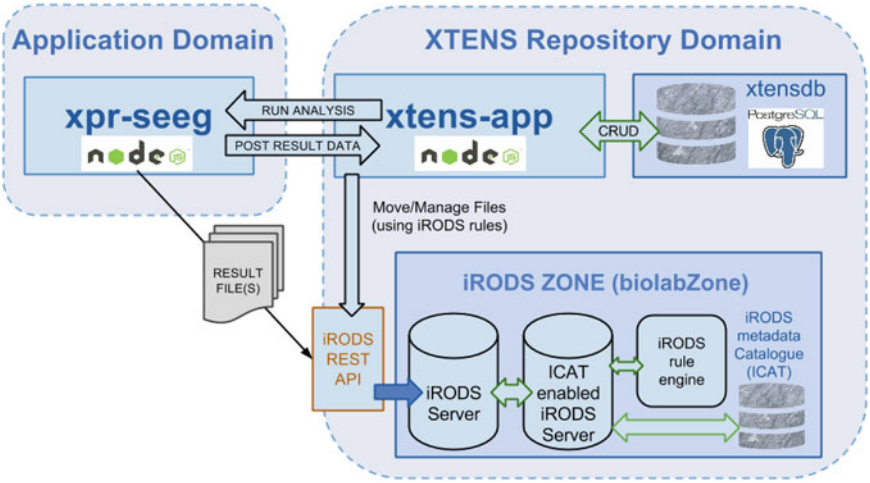

Fig. 5.12 XTENS 2 setup for the Stereo-EEG collaborative project. XTENS communicates with *xpr-seeg* using REST. After the analysis (e.g. segmentation) is run by *xpr-seeg*, the results' file is stored in iRODS and a new Data instance is saved on XTENS

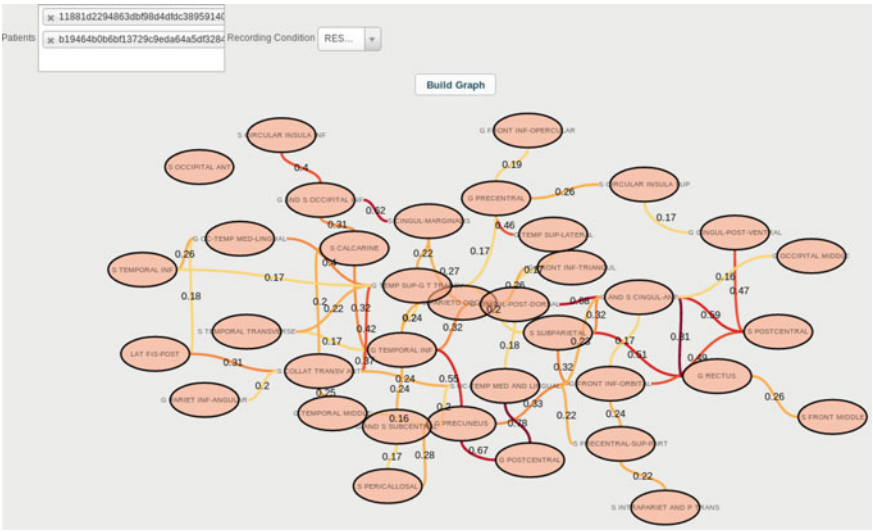

Fig. 5.13 A connectivity graph built from the client interface from the resting state adiacency maps of two subjects. The user selects the subjects and the study condition then runs the analysis and the graph is computed. Each ellipses corresponds to a brain region according to the Destrieux atlas

the tool through the XTENS client interface. In turn, XTENS sends a POST request to xpr-seeg forwarding all the required information about the two data instances. *xpr-seeg* executes a bash script that retrieves the required files from iRODS and runs the segmentation algorithm. Once the procedure is done, the computed SEEG Implant is

stored on a file. A novel data instance of `SEEG_Implant` is composed by *xpr-seeg* and saved in XTENS through a POST request. In a similar fashion, `SEEG_Data` are downloaded by operators, manually investigated to rejected artefactual channels (i.e., non physiological data) and analysed to build the `Adjacency_matrix`. Here, *xpr-seeg* provides the tool to correctly upload the analysed data to its data parent (i.e., Patient) in the XTENS repository. This setup is currently used for testing and development of the new features of XTENS 2. The XTENS 2 setup for SEEG currently hosts 42 patients with Preimplant MRI, Postimplant CT, and SEEG data. I have developed a dedicated module, under testing, to visualise connectivity maps mediating the Adiacency Matrices of a subset of patients under a specific condition (i.e. resting state, sleep, REM sleep, seizure). A provisional result is shown in Fig. 5.13. The details of this prototype are being presented at IWBBIO2015 conference, in Granada [28].

