Universidade do Minho
Escola de Engenharia

Ricardo André Pereira Freitas

Relational Databases Digital Preservation

Programa de Doutoramento em Informática
das Universidades do Minho, de Aveiro e do Porto

universidade de aveiro

Universidade do Minho

U.PORTO

Dezembro de 2012

Ricardo André Pereira Freitas **Relational Databases Digital Preservation**

UMinho|2012

Universidade do Minho
Escola de Engenharia

Ricardo André Pereira Freitas

Relational Databases Digital Preservation

Programa de Doutoramento em Informática das Universidades do Minho, de Aveiro e do Porto

universidade de aveiro

Universidade do Minho

U.PORTO

Trabalho realizado sob a orientação do
Professor Doutor José Carlos Ramalho

Dezembro de 2012

Sónia, especialmente para ti e para os nossos filhos António e André...

Para a minha Mãe, pelo seu amor e força que transmite, e para o meu Pai cuja sapiência influenciou para sempre o meu futuro!

Também para ti Daniela, serás sempre a minha maninha.

Agradecimentos

Agradeço em primeiro lugar à família, pois o seu apoio diário é sem dúvida muito importante quando se leva por diante um trabalho desta natureza...

A nível profissional são algumas as pessoas que merecem o meu agradecimento, o Eng. Rui Lima, meu mentor e amigo, o Dr. Miguel Guerreiro pelo importante apoio prestado, o Eng. Carlos Ribeiro, pelos seus extraordinários conhecimentos técnicos assim como vários outros colegas e colaboradores, especialmente do CIULF. Devo também um agradecimento global à Universidade Lusíada enquanto instituição que me acolheu, formou e apoio nas fases subsequentes da minha formação, e com a qual continuo a colaborar.

Na escola, nas diferentes universidades, em conferências por esse mundo fora, foram vários os professores e professoras que directa ou indirectamente contribuiram para que eu conseguisse atingir este objectivo, por isso, um agradecimento especial para todos eles.

Aos amigos de sempre, também um obrigado.

Finalmente o agradecimento ao meu orientador, o Professor José Carlos Ramalho, o seu enorme conhecimento e experiência foram determinantes no decorrer deste trabalho de 4 anos. Obrigado pelo seu apoio, pelas ideias e orientações que me transmitiu!

Ricardo André Pereira Freitas.

Abstract

With the expansion and growth of information technologies, much of human knowledge is now recorded on digital media. It began in the 20th century, it has been occurring continuously and it seems that there is no turning back. This paradigm brings scenarios where humans need mediators to understand digital information – computer platforms. These platforms are constantly changing and evolving and nothing can guarantee the continuity of access to digital artifacts in their absence. A new problem in the digital universe arises: Digital Preservation. There are huge volumes of information stored digitally and there are also a panoply of different classes, formats and types of digital objects. Our work addresses the problematic Digital Preservation and focuses on the logic and conceptual models within a specific class of digital objects: Relational Databases. This family of digital objects is used by organizations to record their data produced on daily basis by information systems at operational levels or others. This structures are complex and the relational databases software support may differ from one organization to another. It can be proprietary, free or open source.

Previously, a neutral format – Database Markup Language (DBML) – was adopted to pursue the goal of platform independence and to achieve standardization concerning the format in the digital preservation of relational databases. This format is able to describe both data and structure (logical model). The key strategies we are adopting are migration and normalization with refreshment. From our first approach, we evolved the work to address the preservation of relational databases and we focused on the conceptual model of the database. The conceptual model of the database corresponds to the ideas and concepts that in the basis of the designed and/or modeled database, conceived to support a certain information system. We are referring to the semantics of the database and considering it as an important preservation "property".

For the representation of this higher layer of abstraction present in databases we use an ontology based approach. At this higher abstraction level exists inherent Knowledge associated to the database semantics that we tentatively represent using Web Ontology Language (OWL). From the initial prototype, we developed a framework (supported by case studies) and establish a mapping algorithm for the conversion between the database and OWL. The ontology approach is adopted to formalize the knowledge associated to the conceptual model of the database and also a methodology to create an abstract representation of it. The system is based on the functional axes (ingestion, administration, dissemination and preservation) of the Open Archival Information System (OAIS) reference model and its information packages, where we include the two levels/layers of abstraction within the digital objects that are the subject of our research: Relational Databases.

The framework offers a set of web interfaces where it is possible to migrate a database into normalized and neutral formats (DBML + OWL) and perform some minor administration tasks on the repository. The system also enables the navigation or browsing through the database (concepts) without loosing technical details on the database relational model. The end consumers will have at their disposal a broad overview of the preserved object: a) the lower level data and structure of the relational database logical model and b) the higher level semantics and knowledge of the database conceptual model!

Considering the unpredicted future access to a preserved database content and structure, our preservation policy tries to capture the significant properties of databases that should enable the future interpretability and understanding of the digital object.

Resumo

Através do crescimento das tecnologias de informação, grande parte do conhecimento humano passou a ser armazenado em suportes digitais. Esta transformação iniciou-se no século XX, tem vido a ocorrer de forma contínua, e tudo indica que fora já ultrapado o "ponto-sem-retorno". Este novo paradigma implica cenários substancialmente diferentes, cenários estes onde os seres humanos necessitam de mediadores para compreender a informação digital – plataformas computacionais. Estas plataformas estão em constante evolução e não existe nada que nos possa garantir a continuidade de acesso aos artefactos digitais na sua ausência. Surge um novo problema associado ao mundo digital: Preservação Digital. Grandes quantidades de informação estão armazenadas digitalmente numa panóplia de diferentes classes, formatos e tipos. O nosso trabalho concentra-se na problemática da preservação digital, focando concretamente os modelos lógico e conceptual de uma classe específica de objectos digitais: as Bases de Dados Relacionais. Esta família de objectos digitais é amplamente usada pelas organizações para guardar os dados produzidos diariamente pelos seus sistemas de informação, tanto ao nível operacional como a outros níveis. Falamos de estruturas complexas em que os Sistemas Gestores de Bases de Dados que as suportam podem variar de organização para organização. Os sistemas podem ser proprietários, livres e ou de código aberto ("open source").

Inicialmente, um formato neutro – Database Markup Language (DBML) – foi adotado no sentido de garantir a independência de plataformas, e com o objectivo de conseguir establecer um formato normalizado para a preservação de bases de dados relacionais; isto tanto para os dados como para a estrutura (modelo lógico). As estratégias que adoptamos são a migração e normalização com refrescamento. A partir da abordagem inicial, evoluímos o nosso trabalho no que concerne à preservação digital de bases de dados relacionais, focando o estudo também no modelo conceptual da base de dados. O modelo conceptual corresponde às ideias e conceitos na base do desenho e/ou modelação de uma determinada base de dados, e concebido para dar suporte a um determinado cenário "real", i.e., a um determinado sistema de informação. Referimo-nos à semântica da base de dados considerando-a como uma importante ”propriedade” na preservação.

Para a representação desta camada de abstração mais elevada que está presente nas bases de dados, utilizamos uma abordadem baseda em ontologias. A este nível mais elevado de abstração existe informação e conhecimento intrínseco que estão associados à semântica da base de dados que se pretende representar através de Web Ontology Language (OWL). A partir do protótipo inicial, desenvolvemos uma plataforma aplicacional (suportada por casos de estudo) e establecemos um algoritmo de mapeamento para a conversão entre bases de dados e OWL. A abordagem através da ontologia foi adoptada para formalizar o conhecimento associado ao modelo conceptual da base de dados e também foi usada como uma metodologia para criar uma representação abstracta da base de dados. O sistema baseia-se nos eixos funcionais (ingestão, administração, disseminação e preservação) do modelo de referência Open Archival Information System (OAIS) assim como nos seus pacotes de informaçao (information packages) onde são incluídos dois níveis/camadas de abstracção, relativamente aos objectos digitais que são objecto de preservação neste estudo: Bases de Dados Relacionais.

O sistema (framework) fornece um conjunto de interfaces web, onde é possível migrar a base de dados para formatos neutros e normalizados (DBML + OWL), e permitem também executar algumas tarefas de administração do repositório. O sistema possibilita ainda a navegação e pesquisa pelas bases de dados (conceitos), sem que se perca aspectos técnicos associados ao modelo relacional das mesmas. Os consumidores finais têm ao seu dispor uma visão global do objecto preservado: a) a um nível inferior os dados e estrutura do modelo relacional lógico e b) a um nível mais elevado a semântica e conhecimento associado ao modelo conceptual da base de dados!

Considerando a imprevisibilidade no acesso futuro ao conteúdo e estrutura de bases de dados preservadas, a nosso política de preservação pretende capturar as propriedades significativas das bases de dados capazes de possibilitar futuramente a intrepertação e compreensão do objecto digital.

Contents

Acronyms

3NF Third Normal Form. 102

AIP Archival Information Package. 26–28, 32, 69, 70, 75, 87, 89, 95, 97, 98, 112, 123, 127, 130, 131

ANSI American National Standards Institute. 39

API Application Programming Interface. 70, 93

ARELDA ARchiving of ELectronic Data and Records. 48

BLOB Binary Large Object. 51

CCSDS Consultative Committee for Space Data Systems. 20, 125

CD Compact Disc. 12

CentOS Community Enterprise Operating System. 92

CLOB Character Large Object. 52

CML Chemical Markup Language. 43

DANS Data Archiving and Networked Services. 10, 36

DBML Database Markup Language. vii–x, 4–7, 36, 37, 43, 47, 52–54, 56, 59–61, 65, 68–72, 86, 87, 89, 93, 96–99, 101, 107, 109, 112, 114, 119–123, 129–135

DBMS Database Management System. 3, 6, 32, 35, 38–40, 64, 70, 71, 77, 86, 93, 98, 99, 101, 112, 126, 128, 129

DCL Data Control Language. 40

DDL Data Definition Language. 39

DELOS Network of Excellence on Digital Libraries. 10, 11

DIP Dissemination Information Package. 26–28, 32, 69, 70, 75, 87, 95, 123, 127, 130, 132

DML Data Manipulation Language. 40

DOM Document Object Model. 45

DSN Data Source Name. 101

DTD Document Type Definition. 44, 56, 60, 61

DVD Digital Versatile Disc. 12

FrepDB **F**ramework for **Re**lational **D**ata**B**ases **P**reservation. 89, 90, 93, 96, 123, 131, 133

HTML HyperText Markup Language. 42

HTTP Hypertext Transfer Protocol. 70, 92, 93

InSPECT Investigating Significant Properties of Electronic Content. 11

IS Information System. 6, 68, 126

ISO International Organization for Standardization. 21, 39

IT Information Technology. 5

MIXED Migration to Intermediate XML for Electronic Data. 36

OAIS Open Archival Information System. viii, x, 4, 6, 7, 11, 20–29, 32, 47, 49, 69, 72, 75, 89, 93, 95–97, 123, 125, 127, 131, 132

OCLC Online Computer Library Center. 47

ODBC Open Data Base Connectivity. 93, 112

OPF Open Planets Foundation. 9

OWL Web Ontology Language. vii, viii, x, 6, 7, 68, 78, 80–82, 84, 85, 87, 89, 95–99, 101, 102, 105–107, 109, 112, 114, 116, 119–123, 130–135

PDI Preservation Description Information. 25, 26, 95, 96, 99

PHP PHP: Hypertext Preprocessor. 70, 93, 97, 98, 111, 112, 116, 119, 123, 132

PI Persistent Identifiers. 74

PLANETS Preservtion and Long-term Access Through Networked Services. 9, 10, 36, 47, 48

Plato the Preservation Planning Tool. 11

PREMIS Preservation Metadata: Implementation Strategies. 11, 47

RAM Random Access Memory. 91, 119

RDB Relational Database. 3–7, 12, 30, 35–39, 41, 42, 47, 53, 61, 63–65, 67–69, 73, 75, 77, 78, 81, 85–87, 90, 96, 98, 99, 101, 112, 114, 125, 126, 128–136

RDF Resource Description Framework. 68, 78–80, 85–87, 116, 131

RLG Research Libraries Group. 47

RODA Repository of Authentic Digital Objects. 10, 36, 47, 53, 65, 129

SCAPE SCAlable Preservation Environments. 10

SFA Swiss Federal Archives. 36, 48

SGML Standard Generalized Markup Language. 42, 43

SIARD Software Independent Archival of Relational Databases. 5, 10, 35–37, 47–49, 51–53, 61, 64, 69, 129, 134, 135

SIP Submission Information Package. 26–28, 32, 69, 70, 75, 87, 95, 97, 98, 112, 123, 127, 130, 131

SMDL Standard Music Description Language. 43

SQL Structured Query Language. 6, 39, 64, 70, 71, 85, 90, 93, 98, 101, 128

SWRL Semantic Web Rule Language. 135

UML Unified Modeling Language. 74

URI Uniform Resource Identifier. 80

W3C World Wide Web Consortium. 42, 77, 85

WWW World Wide Web. 78

XHTML eXtensible Hypertext Markup Language. 43

XML eXtensible Markup Language. 36, 37, 42–45, 47–49, 52, 53, 56, 59–61, 63–65, 68, 71, 72, 78, 79, 86, 90, 97, 107, 114, 117, 120, 122, 128, 129, 131, 133, 135

XMLSchema eXtensible Markup Language Schema. 44, 49, 52, 56, 59–61, 64, 84, 129

XPath XML Path Language. 42, 44–46, 64, 98, 109, 114, 120

XSL eXtensible Stylesheet Language. 44

XSLFO eXtensible Stylesheet Language Formatting Objects. 44

XSLT eXtensible Stylesheet Language Transformations. 44

List of Figures

List of Tables

List of Code Blocks

Chapter 1

Introduction

The importance of information preservation, throughout the history of humanity, is something that has always followed us and that is fundamental to our own evolution. This concern remains and given the specifics of nowadays the preservation of information now focuses on digital artifacts.

In the current paradigm of information society, more than one hundred exabytes of data are used to support information systems worldwide [Manson, 2010]. The evolution of the hardware and software industry causes that progressively more of the intellectual and business information are stored in computer platforms. The main issue lies exactly within these platforms. If in the past there was no need of mediators to understand the analogical artifacts today, in order to understand digital objects, we depend on those mediators (computer platforms). Nothing can guarantee the continuity of access to digital artifacts in their absence [Lee, Slattery, Lu, Tang, and Mccrary, 2002]. A new problem in the digital universe arises: Digital Preservation.

Digital preservation is undoubtedly an important and promising area of research, contributing decisively to the preservation of the digital universe. By seeing the large increase in the use of digital tools in recent decades, it is possible to verify that the quantity and variety of file formats and associated tools, that are emerging, is huge. This new paradigm has to do with the evolution of technology and consequent improvement in computational performance, capacities and also advances in communication via computer networks [Lee, Slattery, Lu, Tang, and Mccrary, 2002]. Digital preservation is a need created by the constant and rapid change, within the universe of information technologies, either at a physical level - hardware - or at a logical and conceptual level - software. This constant evolution requires humans to be concerned with the preservation of digital artifacts, so that these remain accessible and intelligible over the time. From the present time and for the future we will depend on computer platforms (mediators) to understand the digital artifacts. We are talking about the importance of long-term preservation in order to ensure the continuity of access to digital information legacy.

Taking into consideration these various contingencies associated with the constant evolution of digital technologies, it is important to think about the strategies to be taken to preserve the digital objects.

1.1 Context

We can define digital preservation as a set of activities or processes responsible for ensuring continued access, in long-term scope, to information and other cultural heritage existing in digital formats [National Library of Australia, 2002]. It is important to state that a digital object is any and all kind of information object that can be represented by sequences of binary digits (a bitstream or multi-bitstreams) [Thibodeau, 2002]. This definition is broad enough to accommodate both the information that was born in a digital context (born-digital), and the digital information obtained from analog media (digitalized objects) [Ferreira, 2006]. Text documents, digital photos, vector diagrams, databases, video and audio sequences, virtual reality models, web pages, software applications, even social networks or games are just some examples of digital objects.

Accessing the information does not mean a simple access to the bits that all digital objects are made of, rather it means an access in such a way one can perceive the original object semantics. Although digital information can be exactly preserved in its original form by only copying (preserving) the bits, the issue appears when we notice the very fast evolution of those platforms (hardware and software) where the bits can be transformed into something human intelligible [Freitas, 2008]. Digital archives are complex structures that without the software and hardware – which they depend on – the human being, or others, will certainly be unable to experience or understand them [Ferreira, 2006].

Our work addresses this issue of Digital Preservation and focuses on a specific class of digital objects: Relational Databases (RDBs). These kinds of archives are important to several organizations (they can justify their activities and characterize the organization itself) and are virtually in the base of all dynamic content in the Web.

There are several families or classes of digital objects as we already mentioned a few of them. We dedicate our study to just one class of digital objects which is RDBs. Behind every information system must exist a place where the data is recorded and managed. This is the place that we normally call the database and the associated system that supports the management and treatment of that data. So databases are present all around the world and may contain enormous amounts of data of several associated data types. We focus on one model of databases which is the relational model that we normally address by the term "relational database".

The relational model for databases is the most popular and widely used globally. Edgar Frank Codd was a mathematician who theorized the model on his work of research at IBM. Based on a mathematical model, the relational models properties and features rely on the set theory [Codd, 1970]. Although this is just one model there are several software platforms that enable its implementation and operation. These are known has Database Management Systems (DBMSs). Some of these systems are proprietary while others are free or open source. It is not the purpose of our study to measure the amount of data kept in those systems, that would be impossible accurately.

1.1.1 What is Preservation?

Before talking about digital preservation of relational databases it is important to clarify the concept of preservation widely. Common sense indicates that to preserve means to "perpetuate" something so it can be "reachable" to humans when needed (through time), i.e., the archival and dissemination of certain artifacts, in a long term scope, even if those objects become obsolete or legacy.

Mankind deals with these concerns of preserving artifacts (non-digital) for several centuries, however the digital information that emerged in last century, imputed a different variable to the previous paradigm. The already mentioned mediators (computer platforms) are the new and evolving variable that emerged as digital objects (digital information) spread all over the world. Considering the unpredictable evolution of this variable (software and hardware platforms), preservation can focus on conceptual objects; those that are human readable and that should not be dependent of specific computer technologies.

1.1.2 Digital Preservation and Relational Databases

The nature of the digital object, under the process of preservation, determines possible strategies that must be adopted to achieve that goal. Considering the RDBs family of digital objects, we already, in previous work [Freitas and Ramalho, 2009], adopted an approach that combines two strategies and uses a third technique — migration and normalization with refreshment:

- Migration which is carried in order to transform the original database into the new format – Database Markup Language (DBML) [Jacinto, Librelotto, Ramalho, and Henriques, 2002];
- Normalization reduces the preservation spectrum to only one format;
- Refreshment consists on ensuring that the archive is using appropriate media to the hardware in usage/available throughout preservation [Freitas, 2008].

This previous approach deals with the preservation of the Data and Structure of the database, i.e., the preservation of the database logical model. We developed a prototype that separates the data from its specific database management environment DBML. The first prototype also follows the Open Archival Information

System (OAIS) [Consultative Committee for Space Data Systems, 2002] reference model and uses DBML neutral format for the representation of both data and structure (schema) of the database. This approach of migrating the database into a normalized format is also adopted by the Software Independent Archival of Relational Databases (SIARD) solution [SIARD, 2008]. Later on this thesis we analyze more deeply the preservation of RDBs related and previous work.

From our first approach, we evolved the work to address the preservation of relational databases and we focuses now on the conceptual model of the database. The main focus of this research it the long term access to the information inside a database.

1.2 Motivation

Huge quantities of information, business data, research data, cultural data, heritage data and others, are kept in relational databases. This fact suggests that efforts must be accomplished to guarantee the long-term future access to this body of data and knowledge. Platforms of hardware and software have evolved and will certainly continue evolve into new ones with different capacities, designs and architectures. We have assisted this evolution in the past and in recent decades. Such facts indicate some future problems accessing relational data maintained in platforms that may become obsolete. Nowadays some Information Technology (IT) technicians and engineers already struggle with some problems when accessing some legacy databases.

In order to avoid as much as possible these problems in the future, we research on establishing a preservation policy for RDBs, its data, structure and also its information, i.e., knowledge associated to the database semantics.

1.3 Aim of the Work

This work addresses the preservation of relational databases by focusing on the conceptual model of the database (the Information System (IS)). It is intended to raise the representation level of the database up to the conceptual model and preserve this representation. For the representation of this higher level of abstraction of the databases we use an ontology based approach. At this level there is an inherent Knowledge associated to the database semantics that we represent using Web Ontology Language (OWL) [Mcguinness and van Harmelen, 2004]. We developed a framework prototype (supported by case studies) and established an algorithm that enables the mapping process between the database and OWL.

We consolidate the first prototype into a framework that enables the integration of two levels of abstraction into a package of information for preservation. The framework offers a set of web interfaces where it is possible to migrate a database into normalized and neutral formats (DBML + OWL), preform some minor administration tasks on the repository and enable the navigation or browsing trough the database. The end consumers will have at their disposal a broad overview of the preserved object: a) the lower level data and structure of the relational database logical model and b) the higher level semantics and knowledge of the database conceptual model. This is done without loosing technical details on the database logical model which is preserved on the DBML. The framework also enables the database reconstruction/rebuilding from the Structured Query Language (SQL) code that it also generates for both structure and data.

In this project we created an open framework for ingestion, archival and dissemination of RDBs. The main idea is to separate the data from its specific database management environment (DBMS) and based on the OAIS reference model deploy a system for the preservation of RDBs, addressing its two top levels of abstraction: database logical and conceptual models. Levels that correspond to the conceptual and knowledge levels of the digital object shown in table 1.1.

Digital Object	Preservation Levels	Relational Database
Experienced Object	**Ontology**	**Conceptual Model**
Conceptual Object	**DBML**	**Logical Model**
Logical Object	–	Original Bitstream
Physical Object	–	Physical Media

Table 1.1: Preservation Policy

1.4 Thesis Structure

In the following chapter, we overview the problem of digital preservation, clarifying relevant concepts in this area such as: digital object, significant properties and preservation strategies. Also in chapter 2 we analyze the Open Archival Information System (OAIS) reference model.

Chapter 3 directly addresses the preservation of relational databases. We analyze Relational Databases (RDBs) their specificities and characteristic as well as approaches, related work and projects concerning the preservation of relational databases.

We introduce the main contribution of our research in chapter 4: ”a new dimension in relational databases preservation” throughout the rising of the abstraction level up to the conceptual model of the database. Here we contextualize the usage and relation between ontologies and RDBs.

Chapter 5 details our framework for the preservation of RDBs. We analyze the developed system and its mapping process from RDBs to Web Ontology Language (OWL) also evaluating the preformed tests and results through case studies.

Finally in the last chapter we will summarize the whole thesis and draw out the conclusions obtained from the research and development of this project. Also in the last chapter we state some possible future work.

Chapter 2

Digital Preservation

We can define Digital Preservation as a set of processes or activities that take place in order to preserve a certain object (digital), addressing its relevant properties and ensuring the continuity of access, interpretability and understandability over it for long or undefined periods of time.

The usage of digital media to store information in institutions, whether at corporate, governmental, educational or other levels, lead us to reflect on the importance of preserving that information. The boom in the use of information technologies only exacerbated this problem [Hodge, 2000]. In a few years a platform of hardware or software can become obsolete, with the risk of losing access to information stored there. These problems are the main concern in the research field of digital preservation.

There are a significant number of projects and solutions in the sphere of digital preservation. These frameworks normally tend to provide services in the form of interfaces to users where they can interact in order to perform the desired preservation actions: property identification, migrations, emulations, administration and integration as well as the content dissemination. The Preservtion and Long-term Access Through Networked Services (PLANETS)[1] project, for example, co-funded by the European Union, ended in 2010 and gave origin to the Open Planets Foundation (OPF)[2]. Scientific assets, cultural data, stored in digital formats, as well as

[1]http://www.planets-project.eu/
[2]http://www.openplanetsfoundation.org/

databases are a part of the project objectives in terms of preservation. The project mainly consists on the implementation of several services to address the several types and specificities of the different digital objects (the Software Independent Archival of Relational Databases (SIARD) solution [SIARD, 2008] was adopted by this project for databases preservation).

As general approaches to the problematic Digital Preservation we can enumerate some of the most relevant projects:

- Repository of Authentic Digital Objects (RODA) Project[3]. This project aims to implement a long-term preservation of digital content within the Portuguese administration [Ramalho, Ferreira, Faria, Castro, Barbedo, and Corujo, 2008].
- PLANETS Project. An European Union co-funded project aimed to build services and tools in order to ensure long-term access to cultural and scientific digital material [Farquhar and Hockx-Yu, 2007].
- SCAlable Preservation Environments (SCAPE) Project[4]. The SCAPE project is a large-scale EU-funded project to deal with standard workflows on heterogeneous collections, very large or complex digital objects and also with scalable architectures for monitoring and control of preservation actions and/or operations [King, Schmidt, Becker, and Schlarb, 2012].
- ExLibris Rosetta (ExLibris Group). A provider of products and solutions in the field digital libraries concerning the access and management of organizations digital content [ExLibris, 2012].
- Data Archiving and Networked Services (DANS). It is a Dutch institute that promotes the archival and access to research data, namely research datasets. It intends to provide a platform for sharing research data [DANS, 2012].
- Network of Excellence on Digital Libraries (DELOS). DELOS activities focused on the interoperability between traditional and digital libraries and also the creation of a Reference Model for Digital Library Management Systems

[3]http://roda.dgarq.gov.pt/
[4]http://www.scape-project.eu/

[DELOS, 2009]; the European project DL.org[5] has gave continuity to some of DELOS activities.

There are also some important projects and approaches concerning planning and identification of the objects significant properties such as the the Preservation Planning Tool (Plato)[6] [Becker, Ferreira, Kraxner, Rauber, Baptista, and Ramalho, 2008] or the Investigating Significant Properties of Electronic Content (InSPECT) Project[7]. It is also worthily of mention the Preservation Metadata: Implementation Strategies (PREMIS)[8] working group that produced an important report in the scope of preservation metadata (the main contribution consists on a Data Dictionary for Preservation Metadata).

The importance of digital data preservation determined the appearance of several research projects, working groups and scientific publications. However, and starting from the beginning we need to characterize the subject or artifact under study here: the digital object itself.

In this chapter we start to talk about the anatomy of the object and its digital chain of interpretations. These interpretations are related to the levels where the digital object is characterized – physical, logical and conceptual. Then an overview of the different perspectives and strategies on digital preservation, characterizing the state-of-the-art and enumerating the open issues in this field. The Open Archival Information System (OAIS) reference model recommendations, environment and structure are also detailed in this chapter. Before the end of the chapter the significant properties and its relevance in the process of digital preservation of digital objects are addressed.

2.1 Digital Object

Some distinction can be established between digital objects that were born in a digital context, and those that appear from the process of digitization: analog

[5]http://www.dlorg.eu/
[6]http://www.ifs.tuwien.ac.at/dp/plato/intro.html
[7]http://www.significantproperties.org.uk/
[8]http://www.loc.gov/standards/premis/

to digital. In a comprehensive way and encompassing both cases above, we can consider that a digital object is characterized by being represented by multiple bitstreams, i.e., by sequences of binary digits (zeros and ones).

We can question if the physical structure of the object (original system) is important, and if so, think about possible strategies for preservation at that level, e.g. ”technology preservation” (museums of technology) [Ferreira, 2006]. Nevertheless, the next layer — the logical structure or logical object—, which corresponds to the string of binary digits have different preservation strategies. The bitstream have a certain distribution that will define the format of the object, depending on the software that will interpret it. The interpretation by the software, of the logical object, provides the appearance of the conceptual object, that the human being is able to understand (interpret) and experiment. The strategy of preservation is related to the level of abstraction considered important for the preservation [Thibodeau, 2002]. From a human perspective one can say that what is important to preserve is the conceptual object (the one that the humans are able to interpret). Other strategies defend that what should be preserved is the original bitstream (logical object) or even the original media. Figure 2.1 shows the relationship between the different levels of abstraction (digital object) and the correspondent preservation formats adopted for Relational Databases (RDBs) in this research.