References

1. Mahmoud, Q.:. Servlets and JSP Pages Best Practices. Sun Developer Network (2003)
2. Tilkov, Stefan, Vinoski, Steve: Node.js: using JavaScript to build high-performance network programs. IEEE Int. Comput. **14**(6), 0080–83 (2010)
3. Understanding the Node.js Event Loop. http://blog.mixu.net/2011/02/01/understanding-the-node-js-event-loop/ (2011). Accessed 27 Mar 2015
4. Sails.js - Realtime MVC Framework for Node.js. http://sailsjs.org/ (2015). Accessed 09 Feb 2015
5. Express - Node.js Web Application Framework. http://expressjs.com/ (2015). Accessed 09 Feb 2015
6. Peter Lubbers and Frank Greco. HTML5 Web Sockets: A Quantum Leap in Scalability for the Web. SOA World Magazine (2010)
7. Lerner, RM.: At the forge: PostgreSQL, the NoSQL database. Linux J. **2014**(247), 5 (2014)
8. Postgresql 9.4 Documentation: JSON Functions and Operators. http://www.postgresql.org/docs/9.4/static/functions-json.html (2014). Accessed 10 Feb 2015
9. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Pearson Education, Upper Saddle River (1994)
10. Norlin, L., Fransson, M.N., Eriksson, M., Merino-Martinez, R., Anderberg, M., Kurtovic, S., Litton, J.-E.: A minimum data set for sharing biobank samples, information, and data: MIABIS. Biopreservation Biobanking **10**(4), 343–348 (2012)
11. XTENS-pg: A Query Module for the XTENS Repository. https://github.com/biolab-unige/xtens-pg (2015). Accessed 08 Dec 2015
12. XTENS-Query: A Query Module for the XTENS Repository. https://github.com/biolab-unige/xtens-query (2015). Accessed 17 Feb 2015
13. Exome Variant Server, NHLBI GO Exome Sequencing Project, Seattle. http://evs.gs.washington.edu/EVS/ (2013). Accessed 20 Mar 2015
14. Forbes, S.A., Tang, G., Bindal, N., Bamford, S., Dawson, E., Cole, C., Yin Kok, C., Jia, M., Ewing, R., Menzies, A., et al.: COSMIC (the catalogue of somatic mutations in cancer): a resource to investigate acquired mutations in human cancer. Nucleic Acids Res. **38**(1), D652–D657 (2009) (gkp995)
15. Vogelstein, B., Nickolas, P., Velculescu, V.E., Zhou, S., Diaz, L.A., Kinzler, K.W.: Cancer genome landscapes. Science **339**(6127), 1546–1558 (2013)
16. Nadkarni, P.M.: Metadata-driven Software Systems in Biomedicine: Designing Systems that Can Adapt to Changing Knowledge. Springer, Berlin (2011)