The sequences of binary digits, i.e., the multi-bitstreams are in the essence of all digital objects as we already mentioned.

First, we must discuss the physical media on which the bitstream is kept. For example, the technology used to store the binaries in hard disks is different than the technology used in Compact Discs (CDs) or Digital Versatile Discs (DVDs). The physical structure of the object is important in order to think about possible strategies for preservation at that level. In the next layer we have the logical structure or logical object (binary digits string), which has a certain distribution that will define the format of the object, depending on the software that will interpret it. This interpretation will provide the appearance of the conceptual object, that the human being is able to understand (interpret) and experiment (Fig. 2.1).

Observing this whole chain of interpretations and levels of abstraction, we notice

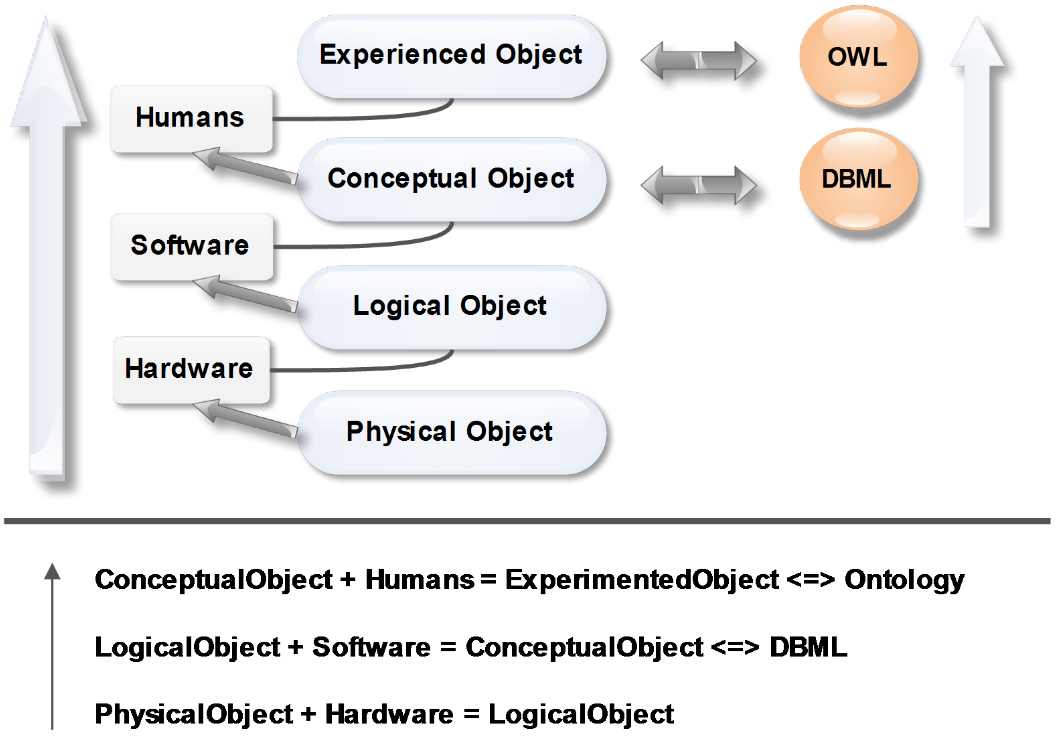

Experienced Object
Humans
Conceptual Object
Software
Logical Object
Hardware
Physical Object
OWL
DBML
ConceptualObject + Humans = ExperimentedObject <=> Ontology
LogicalObject + Software = ConceptualObject <=> DBML
PhysicalObject + Hardware = LogicalObject

Figure 2.1: Levels of Abstraction and Preservation Policy

that the digital preservation strategy will define the state of the object to be preserved (physical, logical, conceptual or even the experienced object). The strategy of preservation is related to the level of abstraction considered important for the preservation [Thibodeau, 2002]. From a human perspective one can say that what is important to preserve is the conceptual object (the one that the humans are able to interpret). Other strategies defend that what should be preserved is the original bitstream (logical object) or even the original media. The possible strategies are examined bellow with more detail.

At this stage it is important to beware of a) the relationships established between the levels of abstraction in the digital object and b) that the existence of such chain of relationships is crucial for preservation. If a breach or failure occurs in the chain, the digital object most certainly cease to be intelligible, which may result in the danger of losing the object forever [Ferreira, 2006].

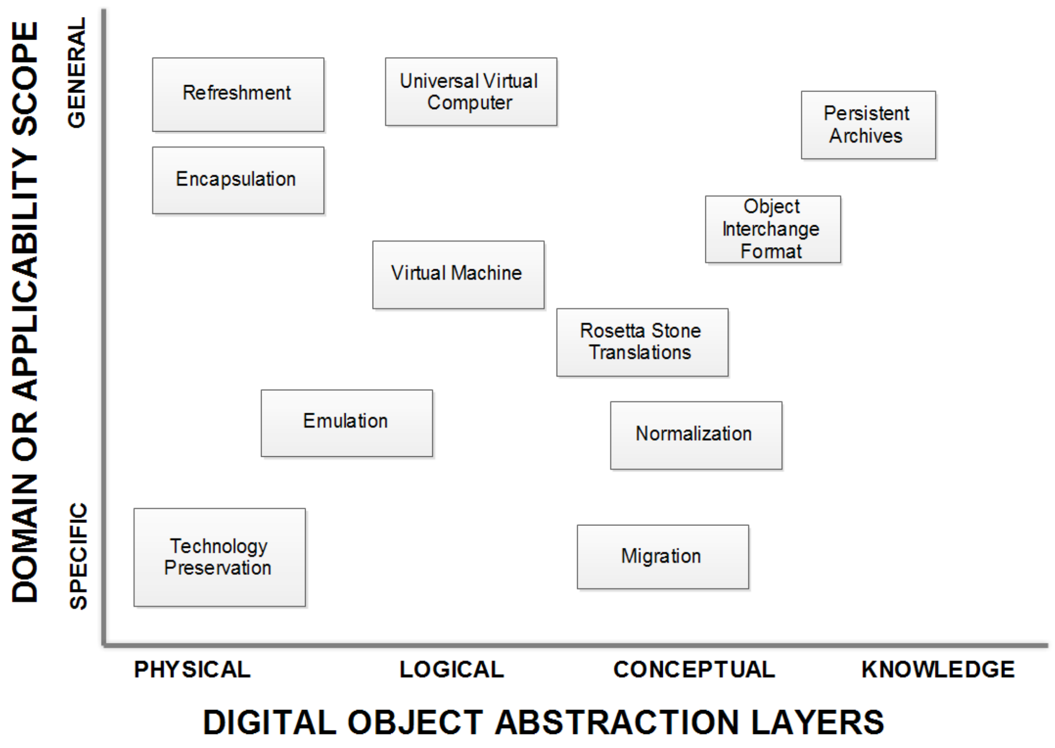


Figure 2.2: Digital Preservation Strategies or Methods

2.2 Digital Preservation Strategies

Different strategies, for digital preservation, are repeated by several researchers on their approaches to ensure that digital objects remain interpretable over time. Different concerns, with regard to what is essential to preserve (physical, logical or conceptual object), will influence the strategy to be adopted. Some authors defend that it is essential to ensure the authenticity and they argue that the only possible strategy is the ”Preservation of technology” (preservation of the digital object in its original form). Others argue that it is enough to preserve the conceptual object, what interacts with the real world, that is what human beings will experience.

The several strategies can be disposed in a graphical form that enables to see the relation between the preservation target levels of the digital object and the scope of the strategy in terms of applicability (Fig. 2.2) [Thibodeau, 2002].

There are several approaches which indicate that this problematic is not consensus and that the adopted strategy might depend of the digital object under preservation. Accordingly, we discuss some strategies for digital preservation.

2.2.1 Technology Preservation

One of the proposed strategies for digital preservation is technology preservation. This approach seeks to preserve the technological environment as it exists or existed. This strategy involves the conservation and maintenance of all hardware and software that characterized and constituted the original digital objects [Lee, Slattery, Lu, Tang, and Mccrary, 2002].

This technique gives special emphasis to the preservation of the physical (and logical) object. The supporters of this approach claim that this is the only way to accurately represent/recreate the original digital object (authenticity). It turns out that in terms of scale, this strategy is very complicated to implement, at least for all types of digital objects. Let's think of what must be done to take forward this strategy: it would be necessary to find physical places and a specific location to store the entire legacy of technology (hardware and software). Basically, we will have to build museums of technology. The cost involved in this type of strategy is its main disadvantage, however it is possible to imagine that for some types of digital artifacts it would be interesting to pursue digital preservation under this approach.

2.2.2 Refreshment

This is another technique used in the context of digital preservation. The Refreshment consists on the migration of information in a particular physical medium, which possibly may become obsolete, for a more actual/current one. As an example of this practice, we have copies of information media that are becoming obsolete, such as floppy disks, and we migrate them to more current media such as CDs. With this technique we can at least ensure that the information remains accessible in terms of hardware. Of course that it will be necessary to combine this approach with other strategies to achieve an effective preservation [Besser, 2001].

2.2.3 Emulation

Emulation is an important strategy since it is already implemented with some impact in the real world. Basically we will have a software piece able to emulate other software. However this software, called emulator, will certainly suffer himself of obsolescence (disadvantage).

So what is an emulator? A software that tries to recreate a technological context so that a given application can run on it, thus recreating the original environment. This strategy assumes primary importance when we talk about applications of software (executable applications) with dynamic and interactive aspects (e.g. games). In these cases it assumes special advantages, since it is capable of achieving high levels of preservation of the properties and characteristics of the original digital object [Lee, Slattery, Lu, Tang, and Mccrary, 2002].

Some examples that use this strategy are the emulators of console games. Old games that were originally developed to run on a specific console/platform, are now able to be played even in the absence of that physical console. That is possible through the use of an emulator.

2.2.4 Migration

Here it is another important strategy used for ensuring digital preservation. This strategy consists on the conversion of existing information in a particular format or in a certain platform software/hardware into other formats/platforms more contemporaneous. The idea is that the information remains always in a state considered actual and able to be interpreted by existing technologies [Lee, Slattery, Lu, Tang, and Mccrary, 2002]. Migration is a strategy widely used for digital preservation and with given proves in the real world. However, this strategy cannot ultimately be the solution for this problem (from time to time it will be necessary to re-migrate to new technological systems more updated).

There are different types of migration, for instance we can have migration into analog media which means that the digital objects will be converted into analog artifacts. It is true that this approach will only fit digital objects whose representation is close to analog formats such as images, text documents and others. The aim is to preserve the analog artifact since in this state, the object, is directly intelligible to humans.

Version update is another migration strategy widely used. Consists mainly in migrating imported digital objects into updated versions [Thibodeau, 2002] of the product of software that reads them. However, the version update strategy may have in itself some problems, since we are dependent on the company that owns the product. We know that many times when a migration into more current version occurs it does not mean that all aspects and attributes of a document/object are incorporated into the new version. Given these discontinuities in software, it is necessary to sometimes carry the documents into competing formats so that the objects are not dependent on a certain manufacturer.

Migration on Request

Migration on request is another approach related to migration in the scope of digital preservation. This is characterized as a type of migration that always uses the original format to operate the conversions to the desired new formats. When a migration into a new format is need, the conversion process uses always the original format of the object, therefore not propagating possible migrations errors [Mellor, Wheatley, and Sergeant, 2002]. Often when we have a migration on a digital object, what happens is that it may start to miss its features or attributes after a series of migrations; it will no longer be possible to represent the original object acceptably. The migration on request has a great advantage here, since migration is always carried out from the original object. It is also important to note that once built the converters that decode the original object, this information may subsequently be used when migration is necessary.

Distributed Migration

This is another migration strategy slightly different. We can imagine a set of converters over the Internet working somehow like web services that enable the conversion of different formats [Hunter and Choudhury, 2004].

The main potential of this approach is the existence of different converters and various routes in terms of conversion to certain formats. By doing this it is possible to ensure that if perchance some converter fails or something becomes obsolete we will have other paths (web services orchestration) to use in order to achieve the migration to desired formats. We would have a global network of converters that can contribute greatly to the success of preservation. The amount of information to be handled might be an issue; if the information has a very high volume, we can have problems since we are talking about the Internet and therefore about the bandwidth to transfer information and its inherent costs.

2.2.5 Normalization

This approach intends to find formats that are widely known and widely used like open international standards. Thus one can migrate the objects into these formats, so that preservation strategies only worry with a limited number of file formats to preserve. When ingesting objects in a repository, all objects (e.g. images) are converted to a unique format. In this way the objects will be easier to preserve because instead of being necessary to preserve a large number of formats, the preservation will focus only in a particular format [Ramalho, Ferreira, Faria, and Castro, 2007]. Therefore, the choice of the normalized format is very important since it should be approved by its designated community and also include all relevant aspects of a particular type of digital artifacts.

2.2.6 Encapsulation

It is a strategy that can be adopted when a digital object is stored for a long period of time without being changed by anyone or anything. In this situation a problem may arise since migration might have a very high cost or even become completely unworkable. To address the problems related to preservation in such situations, we might use the encapsulation approach. This strategy is based on keeping the objects unaltered together with as much information as we can gather (metadata) about the object. Thus, in the future when the artifacts are requested, the metadata will enable the construction of the necessary converters.

2.2.7 Rosetta Stone Translation

The Rosetta Stone is a piece of granite that become famous because it conducted researchers to decode the Egyptian hieroglyphs. The stone, with inscriptions in three different languages (Egyptian hieroglyphic, Egyptian cursive and classical Greek), was used to establish a parallelism between the classical Greek (a known language) and ancient Egyptian dialects [Ferreira, 2009].

Since this unpredicted method proven to be valid concerning the long term preservation of artifacts (analogical), the idea is to somehow apply this strategy also to digital artifacts [Heminger and Kelley, 2004]. A possible application of this method can consist, for instance, in the archival of a certain bit streams (samples) along with their corespondent texts for a specific software (word processor). This might enable the future interpretability of different texts documents/objects, without specific preservation strategies implemented, if those texts are in the same software format as the previous ones [Thibodeau, 2002].

2.2.8 Persistent Archives

This methodology was firstly introduced in [Moore, Baru, Rajasekar, Ludaescher, Marciano, Wan, Schroeder, and Gupta, 2000] and defends the preservation of entities and associated concepts that do not depend on both hardware or software. Moreover, it combines other several approaches, such as normalization and migration, but it is pointed to conceptual notions like context or semantics. Briefly, this approach, suggests the rising of the abstraction level in terms of the preserved digital object.

2.2.9 Open Issues

Somehow, there is still a path to cross in several aspects related to digital preservation. Taking into account the current state of information technologies, as well as the characteristics under which a digital object is presented, these are considered the main valid and usable strategies that try to achieve digital preservation. Migration is one of the strategies that undoubtedly has a role of relevance, however we can not consider it the ultimate answer. Only through a broad vision of the problem we can walk in the right direction [Eager, 2003]. Depending on the digital objects to be preserved appropriate strategies should be chosen [Waters, 2002]. It is also expected that research in areas related to digital preservation will follow the evolution of the digital technologies.

2.3 OAIS Reference Model

In January 2002, a document was approved by the management council of the Consultative Committee for Space Data Systems (CCSDS). The document represented the agreement carried out by the participating CCSDS member agencies.

The document consists of a series of technical recommendations in order to establish a global consensus regarding the preservation of digital information permanently or for long periods of time (long term) - "This document is a technical Recommendation for use in Developing Broad consensus on what is required for an archive to Provide permanent, or indefinite long-term, preservation of digital information"[Consultative Committee for Space Data Systems, 2002]. Generally, it defines the requirements necessary to obtain an Open Archival Information System (OAIS). Commonly we refer to the OAIS by saying that we are talking about the OAIS reference model, and that is exactly what it is: a reference model that establishes a series of recommendations, strategies to follow and guidelines to be taken in consideration. This reference model also identifies components and agents necessary to establish an archive for the preservation of digital information. Broadly speaking, the OAIS reference model, is based on three fundamental operations: information ingestion, archive administration and information dissemination. In this section, the components of a system guided by the OAIS reference model and also how the system should behave, will be described in some detail. All the features and all the recommendations in the reference model point towards the preservation of digital information and also to the requirement of maintaining that information accessible and understandable for long periods of time. The recommendations for its development must be kept in open forums and available to all.

The purpose of the OAIS reference model is to define an ISO – International Organization for Standardization (ISO)[9] – which aims to preserve digital information for long or undefined periods of time. This may implicate the occurrence of technological changes, new formats, different documents or changes at the community of interest level. The model itself consists on a set, or organization, of people and systems that accept the responsibility of maintain digital information "always" available for their community of interest (also called de designated community). The OAIS reference model introduces the necessary tools for understanding and to be aware of concepts related to digital archives and long-term preservation. The model also provides the necessary concepts for organizations that have nothing to do with preservation, to participate actively in this process. Issues related to strategies, techniques, digital preservation and changes over time of the adopted models for preservation are also addressed by the OAIS.

The model also provides a basis for addressing issues related to the preservation of information not digital. This reference model also participates in the identification, orientation and production of related models – *OAIS related standards*. Issues related to the exchange of information between archives are also addressed by the model. The role that the software has in digital preservation is also a concern within this referential.

2.3.1 The OAIS model and Concepts

The OAIS model may be applied to any type of archive, however it is geared towards organizations that need to see their information preserved for long periods of time. The model will also be of great interest to those who wish to extract information

[9]http://www.iso.org

from archives or repositories guided by the same model (OAIS). The rapid growth in the world of computing and communications leaded to an increase in digital transactions between organizations of all kind. The information is increasingly digital and uses digital paths and work flows at the expense of traditional means such as paper. Some organizations already recognize the importance of implementing policies to preserve their digital data, nevertheless the organizations themselves were pushed into this new paradigm without paying attention to the problem they were creating. Probably, several organizations never thought about the possibility of facing this problem of preserving digital information.

The OAIS model has a set of requirements and recommendations that organizations should follow to mitigate or address the issue of preserving digital information for long periods of time (long-term or permanently). It is known that digital information can easily be lost or corrupted. At production of digital information time, we have privileged access to data and on how that information is produced (metadata). So if organizations have an active role when producing their own data and start to consider the preservation of it, they will certainly have future benefits. Due to various factors, the production of digital information increases every day, which means that in many cases the roles of the producer of information and the responsible for archiving that information, are merged into the same individual. The digital preservation concerns are often relegated to second plan, increasing costs that will only be felt in the future. Designers and system developers should take into consideration the extreme importance of documenting the generated information, yet it is easy to realize that this often collides with market objectives for rapid production and dissemination of products to their consumers [Consultative Committee for Space Data Systems, 2002].

The OAIS reference model emerges to provide a knowledge base and a sense about what should we consider in order to obtain or achieve digital preservation. This model leads us towards the creation of an open archive information system as well as to create the necessary functions to access that same information. The system can be updated with information on a regular basis or not. It may have to provide more simple or more complex data depending on the requests, but it is crucial to guarantee the access to preserved information by the relevant community

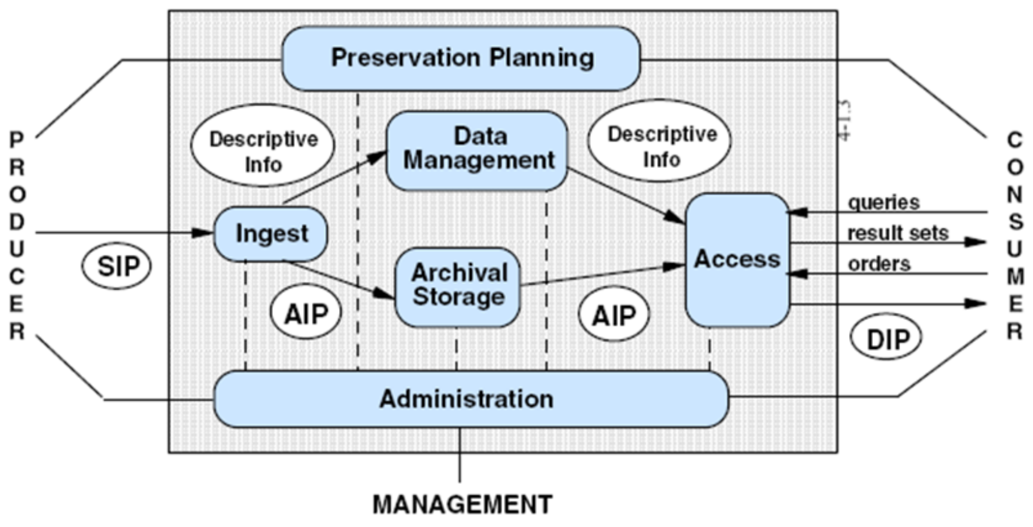


Figure 2.3: **OAIS** Functional Entities.

of interest, and the OAIS strongly stands on this premise.

2.3.2 OAIS Environment, Information and Representation

An OAIS archive system environment seen from the outside, has the producer of information for archival ingestion (Producers), the responsible for the administration of the system (Management) and, finally, the one that consumes the information (Consumers) or those that extract information from the system. The producer provides information to the system with the objective of preserving it. Those with maintenance and administration responsibilities carry out policies and activities to promote the ingested information conservation or/and preservation. The consumer interacts with the system in order to extract information of interest. The community of interest is a particular group of consumers that should understand the preserved information. Figure 2.3[10] shows a system conceptual design based on the OAIS reference model.

Different scenarios are possible where the OAIS system may differ from this explicit environment (conceptual design). There could exist relationships at various

[10]Courtesy of Consultative Committee for Space Data Systems. "Reference Model for an Open Archival Information System (OAIS) - Blue Book," National Aeronautics and Space Administration, Washington, 2002 [Consultative Committee for Space Data Systems, 2002]

levels between archives. A particular OAIS can have the role of producer to another OAIS; for example, when the responsibility to preserve certain information is to deliver information to the other OAIS archive. The opposite can also occur, when an OAIS system assumes the role of a consumer of another system. These interactions between archives should stand on formal basis to ensure communication even when they suffer changes in specifications.

In an OAIS system it is necessary to identify and characterize the information to be preserved. What is information? What should be considered as information? The information is only useful if it can be understood. To understand the information stored in the system we must have a knowledge base. This knowledge base will allow the understanding and interpretation of the archived information. Without this base, and because we talk about preservation for long periods of time, one may not be able to understand the preserved data. So it is necessary to create a "Representation Information" which will be information to understand preserved information – a dictionary to understand a new language, for example. The system must be concerned with the preservation of data object (data object), and also with its metadata to understand the data, so that the Information Object can be obtained. The information object should enable the interpretation of the data and therefore provide knowledge. However, a complication emerges from this specification: the "Representation Information" component can be recursive; this can often lead to the creation of a network of such components.

It important to have such representations in the system so that one might extract intelligible information from it. The knowledge base of the Community of Interest or designated community, should be considered by the OAIS system in order to incorporate in the archive, along with the data, that same knowledge base mentioned previously. The definition of the knowledge base and the correspondent representation will have greater or lesser impact when someone from the community tries to understand the archived data. Updates to the representation are possible so that the archive remains intelligible.

The access to the information object will be provided by a certain software, so it is important that the software does not assume a key role, in the process, because it would become more complicated to preserve the software than the information itself. Issues about the presentation of the information (how it is presented), may also be relevant; however the OAIS reference model is more concerned with the information itself and not with the way it is presented. Concerning all this issues it is clearly that we are consolidating a system that will be more and more complex. In cases where the software is proprietary it may me impossible to satisfy all these requirements.

2.3.3 Information Packages

Each information transmission in an OAIS compliant system, either at submission or at dissemination level, occur as a transaction or as a series of discrete operations. This establishes the concept of the Information Package, associated with these operations. Conceptually an information package consists of two types of information: the information content – the subject (including the Information Object and the Representation Information) that was originally intended to preserve –, and Preservation Description Information (PDI). The Information Package will encapsulate these two types of information and will be seen and detectable as result of the Descriptive Information referring to the entire Information Package (Fig. 2.4[11]).

The PDI must exist to identify the Content Information and the environment in which it was created. The PDI is divided into four parts: Provenance, Context, Reference and Fixity. The Provenance give us the origin or source of the Content Information as well its history and trajectory. Context provides information about how the Content Information may be related to information outside the Information Package, and may also explain the purpose of that content creation. The Reference uniquely identifies the Content Information. Fixity works as a shield so that the information content is not altered without proper documentation of those actions. The information associated with the Information Package needs to somehow relate the information content with the PDI. By its turn the descriptive information helps on searching and discovering a certain information content. Information Packages

[11]Courtesy of Consultative Committee for Space Data Systems. "Reference Model for an Open Archival Information System (OAIS) - Blue Book," National Aeronautics and Space Administration, Washington, 2002 [Consultative Committee for Space Data Systems, 2002]

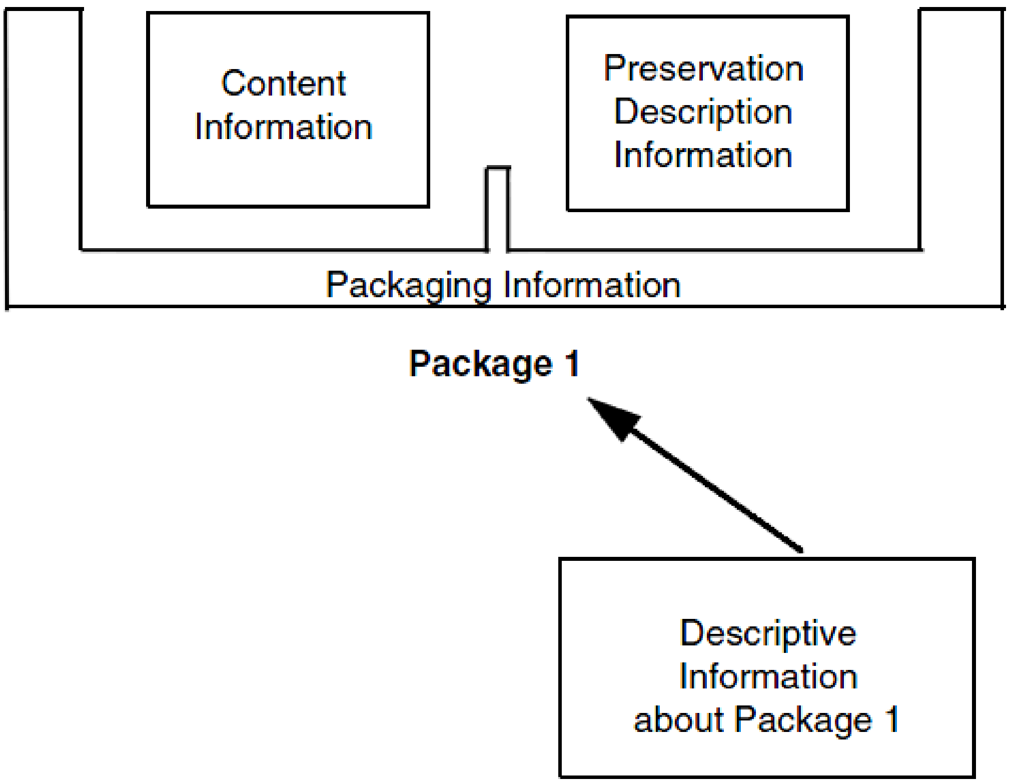

Content Information
Preservation Description Information
Packaging Information
Package 1
Descriptive Information about Package 1

Figure 2.4: Concept and relationships in the Information Package

may take three different forms or three variations: Submission Information Package (SIP), Archival Information Package (AIP) and Dissemination Information Package (DIP). These three forms must meet the specific characteristics of each OAIS process.

The submitted information package may not meet all the requirements for a correct preservation archive, or the consumer may obtain a package that does not contain all the representation information or all the PDI needed. In order to mitigate these and other problems there are some rules to follow. Between the Producer and the system it is negotiated, in detail, the composition of the submitted package. It may take several SIPs to form an AIP, as well as information contained in a SIP may be disperse in several AIPs. An AIP is the actual package that will be preserved "in" the OAIS, i.e., inside the repository or the archive. Such packages must meet the standards of an OAIS and should, for example, contain a set of PDI associated with the Content Information. An AIP can also contain a set of others AIPs. On the other extreme is the dissemination. The response by the system to a certain request for preserved data is achieved via DIPs. These packages can be formed by multiple AIPs. On this dissemination stage, the DIPs, satisfying consumer requirements, are delivered. The preserved AIP may suffer transformations during this phase in order to meet the dissemination necessities and specifications.