17. Smith, G.: PostgreSQL 9.0: High Performance. Packt Publishing Ltd, Birmingham (2010)
18. Chen, R.S., Nadkarni, P., Marenco, L., Levin, F., Erdos, J., Miller, P.L.: Exploring performance issues for a clinical database organized using an entity-attribute-value representation. J. Am. Med. Inf. Assoc. **7**(5), 475–487 (2000)
19. Schema-less PostgreSQL. http://www.slideshare.net/yandex/jsonb-jsquery (2014). Accessed 29 Mar 2015
20. Peck, R., Olsen, C., Devore, J.: Introduction to Statistics and Data Analysis. Cengage Learning, Stamford (2008)
21. Arnulfo, G., Narizzano, M., Cardinale, F., Fato, M.M., Palva J.M.: Automatic segmentation of deeep intracerebral electrodes in computed tomography scans. BMC Bioinformatics (2015). (to appear)
22. Arnulfo, G., Hirvonen, J., Nobili, L., Palva, S., Palva, J.M.: Phase and amplitude correlations in resting-state activity in human stereotactical EEG recordings. NeuroImage **12**, 114–127 (2015)
23. McGonigal, Aileen, Bartolomei, Fabrice, Régis, Jean, Guye, Maxime, Gavaret, Martine, Fonseca, Agnès Trébuchon-Da, Dufour, Henry, Figarella-Branger, Dominique, Girard, Nadine, Péragut, Jean-Claude, et al.: Stereoelectroencephalography in presurgical assessment of MRI-negative epilepsy. Brain **130**(12), 3169–3183 (2007)
24. Cossu, M., Cardinale, F., Castana, L., Citterio, A., Francione, S., Tassi, L., Benabid, A.L., Russo, G.L.: Stereoelectroencephalography in the presurgical evaluation of focal epilepsy: a retrospective analysis of 215 procedures. Neurosurgery **57**(4), 706–718 (2005)
25. Fischl, Bruce, van der Kouwe, André, Destrieux, Christophe, Halgren, Eric, Ségonne, Florent, Salat, David H., Busa, Evelina, Seidman, Larry J., Goldstein, Jill, Kennedy, David, et al.: Automatically parcellating the human cerebral cortex. Cereb. Cortex **14**(1), 11–22 (2004)
26. Destrieux, Christophe, Fischl, Bruce, Dale, Anders, Halgren, Eric: Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature. Neuroimage **53**(1), 1–15 (2010)
27. xpr-seeg: a query module for the XTENS 2 repository. https://github.com/biolab-unige/xpr-seeg (2015). Accessed 22 Mar 2015
28. Izzo, M., Arnulfo, G., Piastra, M.C., Tedone, V., Varesio, L., Fato, M.M.: XTENS-a JSON-based digital repository for biomedical data management. Bioinformatics and Biomedical Engineering, pp. 123–130. Springer, Berlin (2015)

Chapter 6
Discussion

My aims, as stated in the introduction, were:

1. To design a data model for biomedical research, able to describe flexible, poorly structured and constantly evolving data types and metadata. The model should allow the adoption of standards but not enforce them.
2. To implement the data model into a biomedical digital repository where scientists can configure and modify the structure of their data types without executing onerous IT tasks (such as modifying XML/SQL schemas, and recompiling, rebuilding, or reinstalling the application).
3. To test whether the data model is able to handle heterogeneous data in a local scenario (in my case, the Integrated biobank).
4. To test how the data model and the digital repository newly implemented (i.e. XTENS 2) can handle large datasets of semi-structured metadata (e.g. variant annotations).
5. To evaluate how the XTENS repositories 1.5–2 work in multi-centric and multi-disciplinary collaborations.

In Sect. 6.1 I will asses points 1 and 2. Points 3 and 4 will be discussed, respectively, in Sects. 6.3 and 6.4.

6.1 XTENS 1.5–2 and the State of the Art

In Table 6.1 I have summarised the main points of comparison among the last two XTENS implementations and the literature. Here I will provide some explanation, clarifying the advancements done on the current knowledge. XTENS 1.5 provided the first JSON-based data model to manage heterogeneous biomedical information. The adoption of JSON in lieu of XML provided a number of advantages. First of all, being JavaScript the natural scripting language on all major web browsers, the JSON data model can be parsed and processed, in XTENS 1.5 on the client side without

M. Izzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5_6

Table 6.1 Comparison among XTENS 1.5, XTENS 2, and the state of the art

Data repository	Biobank	Data model					File system
	Biobanking support	Format	Structured metadata	Semi-structured and unstructured metadata	Data compliance to code	Data model extensibility	Distributed
XNAT	No	XML	Mixed (tables/EAV)	XML	Requires binding	Rebuilt required	No
COINS	No	SQL	EAV	EAV	Native via JDBC	Rebuilt required	No
CARMEN	No	SQL	Not specified	–	?	Limited	Yes (SRB)
SIMBioMS	Yes	SQL	EAV	XML	Native (no XML)	Programming required	No
openBIS	Yes	SQL	Mixed (tables/EAV)	XML attachments	Native (no XML)	Rebuilt required	No
i2b2	Possible (external)	SQL	EAV	XML	Native (no XML)	Programming required	Yes?
XTENS 1.0	Yes one level	XML	Mixed (tables/EAV)	XML	Requires binding	At runtime (no restart)	Yes (iRODS)
XTENS 1.5	Yes hierarchical	JSON	Mixed (tables/EAV)	JSON	Requires parsing	At runtime (no restart)	Yes (iRODS)
XTENS 2.0	Yes hierarchical multiple BB	JSON	JSON	JSON	Native	At runtime (no restart)	Yes (iRODS)

loading the server with additional processing and computation. In XTENS 2, both the back-end and the front-end are coded in JavaScript, extending the benefit to all levels of the application stack. Compared to the XML-based repositories (XNAT, XTENS 1.0) no binding tool is required in XTENS 1.5 and XTENS 2. Overall, the JSON metadata schema I propose is a novel approach to document and describe in a highly flexible but consistent format heterogeneous datasets and information in biomedical science, both for clinical and research support. The metadata were stored as a JSON document in a textual field within the database, and as separate entities in the EAV catalogue. Queries on metadata fields were always run against the EAV, while the JSON document was used as a model to present and visualise the data. XTENS 2 adopted JSON for the storage of metadata in any format. In XTENS 2, JavaScript is *the* language both on the client and the server, therefore JSON is the native format, not only to exchange messages, but also to construct objects. In the state of the art—namely in XNAT, SIMBioMS, openBIS, i2b2, and XTENS 1.0—semi-structured metadata were stored as XML documents. Being stored in RDBMS (MySQL, Postgres, or Oracle) it follows that semi-structured metadata are stored as textual documents, which hampers query capability and performance. XTENS 2 adopts the binary JSON model, which allows to query large datasets in real-time.[1] An advantage of XML format is that XML schemas can be validated against XSD. JSON specifications do not yet provide validation support, even though JSON Schema [1], a JSON media type for defining the structure and validation of JSON data has been proposed. JSON Schema is currently in draft form, and I did not consider it sufficiently mature to be used in XTENS 2. XTENS 1.5-2.0 web clients perform a client-side validation of the metadata against its data type schema before form submission. A server-side validation of the metadata has been implemented in XTENS 2 core application, using the Joi validation library [2]. The server-side validation step can be turned off to increase write operations performances. The aim of XTENS 2 data model is suggesting a schema, not to force it, supporting the view of metadata in research collaborations as an adaptive, fluid and ephemeral process as stated in Sect. 2.4. For practical purposes, the scientists may want to store unstructured or semi-structured metadata in a different format along with the list of fields that are specified in the data type schema. These metadata will be transparent for xtens-query, but may be searched or inspected by third-party programs or *ad hoc* modules of XTENS developed in the future. A daemon tool can provide offline strict validation of the JSON metadata, if this is required.

The biobanking management features of XTENS 1.5 are comparable to those of SIMBioMS, allowing to describe DNA/RNA derivatives and aliquots as "children" of the primary sample. This hierarchical sample management was missing in XTENS 1.0. Furthermore, XTENS 2, introduces multi-centric biobanking management, a feature that in perspective, will allow to integrate multiple biobanks on a single repository installation.

[1] In my tests large datasets mean tens of million records with about ten-fifteen metadata each, and real-time means that (i) a single parameter query is run in less than 1 ms, and (ii) a multi-parameter query is executed in less than 100 ms, once the cache has warmed up.

The schemaless nature of `jsonb` allows also to implement horizontal scaling on PostgreSQL, a feature that is usually not available on RDBMS, but that characterise NoSQL document stores. The easiest way to scale PostgreSQL horizontally is though the `pg_shard` extension [3], an open-source tool developed by CitusData to shard and replicate PostgreSQL tables for horizontal scale and high availability, distributing the SQL statements without requiring any changes to the application. `pg_shard` does not support table JOINS, therefore the sharded tables should already contain all the information required for distributed search (i.e. they should be denormalised). CitusData also offers a fully scalable (i.e. JOIN support included), distributed extension to PostgreSQL called CitusDB, but this is a proprietary, commercial software.