2.3.4 Interactions and Responsibilities

To end this OAIS analysis, we will now talk about the interactions between agents or entities in the system and the flow of information in several high-level operations. As we have seen above the producer has to deliver a SIP to the OAIS, the OAIS transforms all the SIPs in AIPs packages that are actually stored, and finally a consumer will make requests to the system that responds in the form of DIPs. The system administration, is responsible for defining the scope, conditions and structure of the OAIS. The administration component covers all the activities on the archive (OAIS) and it also determines the groups of Producers and Consumers covered by the system. At this level, some specific administration activities are: project financing, definition of the general following lines, resource management, conflict management, review of objectives, etc... The administration is also concerned about the path of the project in order to follow standards and others, as well as the entire sphere surrounding the project. At a more technical level the system administration is concerned with monitoring the state of obsolescence of objects stored. This is performed by conducting audits to the system and triggering migration actions (preservation strategy adopted) when necessary.

There is a relationship between the producer and the OAIS in which the main requirement states that the system (OAIS) should preserve the production of data by the former. A data submission contract, or Submission Agreement is defined. In the Submission Agreement one or more Data Submission Sessions are established. The sessions may occur from time to time and may contain one or more SIPs. A negotiation between the Producer and the OAIS will establish a model that defines the contents of the Data Submission Sessions: Content Information, PDI, Information Package, Descriptive Information; all of this is identified through the established model. The Submission Agreement also defines the frequency of sessions. Each submitted information package, i.e., each SIP must meet the requirements of the system. However, in some situations it may occur that it will take several SIPs to provide an AIP, or the information contained in a given SIP may be dispersed in several AIPs. The verification protocols, by the OAIS, concerning the accordance with the data submission sessions are also to be defined in the Submission Agreement .

The OAIS reference model advocates that there should be a series of interactions between the system and consumers to meet the information needs of the latter. These interactions, may be in the form of questions launched to the system, literature searches, searches in the catalog, ordering and consults about the status of the requests. In parallel to the interaction that exists between the producer and the system, here there is also set an Order Agreement between the system and a Consumer. It is this contract or agreement that indicates the frequency of the Data Dissemination Sessions. This Agreement will identify the AIPs of interest and how these will be transformed into DIPs to be integrated in the Data Dissemination Session. Two types of requests are available to the Consumer: Event Based Order and Adhoc Order.

In cases where an Adhoc Order is taken, the Order Agreement is based on information allegedly present in the system. The Consumer may need to search for information of interest in the archive. It is established a search session that can be based on descriptive information or even in information that is inside the AIPs. Searches can be interactive and once the desired information is found, i.e., the interested AIPs, the Order Agreement can be defined. The Agreement identifies the AIPs of interest and later defines the DIPs that have to be acquired from the system. If all the above premises are verified an Adhoc Order is established. If the AIPs of interest are not in the archive an Event Based Order is deployed.

The Event Based Orders also establish Order Agreements, but in this case, for information that is expected to be received by the system, triggering future events. The event may have some periodicity or just be a single triggered event by some particular information ingested into the archive. The Order Agreement also informs the system when it should trigger a new Data Dissemination Session based on a particular event. The Agreement will contain the criteria for the selection of information to be included in the Data Dissemination Session.

Among the responsibilities that an OAIS assumes, there are negotiation with producers and consumers in order to accept and disseminate correct information packages. The system must obtain the necessary control to ensures the long-term preservation of the information packages ingested. The system should also be able to determine and be aware of the acquaintance by the community or communities of interest in terms of the preserved information. The preserved data should be understood, and maintain interpretability, independently of the communities of interest. The OAIS recommendations also point out policies and documented procedures to ensure that the information will be preserved in any situation over unplanned events, or against any contingency [Lavoie, 2004] [Consultative Committee for Space Data Systems, 2002].

2.4 Significant Properties

Digital objects have several associated aspects (characteristics or properties) that we should consider whether or not to preserve. The designated community plays an important role and helps to define

> *"The characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record"*[Wilson, 2007].

The selection and identification of these characteristics already divides the scientific community, but there is also a discussion surrounding the terms that should be used to address those aspects of the digital objects that should be preserved. Some will defend the terms "significant properties", others use "significant characteristics" and so one [Dappert and Farquhar, 2009].

To elucidate the reader we can take a look to the example of a text-document. Considering for example a document containing only text (without images for instance) it is worthily to question if it would by enough to keep only the text. A text document normally has certain number of pages, specific letter fonts, pages margins and other properties. So, should the number of pages of the original document be preserved? Should we preserve the fonts or not? These questions are directly related to the topic of significant properties or characteristics, present in digital objects, that may or may not be addressed by the strategy or policy of preservation adopted.

Angela Dappert and Adam Farquhar state that the "Significance Is in the Eye of the Stakeholder" [Dappert and Farquhar, 2009]. However this approach can lead to some confusion because the stakeholder vision of the problem may not always correspond to the significant properties identified by a community of interest (designated community). The discussion is open! The perspective of those who intend to develop an action of preservation, over a certain artifact, will restrain/determine the significance of the properties. However, in order to be rigorous and address the problem in its whole essence, the analysis and determination of the significant properties cannot depend on the probable ambiguity of perspectives. Since our study already restrain the family of digital objects addressed for digital preservation — RDBs —, there must be a standard that determines the main characteristics/properties to ensure preservation within this class of objects.

Some relevant issues stand out: should the environment be preserved? should we question everything? is it feasible? and the answer is that we must be focused; a scientific/specific method must be developed/followed to address these questions. When we have an artifact for preservation — the preservation object or the digital object — in our case relational databases, we could question the effects of cutting/extracting the object from its original context. Can we do this even when we are referring to objects that are platform (hardware/software) dependent? The interaction between the source of the digital object and the platform results on a conceptual object that can be different if the platform changes [Wilson, 2007]. The output can be different. The important is that the essential parts purport what the object where made for. So either the source or the platform can be altered if what is essential is obtained maintain the meaning of the digital object over time. The "essential" here is translate into significant properties. In order to be rigorous three things must be defined:

- the artifact for preservation;
- what are all the implications that should be preserved to guarantee the preservation;
- what actions should be taken.

It is important to establish the properties that will ensure that in the future, when the preserved object is required (e.g. dissemination), the archive will be able to provide the object maintaining an acceptable level of originality: the object should be able to act as an evidence of what it was! The digital artifact must be exhaustively analyzed in order to be completely characterized. Then a consensus must be establish over what should be preserved.

As we will see later on this thesis, our preservation object contains mainly operational data of some kind, structure and semantic that can lead us to the understandability of the data itself. This characterization to the object is concerned with two different levels of abstraction. The higher abstraction level that focuses on the database conceptual model (database semantics), and the a lower abstraction level, seen as the database logical model (structure and data).

2.5 Summary

The main topics related to digital preservation were analyzed in this chapter. A broad overview about the technics and strategies currently used and addressed by several authors concerning the preservation of digital objects is provided. To do so we firstly characterize the digital object and its layers of abstractions along with its chain of relationships. There are different types of digital objects with specific characteristics, but it is possible for all of them to establish their physical, logical and conceptual levels. Also, like in all objects, digital or not, there is also possible to apprehend knowledge from the objects experimentation (from our experienced with the object). Synthesizing, the digital object can assume 4 variations: i) the physical object ii) the logical object iii) the conceptual object and iv) the experienced object. They relate to each other by having a) hardware to interpreter the physical (to logical) object; b) the software to transform a logical into a conceptual object; and c) humans that by experimenting the conceptual object produce the experienced or knowledge object.

The main strategies referenced in the literature and studied under the topic of preservation of digital objects were presented in this chapter. For each one of them, their main characteristics and there scope was analyzed. The strategy or strategies to be adopted depend on the digital object under preservation and on the level of abstraction intended to preserve. There is no complete solution (considering the huge diversity of formats) and in some situation, to specific families or classes of digital objects, combined strategies could be adopted if needed.

A detailed analysis about the Open Archival Information System (OAIS) [Consultative Committee for Space Data Systems, 2002] reference model is given. The OAIS reference model does not impose rigidity with regard to implementation, but rather defines a series of recommendations. This model of reference is concerned about a number of issues related to digital preservation: the process of information ingestion into the system, the information storage as well as its administration and preservation, and finally information access and dissemination. As it was previously mentioned, the OAIS model does not impose any computer platforms, development language, Database Management System (DBMS), interfaces, i.e., does not condition the development of the system at the technological level involved. Instead the model acts as a guide for those who wish to develop digital archives.

Three information packages are the base of the archival process: Submission Information Package (SIP), Archival Information Package (AIP) and Dissemination Information Package (DIP). Before ingestion begins it is necessary to establish a Submission Agreement between the Producer and the Archive. Before dissemination starts an Order Agreement between the Archive and the Consumer is established. Through these agreements both SIP and DIP constitutions are defined well as the specifications of the sessions for data submission and dissemination. The Administration component is responsible to define the AIP constitution – package that will be stored — and responsible monitoring/implementing the preservation.

When we have in mind the long-term preservation of a specific digital object or a specific class of digital objects we immediately face the question: what are the properties that should be consider important and that should be addressed by the preservation strategy? In this chapter we also addressed these significant properties, stressing their importance in terms of the preservation policy adopted and in terms of their relevance for an assertive representation of the preserved object. The definition of such properties can have a major impact on the success of preservation. If the selected properties are suitable to enable a correct representation of the object, the preservation will probably occur successfully; otherwise, we may be able to preserve the object but without an acceptable level of authenticity in comparison with the original one.

Chapter 3

Preservation of Relational Databases

Along this study, Relational Databases (RDBs), are categorized as a class or family of digital objects. Like other families of digital artifacts, such as images, text documents, web sites and others, RDBs have its own specificities and are orthogonal to several domains within the digital universe. Considering a database as an abstract concept, it is possible to state that this notion is transversal to the whole information society: the substructure where the *data* stands on with the purpose of being manipulated (accessed or changed). However, in this work, we confine our efforts by focusing on a specific kind of substructure model – RDBs.

The preservation of a digital object of this nature involves several technological issues because databases are complex structures in such a way that it is possible to consider it as an "hard format". Also because of the heterogeneity of the systems (Database Management System (DBMS)) that support RDBs, the path to be followed towards preservation assumes the direction of normalization, within the RDBs world. So, as in current approaches over this matter, we point to the migration of the database into a normalized format for archival.

Few real solutions exist considering the preservation of RDBs. Nevertheless, it is important to stress two main approaches which are the Software Independent Archival of Relational Databases (SIARD) [SIARD, 2008] solution and the

Portuguese Repository of Authentic Digital Objects (RODA) project [Ramalho, Ferreira, Faria, Castro, Barbedo, and Corujo, 2008] with the Database Markup Language (DBML) [Jacinto, Librelotto, Ramalho, and Henriques, 2002] approach.

There is an European strategy encompassed in the Preservtion and Long-term Access Through Networked Services (PLANETS) project [PLANETS, 2010] to enable RDBs long term access. The project adopted the SIARD solution, which is based on the migration of database into a normalized format (eXtensible Markup Language (XML) [Bray, Paoli, Sperberg-Mcqueen, Eve, and Yergeau, 2003]). The SIARD was initially developed by the Swiss Federal Archives (SFA).

Another approach, also based on XML, relies on the main concept of "extensibility" – XML allows the creation of other languages [J. Ramalho, 2002] (it can be called as a meta language). The DBML was created in order to enable representation of both **DATA** and **STRUCTURE** of the database.

Another approach is the Migration to Intermediate XML for Electronic Data (MIXED)[1] project promoted by Data Archiving and Networked Services (DANS)[2] of Netherlands. The main idea of the MIXED solution consists also on migrating the data (spreadsheets and databases) into XML. The XML file here is seen as an intermediate format as the name of the project indicates [Ren van Horik, 2011]. Even the above mentioned SIARD solution already has an extension the SIARDDK, which the Danish National Archives develop based on the SIARD solution [Werf, 2012].

These approaches (SIARD, DBML and few others) adopt the strategy of Migration of the database to XML, and we may ask why? A neutral format that is hardware and software (platform) independent is the key to achieve a standard format to use in digital preservation of relational databases. This neutral format should meet all the requirements established by the designated community of interest. This related work is later analyzed in this chapter, focusing the SIARD solution and the DBML format.

First we will take a closer look to relational databases as a digital object as

[1]https://sites.google.com/a/datanetworkservice.nl/mixed/
[2]http://www.dans.knaw.nl/

it is! Then, in this chapter, we talk about the importance of databases (RDBs) to organizations and also characterize its specificities, databases management environments and properties. Stating that the database semantics is an important or significant property in terms of database preservation. Before starting the analysis of related work, a background section is provided to introduce the XML, associated technologies and metadata. In the last part of the chapter we analyze the related work considering current approaches and solutions for the preservation of RDBs, namely the DBML format and the SIARD solution.

3.1 Relational Databases: digital object

In the following lines we concentrate our efforts on characterizing RDBs as a digital artifact and as an important component in information systems. The creation of a database consists on process with several stages, from requirements identification, passing trough the database modeling, deployment and system testing. This database lifecycle exists to provide the organization with a database capable of answering its necessities. In other words, the database that support a part or all of the information system, intends to reflect the reality of the organization "business model".

3.1.1 Organizations Cornerstone

The importance of databases to organizations is unquestionable. It is the place where the data of government institutions, companies or any other form of organizations can be stored and structured for computer processing or human consumption. The fact that databases have a determined structure enables the processing of data by machines, for example, at an operational level. The database also has the potential of offering important information (reports, statistics and others) to the organization when needed.

It is possible to claim that part of the organization activity is recorded on databases. Since the database is designed to match the organizations activities, their needs and demands, the structure of the specific designed model (entities, relations and relationships), has the potential to characterize the organization. An idea of the organization "business model", activities or information system can be understood trough the analysis of the relational database specific model. Not only the structure of the model, but also the semantics associated to the conceived RDB.