6.2 XTENS 1.5 for Integrated Biobanking

At the IGG installation, XTENS 1.5 and its data model provide the users three main advantages, relative to data management. First, the process-event model, implemented as in Fig. 3.3, can manage both clinical visits/events histories and genomic experiments (with subsequent analyses and post-processing steps) in a uniform yet fluid way. Second, researchers and clinical operators can define new data types and describe them with customised metadata using a graphical web form, without dealing with JSON or XML formats directly, thus not requiring the help of a computer science expert. Third, XTENS 1.5 provides a user-friendly interface with a data grid system, which can easily scale-up to manage huge files such as high-resolution clinical images or WGS data. The choice of iRODS as a distributed storage manager has proved successful, because its metadata capability fits the scientists requirements of annotating the files with complex user-defined metadata. Moreover, iRODS is relatively easy and quick to install, and enables flexible data management thanks to its Rule Engine. Through iRODS, XTENS 1.5 can manage and track in a seamless and efficient way both relatively small datasets, such as microarray expressions profiles, and, in a soon-to-be future, larger ones, as whole genome sequences. iRODS is already used in production in various genomic centres and biomedical consortia, such as the Wellcome Trust Sanger Institute [4] and the Services@ MediGRID project [5]. In the two years of production, iRODS did not require any particular maintenance operation, so it presented no overhead when compared with a standard file system. No consistency errors have been found so far on the stored files.

6.3 XTENS 2: Performance and Scalability

In the performance tests I have run on XTENS 2, the JSON metadata model was a better storage tool than a general purpose EAV catalogue, for data search and retrieval. While the performances are on the same time scale for 1-parameter queries,

they worsen as the number of parameters increase. Overall, XTENS 2 schemaless metadata in PostgreSQL `jsonb` format work better than an EAV catalogue in these scenarios:

- the storage of large poorly structured (or semi structured) metadata. An example is the attribute `alternate_base(s)` of the "Human Variant" data type described in Fig. 5.2. This attribute may contain a very long base sequence in case of long insertions (such as the mutation of Exome Variant Server). These sequences are poorly suited to be stored in an EAV. If they are too long they will exceed the limit size (which currently is 8191 bytes) for a cell to be indexed with a BTREE index in Postgres. I did incur in this issue when populating the dataset for the performance tests. A possible solution, and the one that I have adopted, is indexing the column using the MD5 hash of the actual value. MD5 is 32-digit hexadecimal value and always fits the BTREE size requirements. This approach is safe enough but requires MD5 computation every time a new value is inserted in the catalogue and every time a search is performed. Even though it is extremely unlikely, MD5 uniqueness is not guaranteed as a consequence of the pigeonhole principle [6]. I did not take into consideration other collision-safe hashing strategies for my tests.
- multi-parameters pattern matching queries, where the xtens-query module can leverage the `jsonb` containment operator on the GIN-indexed *metadata* column. A good example is provided by the the first query shown in Sect. 5.4.2 (`@>`).

EAV is a valid alternative to the schemaless `jsonb` metadata format to perform range-only queries on numeric and data fields. On EAV catalogues, the query planner can leverage effectively the indexes if the selected range is selective enough, and only a small portion of the table must be retrieved. Using the JSON metadata format, it is not possible to leverage the GIN index for range-based queries on integer or float values. Threfore the query must access the values within the *metadata* column using the `jsonb` accessor operator, *first* convert them from JSON numerical to text, and *then* convert them from text to either integer or float. The first operation is time consuming, because the values are stored as JSON numericals in *metadata*, and at the moment there is no operator to directly cast a JSON numerical to SQL integer/float. This feature is under discussion in the Postgres community and it may be introduced in the next release. The double conversion is less costly for dates, because they are already stored as strings in JSON. The `jsonb` format is new to PostgreSQL, having been introduced in the 9.4 release (December 2014). New operators and functions will be introduced with new releases, increasing its applicability and performance, at the same time polishing the rough edges in the implementation. As a final consideration, the dataset that I have used in the performance tests is not optimal for an EAV catalogue: indeed, it falls in the third scenario exposed by Nadkarni (hybrid data types, see Sect. 2.6). Even though there are four different data types for the Data model—Clinical Situation, CGH Report, Microarray, Whole Genome Sequencing, and Human Variant—over the 99 % of the Data entities belong to the last type. However, because of the semi-structuredness of the variants as described by VCF format, they do not fit easily in a regular table: there might be more than one alternate base and associated gene. Further dedicated database engineering would

be required to provide a suitable schema, and this would violate my requirement of user-configurability of XTENS 2. Using the schemaless `jsonb` metadata storage, the user can configure all the data types described in the previous chapter, without any additional programming or database design. The database schema does not match the data structure: flexibility comes at a price in performance optimisation. But, as I have shown in Sect. 6.3, the cost is lower than adopting an EAV catalogue.