The data stored in databases is so important to some organizations that without the data of databases they will simply not work. If a main database crashes, the normal operation of an organization can be compromised. Best practices state the importance of database backups and strategies to maintain the main databases in redundant machines to ensure a fast database recover when needed.

Also in the web, databases have a determinant importance considering the websites that have their contents recorded in RDBs. Information about products on e-commerce, news, clients data, credentials or others are just some examples of the impact and transversality that databases have on the information society.

3.1.2 Database Management System & SQL

The technological environment to support database can be widely heterogeneous. From the hardware to the engine of software that gives support to a database, several possible configurations exist. Putting aside the hardware and the operating system of the machine, what remains is the DBMS.

An important issue to notice is that the term "relational database" is related to the conceptual level of digital object; the level that us humans are able to understand and interpret. The underlying levels are the physical and logical levels (hardware and software respectively). So, the logical level corresponds the sequences of multi-bitstreams manipulated by the software engines that handles the database: precisely the DBMS. Some of the several environments capable of handling RDBs (DBMS) are:

- MySQL
- MS SQL Server

- Oracle
- Firebird
- PostgreSQL
- MS Access

The different engines have specific features, associated tools and environments, however is not the purpose of this work to study DBMSs. Nevertheless, there is an important characteristic shared by DBMS: the language to manipulate the data and structure of RDBs - Structured Query Language (SQL). This language interacts with the databases at different levels, from the creation and manipulation of the database structure to the queries and changes to the data itself.

The SQL language was standardized by both American National Standards Institute (ANSI)[3] and International Organization for Standardization (ISO)[4] [Eisenberg and Melton, 1999]. The first version appeared in the early '70s, by the hand of Donald D. Chamberlin and Raymond F. Boyce at IBM [Chamberlin and Boyce, 1974]. This language was created in order to deal with a proprietary system called the IBM System R [Chamberlin, Astrahan, Blasgen, Gray, King, Lindsay, Lorie, Mehi, Price, Putzolu, Selinger, Schkolnick, Slutz, Traiger, Wade, and Yost, 1981], based on the relational model. Then the language evolved, trough some revisions, and turned into a standard [Melton and Simon, 2001]. However, from one vendor of a DBMS to another it is possible to find differences within the SQL implementations and extensions. So, despite of the standardization efforts, some extensions were and are being added by individual owners of the different DBMSs to the standard.

In its essence, the SQL intends to be a very specific language, dedicated to working with databases, which was the purpose of its creation. The language is divided into three types of commands: those to create, change or delete tables, basically to manipulate the structure of the database – Data Definition Language (DDL), and those who are used to insert, update, delete or select data from tables

[3]http://web.ansi.org
[4]http://www.iso.org

– Data Manipulation Language (DML); and the commands to set access privileges to the database – Data Control Language (DCL).

3.1.3 Specificities and Preservation

The information in a relational database has a particular structure based in relations usually called tables [Codd, 1970]. These tables also relate to each other trough relationships. Here enters the notion of primary and foreign keys which support the relationships between tables. Structure features considered important for preservation are:

- **TABLES** (Name)
- **COLUMNS** (Name, Type, Size, ...)
- **KEYS** (Primary keys, Foreign keys, ...)

These elements are crucial to preserve all the database structure – relations (tables) and the relationships between them. There are other features in a database, such as triggers, stored procedures, users privileges, forms, or other application issues that should be consider whether or not to preserve them. Depending on what is considered significant to archive we may choose to preserve it or not. If we choose to preserve application issues, such as a form that interacts with the database, it may be enough to preserve its code or it may be necessary to preserve an image of its appearance. Over this work we concentrate only on elements directly related to the relational model: relations, attributes, relationships and data.

Nevertheless, information that indicates the original operating system and the DBMS used to support the database should be preserved because it is important to characterize the environment of the original database. The date of creation of the database and identification of its creator should also be preserved. This information is identified as technical metadata. There is also the data and associated information which are the database records and that must be subject of preservation. Nevertheless, the significant properties in a digital object are those that are identified by its community of interest.

Lee Buck [Buck, 1999] and Ronald Bourret [Bourret, 2005] on their approaches concerning XML and Databases do not mention any information about the database structure. However, the structure and the semantic of the database may provide a way of interpreting the data in order to work with it and extract valid information – knowledge from the data and knowledge about the organization (as above mentioned). On one hand we have the data stored in the database and on the other hand its structure. The data contained in the records of the database obviously has to be preserved but through this analysis we conclude that it will be necessary to also preserve the database structure and semantics.

3.1.4 Database Semantics

The database modeling process implicates the study and analysis of a real scenario in order to conceived a relational model that gives response the presented requirements. The identified entities, that will result on normalized tables with their attributes and relationships, are modeled from that real scenario (modeled from the real world). The concepts behind the entities, and that give origin to a database in the relational model, have a meaning in the real word that the database architect had the need to understand. The names used for the tables and attributes normally try to reflect those concepts to facilitate the model analysis. Nevertheless, it is important to stress that this assumption is not always true. These concepts associated to the database are considered the database semantics.

Beside the structure and the data, it is worthy to consider the preservation of the semantics of the database. This is the higher abstraction level of the RDBs digital object and is above the conceptual level which corresponds to the relational model. The higher level consists on the experienced object, the one that humans picture inside there minds and has a direct correspondence in the real world (conceptual model). Please note that a conceptual model corresponds to the experienced object and that, in this case, the relational model (for databases) corresponds to the conceptual object.

3.2 Background

Before moving directly into the related work concerning the digital preservation of relational databases, it is important to establish some background concepts. The XML language and some associated technologies play an very important role in this domain and are therefore analyzed here.

Metadata, by its transversally in the field of digital preservation, is also addressed in this section with the purpose of seeing how the preservation of RDBs treats this issue and what methodologies are being adopted.

Lets start by introducing the XML technology as well as its associated standard, the XML Path Language (XPath).

3.2.1 XML & XPATH

Starting with XML language, it is characterized not as a programming language but rather as a markup language. Markup languages to emerge in 1986 with the appearance of the Standard Generalized Markup Language (SGML) also a markup language [J. Ramalho, 2002]. SGML appeared due to the necessity of finding a support for the digital information that should be neutral and platform independent (hardware and software). At the time, already existed a variety of formats for digital documents which made it costly to transfer and manipulate documents between different platforms. Some concern about the longevity of the documents also began to be considered.

In the late 80s Tim Berners-Lee gives rise to HTML, a derivative of SGML. Unlike SGML, which held great complexity, HyperText Markup Language (HTML) has one important feature witch is simplicity. Through the junction between the power of SGML and the simplicity of HTML, XML arises, developed by the XML Working Group, originally known by SGML Editorial Review Board. Some people who were directly involved with SGML later developed the XML under the umbrella of the World Wide Web Consortium (W3C) [W3C, 2012]. The XML appears then in 1996 and in its development a number of initial objectives were present. We quote these original objectives [Cowan, 1998]: "XML Shall be straightforwardly usable over the Internet." "XML Shall support a wide variety of applications." "XML Shall be compatible with SGML." "It Shall be easy to write programs Which process XML documents." "The number of optional features in XML is to be Kept to the absolute minimum, ideally zero." "XML documents Should be human-legible and reasonably clear." "The XML design Should Be prepared quickly". "The design of XML Shall be formal and concise." "XML documents Shall be easy to create." "Terseness in XML markup is of minimal importance".

It was with these objectives that the XML language has grown as well as its usage. The language features prove to be adequate for long-term preservation, given that it is based on the assumptions above mentioned and as begin as an open language as well. Today, and increasingly, the usage of XML provide, among other things, cross-platform interoperability. The generic nature of the language as well as its neutrality offer many advantages. It is also important to note that XML is not really a language but a metalanguage, i.e., a language in which it is possible define or crate new languages. Currently, new XML defined languages are continuously emerging – eXtensible Hypertext Markup Language (XHTML), Standard Music Description Language (SMDL), Chemical Markup Language (CML), etc. The DBML, also dialect derived from XML, arises to accommodate databases and is the one adopted in our work.

Highlighting more technical aspects of language, we emphasize the its tree structure and its constitution based on elements, so-called annotations or labels. Each element can have one or more children and those child elements may also have children themselves. There is no limit to the number of children so that their complexity increases in proportion to the size of the documental tree. The elements are identified by language reserved characters '<' and '>'. The elements can also contain attributes. These features aim to qualify the elements to which they are associated. At the beginning of the XML file must exist that the XML declaration which is indeed a processing instruction. XML documents can also have comments. Following a very simple example of an XML document (Code Block 1):

There are some rules that dictate whether a document is well formed or not,

Code Block 1 : XML sample document

```
<?xml version="1.0"?>
<conversation>
    <message>
        <to>Antonio</to>
        <from>Andre</from>
        <heading type="urgent">Ideas for the weekend</heading>
        <content>This weekend let’s play football or cycling.</content>
    </message>
    <message>
        <to>Andre</to>
        <from>Antonio</from>
        <heading type="normal">Ideas for the weekend</heading>
        <content>Ok. I am looking forward to do it.</content>
    </message>
</conversation>
```

one is for example the impossibility of crossing annotations: <A>Hello <B>World </A>... So initially the document has to respect these rules to be well-formed. In a second phase the XML document may be linked to a Document Type Definition (DTD)[5] or a eXtensible Markup Language Schema (XMLSchema)[6] which is its form of validation. The DTDs were the first to appear and most recently appeared XMLSchema. These last have the advantage of being also written in XML themselves. The aim of either DTDs or the XMLSchemas is to validate XML documents. This validation concerns checking the names of elements and attributes as well as verification of the order which they occupy in the XML documental tree. So, if the XML document respect its schema definition it is considered valid.

To conclude this overview of XML we must refer to eXtensible Stylesheet Language (XSL)[7]. The XSL standard consists of three main standards, XPath, eXtensible Stylesheet Language Transformations (XSLT) and eXtensible Stylesheet Language Formatting Objects (XSLFO). The standard XPath lets you locate and select information in XML documents. The XSLT standard defines methods to transform XML documents into other formats. The XSLFO standard is used to specify the desired presentation format for XML documents. The XPath can be seen as a query language for XML documents and is now analyzed.

[5]http://www.w3.org/TR/REC-xml/
[6]http://www.w3.org/XML/Schema
[7]http://www.w3.org/Style/XSL/

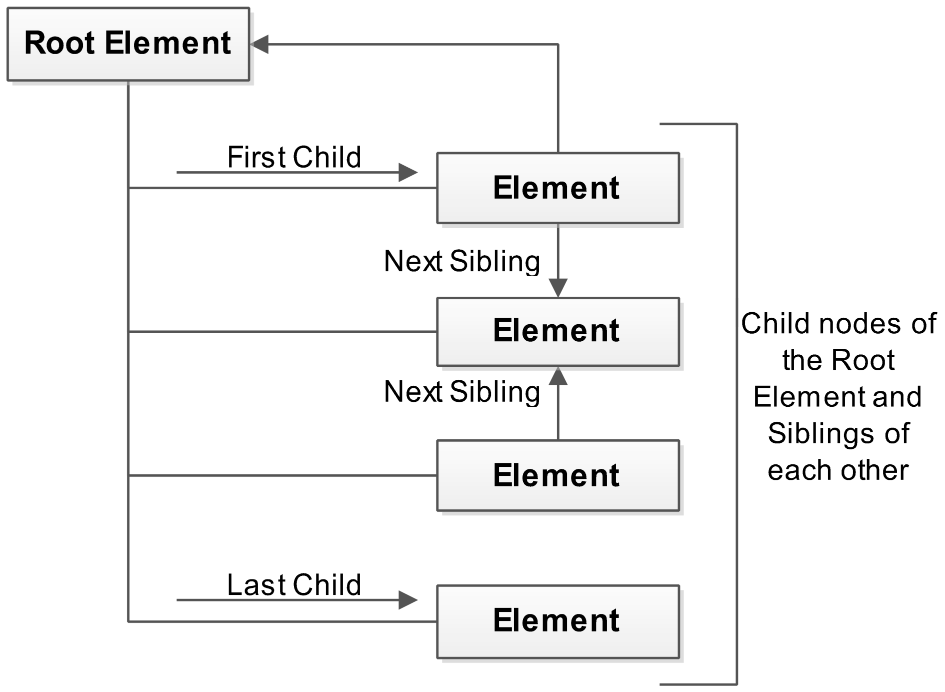


Figure 3.1: Existing relationships in the documental tree (XML)

In a XML document the information is well structured and have a determined organization. But if it is necessary to find or to select specific information (elements or attributes) within the XML document it is necessary a sort of query language to provide the desired information. This language is materialized with the XPath standard. XPath lets you locate and select information in XML documents.

Attending to the Document Object Model (DOM)[8] the entire XML document is made by nodes. An attribute is considered a node, a element is also a node, the text is considered a text node, a comment is a node, and the complete document is itself considered a node. This view fits in a tree structure where we can draw a parallel relationship with the father and son relations. Thus we know, for example, that the children of a node (parent) are considered brothers/sibling. Figure 3.1 presents the possible relations between nodes, considering only part of the documental abstract tree.

Through this analysis it is possible to cross or walk along the structure or tree of the document without knowing in advance the shape of the tree nor the type of data it contains.

[8]http://www.w3.org/DOM/

The selection of the nodes is made in XPath through expression that have parallelism with hierarchy of the file system (Unix or Windows). Following, tow examples to elucidate da reader:

- */conversation/message/content* (selects all <content >children of <message >, which are children of <conversation >)
- */conversation/message/heading/@type* (selects all @type attributes of the element <heading >, children of <message >, which are children of <conversation >)

The selectors can be absolute and relative. There are also the navigation axes which enable the navigation in the documental abstract tree through different perspectives than the child-father relation. For filtering selections exists the predicates the predicates. The XPath functions can return, for example, the number (count) of elements. For a full specification see [XPath W3C Recommendation, 1999] or [J. Ramalho, 2002].

3.2.2 Metadata

Metadata plays an important role in the field of digital preservation since it is a vehicle to pass on information, technical details or information about data (or even information about information) to future consumers of the preserved artifact. If we put ourselves in the position of future consumers of preserved objects, easily we come to the conclusion that the more information available useful to understand and interpreter the artifact, the better. When talking about digital objects, metadata can consist on store information on how the artifact should be interpreted.

Specifically, preservation metadata can assume different types, for example, descriptive information that intends to provide the consumer with an idea about the original object and what was its purpose; or technical information about the preserved object and that will, for instance, help in the rendering or parsing of the digital object. In short metadata can be more conceptual (higher abstraction level) or more technical (lower abstraction level).

Moreover, preservation metadata should document a set of requirements, such as the provenance of the digital object and its authenticity; all the preservation activities should also be registered as well as the necessary technical environment to produce the conceptual object; furthermore, metadata can be used in the documentation and management of legal rights over the preserved artifact [Lavoie and Gartner, 2005][Ferreira, 2009][Consultative Committee for Space Data Systems, 2002].

The Open Archival Information System (OAIS) reference model explicitly talks about "Representation Information" which is indeed metadata about the "Information Object". Additionally, the constitution of the information packages, in the OAIS, besides the information object (the digital object for preservation), is all about metadata ("Descriptive Information", "Preservation Description Information" and "Representation of Information") [Consultative Committee for Space Data Systems, 2002].

Another important work on metadata in the field of preservation of digital objects is the Preservation Metadata: Implementation Strategies (PREMIS) data model [Premis, 2008]. It emerged due to the fusion between the Online Computer Library Center (OCLC) and Research Libraries Group (RLG) (OCLC/RLG) [OCLC/RLG Preservation Metadata Working Group, 2002]. While the OAIS is a reference model, the PREMIS intends to be an concrete implementation of data dictionary for preservation metadata.

3.3 Related Work

Concerning RDBs and its digital preservation, there are two recognizable approaches that claim to support the preservation of these hard format. The SIARD solution encompassed in the PLANETS project and the DBML approach within the RODA project. They were both studied and are detailed here. As it was already mentioned in the beginning of this chapter, both solution adopted a preservation strategy that consists on the migration of the original database into XML. The database elements addressed for preservation and the XML document format differ from one approach to the other.

Before moving forward, some question should be considered: what are the effects of cutting/extracting the object from its original context? We must not forget that the interaction between the source of the digital object and the platform results on a conceptual object that can be different if the platform changes [Wilson, 2007]; the output can be different (will the object maintain its original behavior?).

Considering that there are no perfect scenarios, the main goal is the preservation of the essential parts that purport what the object were made for as must as possible, creating a platform independence. Either the source or the platform can be altered if the essential of the digital object is obtained and also maintaining the meaning of the digital object over the time.

The XML for its neutrality is a perfect vehicle to achieve platform independence and therefore is used also in the digital preservation of relational databases.

3.3.1 SIARD Solution

The PLANETS project incorporated the SIARD solution, which consists on the migration of database into a normalized format – XML. Initially developed under the SFA ARchiving of ELectronic Data and Records (ARELDA) project, it intends to be a normative description of a specific XML schema for relational databases preservation.

The SIARD format claims to be an open standard, into which databases can be converted for preservation, because it uses open standards as Unicode [9], XML and SQL1999 [10] as well the ZIP[11] file format. Although the format specification is available to anyone, the SFA implemented interim a prototype: the SIARD Suite. We analyzed the SIARD format but it was not possible to study in detail the Suite application.

The SIARD approach addresses the following database properties:

- Tables (columns, primary and foreign keys);

[9]http://unicode.org/
[10]A revision of the SQL standard
[11]http://www.info-zip.org/

- Views and Routines;
- Users, Roles and Privileges

The tables, views and routines are considered by SIARD as part of a "Schemata" as they called (database schema), and according to the SIARD format description document there is a relation between the schemata and the database catalogue [SIARD, 2008]. The idea is that a database can has one or more database "Schemata".

The SIARD format can be used independently of any application and it confines itself specifically to the above mentioned database properties, not being its purpose to represent package structures such in the OAIS reference model [SIARD, 2008]. So, the SIARD specification converts the database into a set of XML and XMLSchema documents with a specific structure in the filesystem (files and folders) that in the end are putted together in an uncompressed ZIP file.

Analyzing these documents specifically, they are divided in tow parts (Table 3.1): a) the header and b) the content.

header	metadata.xsd metadata.xml		
content	schema1	table1	table.xsd table.xsd
		table2	table.xsd table.xsd
	schema2	...	...

Table 3.1: SIARD archival file structure

The header folder contains tow files which they called metadata: *metadata.xsd* and *metadata.xml*. The *xsd* file is precisely the XMLSchema to validate the correspondent XML document. These "metadata" contains some technical and descriptive information about the original database and about the archival process. The database structure (schema) of the elements previously mentioned are also incorporated in the header (metadata). Figure 3.2 provides as overview of the XMLSchema file to which the XML document has to comply.

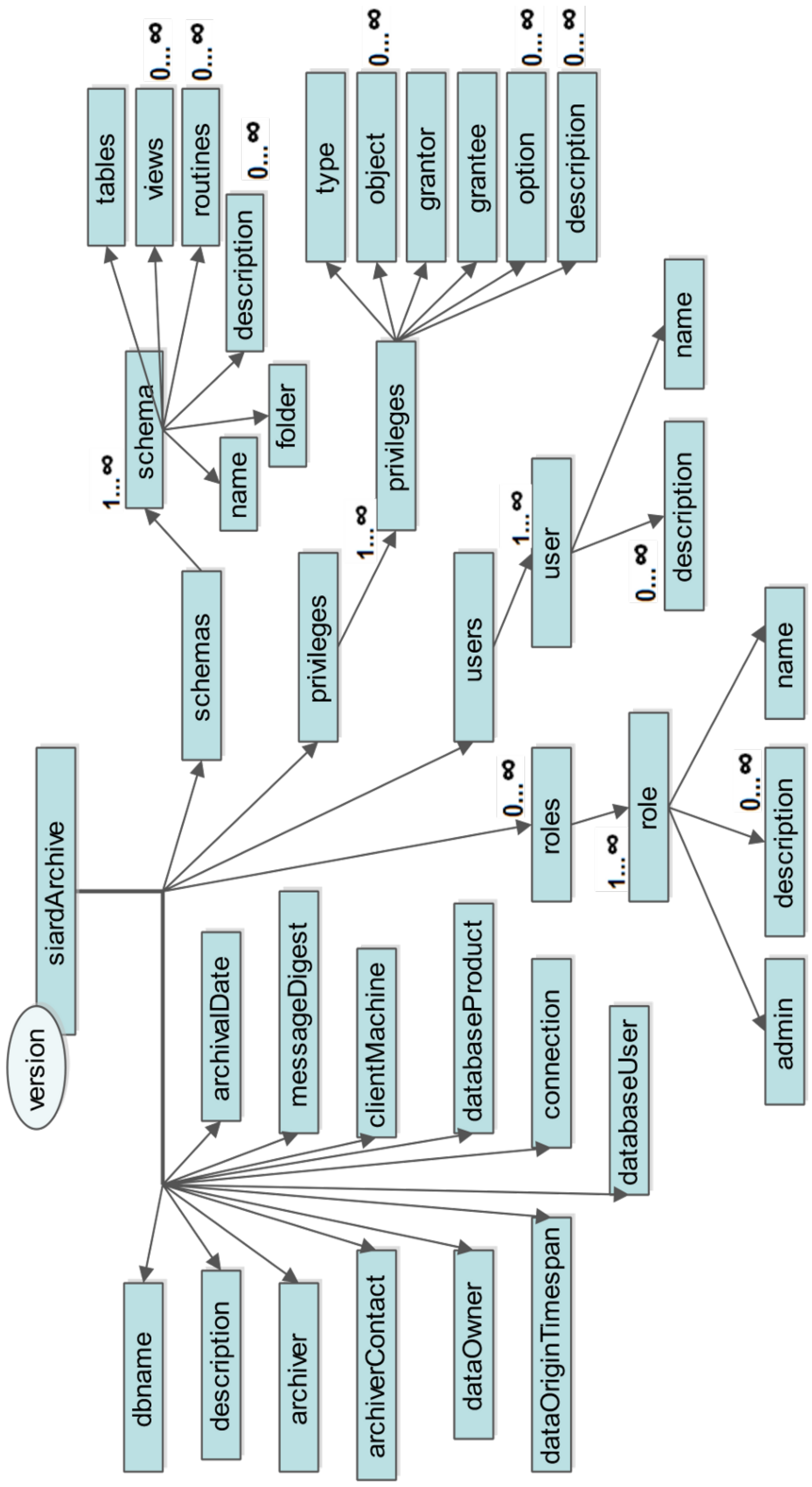


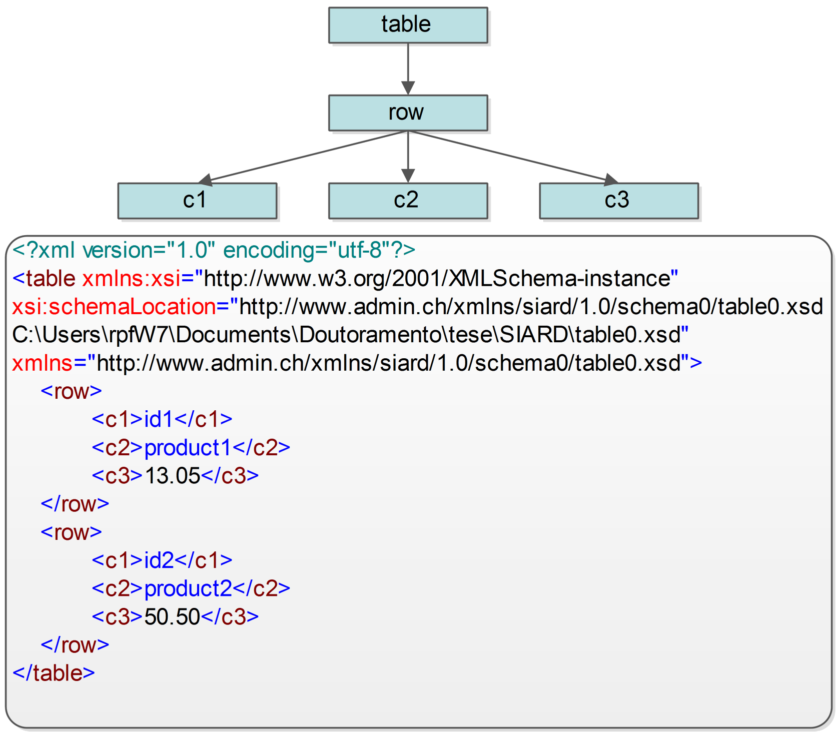
Figure 3.2: SIARD Schema – *metadata.xsd*

The root element is *siardArchive* whom has set of child elements and only one attribute: *version*. The set of child elements are the follwing: *dbname, description, archiver, archiverContact, dataOwner, dataOriginTimespan, archivalDate, messageDigest, clientMachine, databaseProduct, connection, databaseUser, schemas, users, roles, privileges*. Only the last four (*schemas, users, roles, privileges*) have children elements themselves (Fig. 3.2). For example, the *schemas* element can has one or more child elements *schema* under which the *table* element is nested and has the following elements:

- **NAME:** the name that the table assumes in the schema;
- **FOLDER:** contains the name of the table folder (in the schema folder);
- **DESCRIPTION:** information of the about the meaning and content of the table;
- **COLUMNS:** columns/attributes of the table;
- **PRIMARYKEYS:** indicates the primary key of the table;
- **FOREIGNKEYS:** the list of foreign keys (table);
- **CANDIDATEKEYS:** the list of candidate keys (table);
- **CHECKCONSTRAINTS:** the list of the check constraints (table);
- **TRIGGERS:** the list of the triggers (table);
- **ROWS:** the number of rows (table).

A full description of the SIARD format is described in [SIARD, 2008].

The other set of files (content folder) are used to store the data of the database tables. SIARD specification dictates that there will be created 2 files (table.xml and table.xsd) for each database table. A brief example of a SIARD converted table is given in figure 3.3. For tables containing Binary Large Object (BLOB)[12]



```
<?xml version="1.0" encoding="utf-8"?>
<table xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd
C:\Users\rpfW7\Documents\Doutoramento\tese\SIARD\table0.xsd"
xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd">
   <row>
         <c1>id1</c1>
         <c2>product1</c2>
         <c3>13.05</c3>
   </row>
   <row>
         <c1>id2</c1>
         <c2>product2</c2>
         <c3>50.50</c3>
   </row>
</table>
```

Figure 3.3: SIARD Table

or Character Large Object (CLOB)[13] data, separated files must be created and referenced in the XML documents.

Then SIARD format lays on a complex set of XML documents (XML and XMLSchemas) that must relate to each other to be possible to interpreter the archived database. For example the structure (database schema) is separated from the database data itself (it uses tow separated files). This approach attempts to encompass many properties of the database such as users, privileges, roles, routines or views, in addition to tables and relationships. This is an advantage however it bring more complexity to the format.

3.3.2 DBML

The DBML was introduced by the paper ”Bidirectional Conversion between Documents and Relational Data Bases” [Jacinto, Librelotto, Ramalho, and Henriques, 2002]. It is as XML dialect created with the purpose of enabling the interchange

[12]Binary Large Object
[13]Character Large Object

of databases between different database management systems. The XML neutrality (software and hardware platform independency) has the potential of creating languages, such as the DBML, aimed to accommodate specific digital objects and their properties. The DBML appeared previously than the above mentioned SIARD format, although it was not its first purpose to address relational database preservation. However, it was naturally adopted to address the issue of long-term preservation of RDBs.

The Portuguese RODA Project [Ramalho, Ferreira, Faria, Castro, Barbedo, and Corujo, 2008], incorporated the DBML to enable the preservation of RDBs. The project deals also with different families of digital objects such as images or text documents, for example. In the broadest sense it intends to be, as the acronym indicates, a "Repository of Authentic Digital Object".

Backing to the DBML itself, it is characterized to include, in the DBML document, both structure (database schema) and data of the database. Although, it may have suffered some evolutions during its implementation within the RODA project, here we analyze the DBML as it was first presented and published. Nevertheless, some suggestion already included in the first prototype developed [Freitas, 2008] are also presented on this overview. For example, metadata that indicates the date on which the preservation event happened, information about the preservation agent, original database environment or others. Small excerpts from a case study to demonstrate the organization and content of a DBML document are presented (Code Block 2 and the following).