6.4 XTENS for Multidisciplinary Collaborations

XTENS 1.5, as adopted for institutional use at IGG, has allowed to integrate data produced by four different laboratories at IGG: Molecular Biology (sample biobanking, derivatives extraction, and Microarray), Anatomical Pathology (tissue characterisation), Oncology Facility (aCGH), and the Haematology and Oncology Clinical Department (clinical reports). In addition, the "ALK Research" of the San Martino-IST hospital provides data relative to ALK mutations in neuroblastoma, as described in Sect. 4.2.2. Overall, eleven people from five different departments have been granted access to the digital biobanks: a paediatrician, two oncologists, two "internal" biologists (i.e. belonging to the biobank) and five "external" biologists, and a bioinformatician. The five user groups have different security access level regulated by functions (as defined in 2.5.4).

References

1. JSON schema and hyper-schema. http://json-schema.org/ (2013). Accessed 09 December 2015
2. Joi: Object schema description language and validator for JavaScript objects. https://github.com/hapijs/joi (2012). Accessed 09 December 2015
3. PostgreSQL extension to scale out real-time reads and writes. https://github.com/citusdata/pg_shard (2015). Accessed 22 March 2015
4. Chiang, G-T., Clapham, P., Qi, G., Sale, K., Coates, G.: Implementing a genomic data management system using iRODS in the wellcome trust sanger institute. BMC Bioinform **12**(1), 361 (2011)
5. Dickmann, F., Falkner, J., Gunia, W., Hampe, J., Hausmann, M., Herrmann, A., Kepper, N., Knoch, T.A., Lauterbach, S., Lippert, J., et al.: Solutions for biomedical grid computing–case studies from the D-Grid project Services@ MediGRID. J. Comput. Sci. **3**(5), 280–297 (2012)
6. Wojciech, A.: Trybulec. Pigeon hole principle. J. Formaliz. Math. **2**(199), 0 (1990)

Chapter 7
Conclusions

I have designed a novel highly configurable JSON-based metadata model, to overcome data sharing limitations in biomedical research collaborations. The JSON-based model has been first tested on XTENS, a previously existing data repository written in Java language and has been successfully used in production to manage an Integrated Biobank at the IGG paediatric hospital in Genoa. The biobanking data repository manages different sample workflows—tissue and fluid banking, DNA and RNA extractions, aliquot deliveries—and keeps track of the genomic analysis that are subsequently performed on the nucleic acids, namely aCGH, microarray and targeted Sanger sequencing. As of March 2015 the XTENS biobank at IGG stored 2400 tissue and fluid sample from 1240 individual patients. The JSON metadata model demonstrated its good flexibility in describing a heterogeneous sources of information: quality control reports, deliveries, patient clinical records, and genomic profiles. Given its good applicability in managing an integrated biobank involving works from different laboratories, my next step was to extend the repository capabilities to support multi-centric data sharing. To this end I have developed a novel repository, XTENS 2, completely written in JavaScript code to provide an environment where the JSON model is a first-class citizen. XTENS 2 runs on a Node.js server and takes advantage of the `json`/`jsonb` data formats of the PostgreSQL database to store structured and semi-structured metadata as defined by the data model. XTENS 2 has been provided of various modules to ensure (i) transactional safety of data, sample and subject creation and update, (ii) distributed file system support for bulk data, and (iii) composition of complex multi-level queries based on parameters specified by the user (e.g. via graphical form). The flexible query composition is handled in a safe way, converting a JSON element into a sanitised prepared statement. XTENS 2 performances have been tested, building up a multi-level data hierarchy and populating the database with increasingly greatest numbers of subjects and associated datasets. I have used a set of 3 relevant queries to evaluate the retrieval time and adopted different index strategies to envision the better approach for optimising the database.

M. lzzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5_7

I have tested a first XTENS 2 prototype in an ongoing SEEG multi-centric project where external programs interact with the repository using a service-oriented RESTful interface. I have demonstrated the usefulness of XTENS 2 also in computational neuroscience because, using it, the project collaborators could (i) remotely input imaging (CT/MRI) and signal (SEEG) recordings, (ii) run the required processing tools, and (iii) output the relevant connectivity results on both individual and common anatomical spaces. The next objective is to migrate the Gaslini biobank to XTENS 2 using a multi-biobank environment, with the goal of bringing all the neuroblastoma biobanks under a single platform. This setup will provide a serious test scenario for management of Biomedical data that are at the same time heterogeneous *and* geographically distributed. A point that I have not discussed in detail here is the security and privacy of data. In XTENS 2, personal details are stored in a separate table as it is required by Italian and European regulations. The access to sensitive information is currently controlled by Sails.js policies.

7.1 Future Developments

XTENS 2 development and testing have been the main activities during my Ph.D. thesis and are in progress. Given the modular structure of XTENS 2, new functionalities can be added afterwards while the core system, beforehand tested, is used for pilot projects, as it happens for the SEEG multi-centric collaboration. There are some concepts from "old XTENS" that I have not yet adopted in XTENS 2: the most egregious is the Process-Event model model detailed in Sect. 2.5.4. While an event could be modelled as a data instance without metadata but a timestamp, I plan to add a fourth *model* (in the Sails.js sense), to describe studies and/or processes. In this way, a process would be a specialised data instance with its own metadata, and, possibly, and additional JSON structure that describe its workflow or protocol. In this way a process could consist of a sequence of sub-processes and events that produce data, just as it happens in "old XTENS". An important "old XTENS" administrative tool are the *functions* (see [1]), used to handle fine grained authorisation control to specific parts. They have the highest priority before the system is installed in production. Concerning the BIT digital biobank, I plan to upgrade the installation from XTENS 1.5 to XTENS 2 before the end of June 2015. I will have to develop ad hoc procedures to migrate the database from MySQL to Postgres, which will be the most onerous task. The BiolMol group has proposed to adopt the system to manage all the neuroblastoma biobanks and repositories across Italy.

7.2 Implications and Predictions

As the size of data produced by biomedical instruments increases and its complexity grows apace, more and more scientists and clinicians will require adaptive and schemaless solutions to manage their datasets. This has been the trend in the

Enterprise for the last five years and I see no reasons for clinical research to follow a different path. If possible, given the more fluid nature of life science, the demands of biomedical researchers will be even more pressing. I think that digital repositories—like XTENS 2—able to accommodate both standardised and fluid data represent a solution to tackle the Big Data challenge for biology clinical research, and life science as a whole.

Reference

1. Corradi, L., Porro, I., Schenone, A., Momeni, P., Ferrari, R., Nobili, F., Ferrara, M., Arnulfo, G., Fato, M.M.: A repository based on a dynamically extensible data model supporting multidisciplinary research in neuroscience. BMC Med. Inform. Decis. Mak. **12**(1), 115 (2012)

Appendix
XTENS 2 Database Schemas

Here I report the database schemas used in XTENS 2 as described in Chap. 5. The schema in Fig. A.1 represents the persistence layer for the UML class diagram shown in Fig. 5.1. The schema in Fig. A.2 refers to the EAv catalogue implementation used to execute the performance tests. The schemas have been drawn from the database testing installation using the graphic tool DbVisualizer Free 9.2.

M. lzzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5

Fig. A.1 XTENS 2 core database schema. The table dedicated to the repository administrations (i.e. `xtns_group`, `operator`, …) are not shown

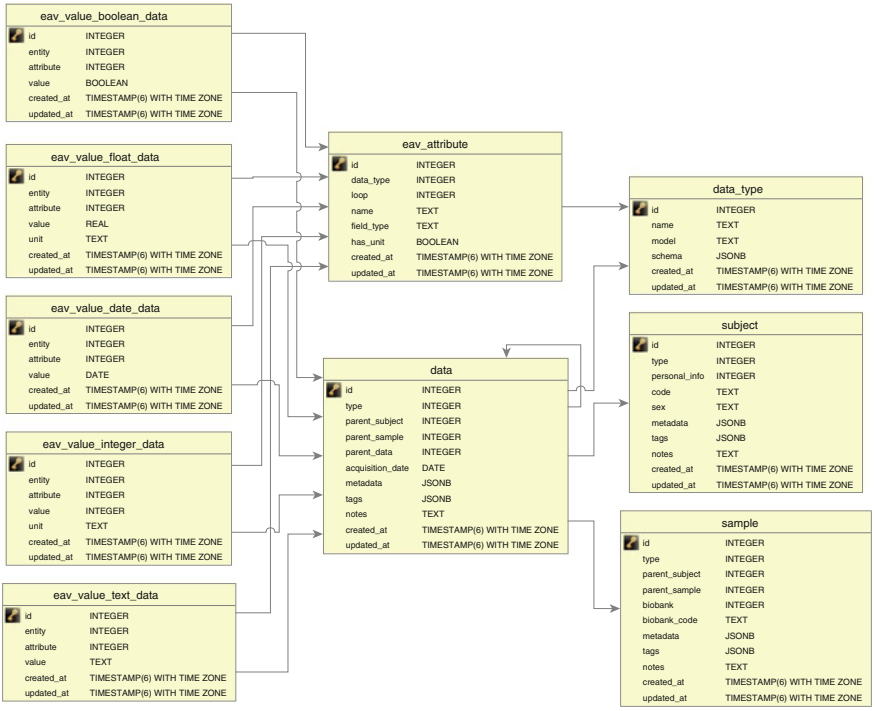

Fig. A.2 Database schema of the EAV architecture adopted for the performance tests detailed in Sect. 5.4. The schema is outlined only for the `data` table

Curriculum Vitae

Massimiliano Izzo

Date of birth: 22 February 1982
email: massimorgon@gmail.com
Citizenship: Italian
ORCID iD: 0000-0002-8100-6142

Professional Experience

- **Since January 2016**: Software research engineer ad Oxford e-Research Centre, University of Oxford.
- **Since January 2012–December 2015**: Research collaborator at the Molecular Biology Lab at the Giannina Gaslini paediatric hospital, Genoa. Main activities: design and development of a distributed digital biobank for paediatric tumours; development of data management solutions for clinical research projects.
- **June 2011–December 2011**: Research collaborator at Neuromed hospital, Pozzilli (IS), Italy. Main topics: diffusion tensor MRI, image processing.
- **November 2010–April 2011**: Internship at the cyclotron-PET radiopharmacy centre at the Department of Nuclear Medicine, S. Martino Hospital in Genoa. Main activities: synthesis and quality control on ^{18}F-based radiopharmaceuticals.
- **March 2010–June 2010**: Automation engineer at "Automazione e Servizi srl" in Loano (SV). Main activities: automation of the "Savona-S.Giuseppe" cable-car and home automation in villas and residential hotels.
- **January 2007–November 2009**: Ph.D. Student at the Italian Institute of Technology, Department of Robotics, Brain and Cognitive Science, in Genoa. Research activity on sensorimotor learning on humanoid robots. Study activity not concluded.
- **April 2006–September 2006**: University of Genoa, Department of Biophysics and Electronic Engineering. Research assistant, working on computational models of the visual cortex.

M. lzzo, *Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management*, Springer Theses,
DOI 10.1007/978-3-319-31241-5

Education

- Doctoral Degree in Bioengineering from the University of Genoa (2012–2015). Design and development of an innovative metadata model for heterogeneous data management in life sciences.
- Master's Degree in Bioengineering from the University of Genoa (2003–2005), with a final evaluation of 110/110 with honours (Magna cum Laude). Thesis title: "Models of cortical receptive fields for the joint perception of motion and depth".
- Bachelor's Degree in Biomedical Engineering from the University of Genoa (2000–2003), with a final evaluation of 110/110 with honours (Magna cum Laude). Thesis title: "Complex dynamics in prey-predator systems".
- International postgraduate course in entrepreneurship "Mediterraneo", under the patronage of the Italian Ministry of Foreign Affairs (June 2010–January 2011). The course was offered to graduate students coming from Italy and Northern African countries.
- High School Diploma from the Scientific High School "Leonardo da Vinci" in Genoa (1995–2000), with a final evaluation of 98/100.

Others

Languages: Italian (native speaker), English, Spanish.
Computer Literacy: Over three-year experience in object-oriented programming, web programming and data management in a distributed environment. Three-year experience in the design and development of RESTful APIs. Programming languages proficiency: JavaScript, Java, Python, C, C++.
Interests: Mythology and speculative fiction literature; creative writing and world-building; hard rock and heavy metal music; backpacking and couchsurfing.