Code Block 2 : Attributes for the Root Element (DB)

```
<DB NAME="CoursePrograms" SGBD="MSSQLServer" SO="Win2008"
    DATEC="2009-05-11" CREATOR="rfreitas" DATEP="2012-01-09" >
```

The existing information in the database has a specific structure, as we saw earlier, based in relationships. Although the data may be dispersed into several tables, the relationships provide means for interpreting the data, to extract valuable information and also to put the database working at operational levels. So there we have the stored data of the database on one hand and on the other hand its structure or schema. Through this analysis, it is valid to conclude that it is necessary to maintain not only data but also the structure of the database. Code block 3 presents the main elements (top level) of a DBML document .

Code Block 3 : Main Elements - Structure and Data

```
<DB NAME="CoursePrograms" SGBD="MSSQLServer" SO="Win2008"
    DATEC="2009-05-11" CREATOR="rfreitas" DATEP="2012-01-09" >
<STRUCTURE>
...
</STRUCTURE>
<DATE>
...
</DATA>
</DB>
```

Structure DBML Element

Starting with the *structure* element, and knowing that a database is composed by tables, one element for each table is included. To define the structure for each table it is necessary to identify all of its attributes (table columns or fields). For each of the attributes it is important to identified its domain or data type and size or length are defined; also it is important to determine whether or not the attribute can be null. It follows (Code Block 4) the structure in DBML to characterize the attributes of our case study (in this example we have suppressed the *SIZE* attribute).

With the characteristics defined for the attributes, now lets concentrate on the keys. For each table, either primary keys and/or foreign keys are identified and incorporated in the DBML document. For the primary keys, the attribute or set of attributes that compose the key are identified and if the primary key has only one attribute is defined as ”simple”; in the case of primary keys consisting of more than one attribute these are defined as ”composite” (Code Block 5).

We are now left with the foreign keys (Code Block 6). In the foreign keys, it is essential to identify which attribute or attributes are part of it, and it is also crucial to identify the table and the attribute or attributes to which every foreign key refer to (the referenced primary keys).

When all these definitions are archived it is possible to establish a complete the

Code Block 4 : Organization of the fields in Table

```
<DB NAME="CoursePrograms" SGBD="MSSQLServer" SO="Win2008"
   DATEC="2009-05-11" CREATOR="rfreitas" DATEP="2012-01-09" >
<STRUCTURE>
  <TABLE NAME="Courses">
   <COLUMNS>
    <COLUMN NAME="CodDisciplina" TYPE="varchar" NULL="NO "/>
        <COLUMN NAME="DesDisciplina" TYPE="varchar" NULL="NO "/>
        <COLUMN NAME="AbrevDisciplina" TYPE="varchar" NULL="YES"/>
        <COLUMN NAME="DesDisciplina_ING" TYPE="varchar" NULL="YES"/>
        <COLUMN NAME="DesDisciplinaOficial" TYPE="varchar"
                                            NULL="YES"/>
        <COLUMN NAME="SubstituirNomeQuandoEquivalencia" TYPE="bit"
                                            NULL="YES"/>
        <COLUMN NAME="DesDisciplina_POR_QuandoPreenchimento" TYPE="varchar"
                                            NULL="YES"/>
        <COLUMN NAME="DesDisciplina_ING_QuandoPreenchimento" TYPE="varchar"
                                            NULL="YES"/>
        <COLUMN NAME="SiglaAreaCientifica" TYPE="varchar" NULL="YES"/>
        <COLUMN NAME="CodAreaCientifica" TYPE="varchar" NULL="YES"/>
   </COLUMNS>
   <KEYS>
   ...
   </KEYS>
  </TABLE>
...
</STRUCTURE>
```

Code Block 5 : Structure of Primary Keys

```
<TABLE NAME="Courses">
 ...
 <KEYS>
  <PKEY TYPE="simple">
   <FIELD NAME="CodDisciplina"/>
  </PKEY>
  ...
 </KEYS>
</TABLE>

<TABLE NAME="ProgramBibliography">
 ...
 <KEYS>
  <PKEY TYPE="composite">
   <FIELD NAME="IDPrograma"/>
   <FIELD NAME="IDLivro"/>
  </PKEY>
  ...
 </KEYS>
</TABLE>
```

Code Block 6 : Structure of Foreign Keys

```
<TABLE NAME="ProgramContent">
 ...
 <KEYS>
  ...
  <FKEY NAME="IDLingua" IN="Languages" REF="IDLingua"/>
  <FKEY NAME="IDPrograma" IN="Programs" REF="IDPrograma"/>
 </KEYS>
</TABLE>

<TABLE NAME="Programs">
 ...
 <KEYS>
  ...
  <FKEY NAME="CodDisciplina" IN="Courses" REF="CodDisciplina"/>
 </KEYS>
</TABLE>
```

schema, in the DBML format, of the whole structure of the database. It is possible to find an example (instance) of part of the element *STRUCTURE* used in one of the case studies, in code block 7.

After the analysis os these portions of DBML code, it is also important to clarify the full schema of the *STRUCTURE* element by detailing the correspondent DTD and XMLSchema files (files that will enable the validating of the XML document).

In summary, the element *STRUCTURE*, which concerns to the schema of the database, is formed by a sequence of one or more elements TABLE. By its turn, the element TABLE is defined by a sequence of two children elements: *COLUMNS* and *KEYS*. The *COLUMNS* element consists on a sequence of one or more elements *COLUMN*; the element *KEYS* consists on a sequence of elements *PKEYS* and zero or more elements *FKEYS*.

- **STRUCTURE**: element that accommodates the whole structure/schema of the database. Its composition is defined by a sequence of elements TABLE.
- **TABLE**: Each table exists in database is mapped to a element *TABLE*. This element has one attribute - *NAME* - that matches the name of the table. The *TABLE* element has two children - *COLUMNS* and *KEYS*.
- **COLUMNS**: This element aims to add a sequence of elements that match

Code Block 7 : Fragment DBML document the case study chosen - STRUCTURE

```
<? Xml version = "1.0" encoding = "ISO-8859-1"?>
<DB NAME="CoursePrograms" SGBD="MSSQLServer" SO="Win2008"
    DATAC="2009-05-11" CRIADOR="rfreitas" DATAP="2012-01-09" >
<STRUCTURE>
 <TABLE NAME="Courses">
  <COLUMNS>
   <COLUMN NAME="CodDisciplina" TYPE="varchar" NULL="NO "/>
   <COLUMN NAME="DesDisciplina" TYPE="varchar" NULL="NO "/>
   <COLUMN NAME="AbrevDisciplina" TYPE="varchar" NULL="YES"/>
   <COLUMN NAME="DesDisciplina_ING" TYPE="varchar" NULL="YES"/>
   <COLUMN NAME="DesDisciplinaOficial" TYPE="varchar"
                                       NULL="YES"/>
   <COLUMN NAME="SubstituirNomeQuandoEquivalencia" TYPE="bit"
                                       NULL="YES"/>
   <COLUMN NAME="DesDisciplina_POR_QuandoPreenchimento" TYPE="varchar"
                                       NULL="YES"/>
   <COLUMN NAME="DesDisciplina_ING_QuandoPreenchimento" TYPE="varchar"
                                       NULL="YES"/>
   <COLUMN NAME="SiglaAreaCientifica" TYPE="varchar" NULL="YES"/>
   <COLUMN NAME="CodAreaCientifica" TYPE="varchar" NULL="YES"/>
  </COLUMNS>
  <KEYS>
   <PKEY TYPE="simple">
    <FIELD NAME="CodDisciplina"/>
   </PKEY>
  </KEYS>
</TABLE>

<TABLE NAME="ProgramContent">
 <COLUMNS>
  <COLUMN NAME="IDPrograma" TYPE="int" NULL="NO "/>
  <COLUMN NAME="IDLingua" TYPE="int" NULL="NO "/>
  <COLUMN NAME="Objectivos" TYPE="varchar" NULL="YES"/>
  <COLUMN NAME="ObjectivosEsp" TYPE="varchar" NULL="YES"/>
  <COLUMN NAME="Competencias" TYPE="varchar" NULL="YES"/>
  <COLUMN NAME="Metodologia" TYPE="varchar" NULL="YES"/>
  <COLUMN NAME="Programa" TYPE="varchar" NULL="YES"/>
  <COLUMN NAME="Avaliacao" TYPE="varchar" NULL="YES"/>
  <COLUMN NAME="Recursos" TYPE="varchar" NULL="YES"/>
  <COLUMN NAME="PalavrasChave" TYPE="varchar" NULL="YES"/>
 </COLUMNS>
 <KEYS>
  <PKEY TYPE="composite">
   <FIELD NAME="IDPrograma"/>
   <FIELD NAME="IDLingua"/>
  </PKEY>
  <FKEY NAME="IDLingua" IN="Languages" REF="IDLingua"/>
  <FKEY NAME="IDPrograma" IN="Programs" REF="IDPrograma"/>
 </KEYS>
</TABLE>
...
</STRUCTURE>
<DATE>
...
</DATA>
</DB>
```

the *COLUMN* table fields.

- **COLUMN**: For each field or column of the table will have an element *COLUMN*. This element has four attributes: *NAME*, *TYPE*, *SIZE* and *NULL*.
  - **NAME**: Attribute that indicates the name of the table's column (field or attribute).
  - **TYPE**: Attribute that designates the domain or data type of the column.
  - **SIZE**: Attribute that indicates the possible size for the data in the certain column.
  - **NULL**: Attribute that indicates whether the field under analysis may or may not be null (empty).
- **KEYS**: The *KEYS* element has a child element PKEY and may or may not also have FKEY elements as children.
- **PKEY**: the element which defines the table's primary key. This element has a *TYPE* attribute and has a sequence of one or more *FIELD* elements.
  - **TYPE**: Attribute that can take two values: *COMPOSITE* or *SIMPLE*. If the primary key is formed by more than one table field, the attribute, takes the *COMPOSITE* value; otherwise, where the primary key is formed by only one field, the attribute takes the value *SIMPLE*.
- **FIELD**: For each field that makes part of the primary key there will be a *FIELD* element. This element has also the *NAME* attribute.
  - **NAME**: Attribute that defines the name of the field or fields that compose the primary key.
- **FKEY**: Element in which it is defined the foreign keys. This element contains three attributes: *NAME*, *IN* and *REF*.
  - **NAME**: Attribute that defines the name of a field or fields that compose the foreign key.
  - **IN**: This attribute that indicates the name of the table on which this foreign key is primary key, i.e., the table to which this foreign key refers.

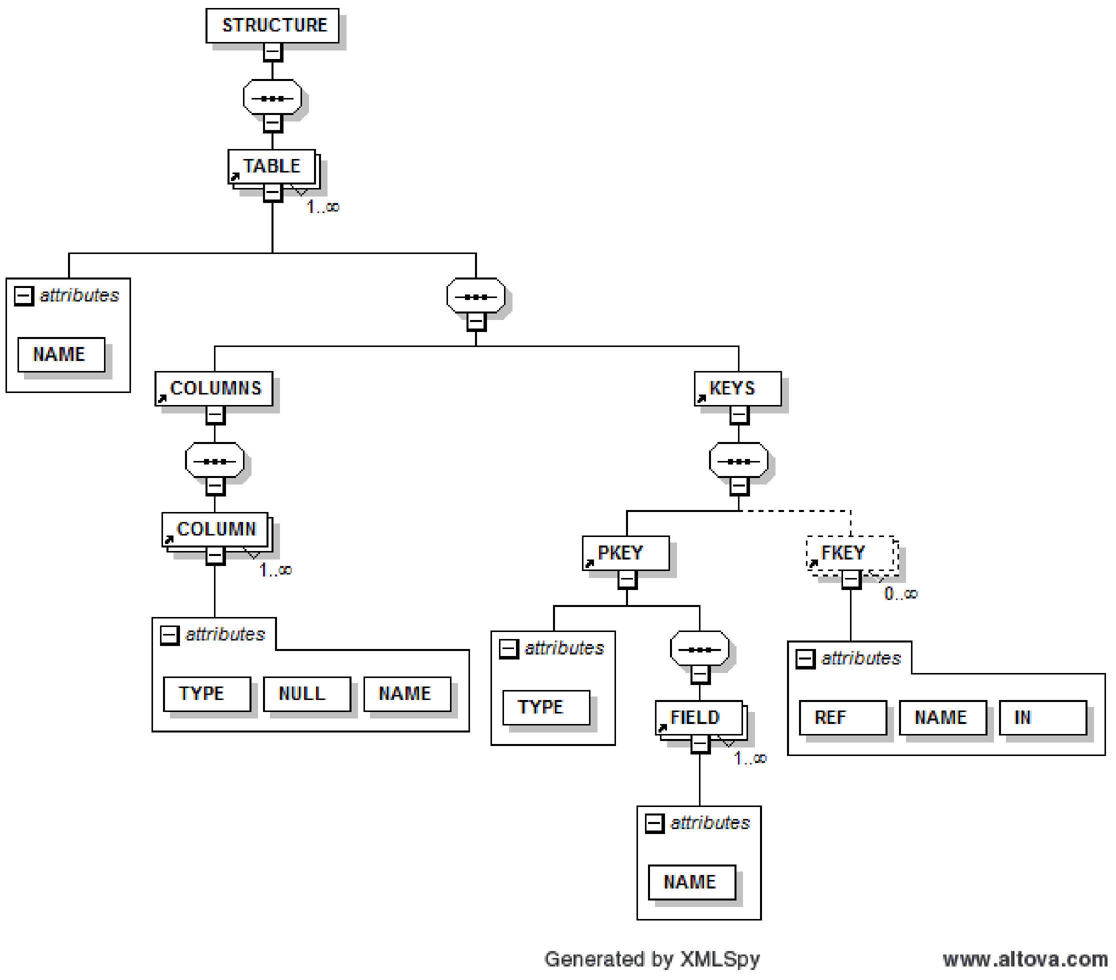


Figure 3.4: DBML - XMLSchema for the STRUCTURE element

– **REF**: Attribute that indicates the field to which this foreign key corresponds in the table (defined in the attribute IN) where it is the primary key.

Figure 3.4 presents the XMLSchema corresponding to the *STRUCTURE* element definition (more details in Appendix A).

Data DBML Element

The XML document – DBML – also contains the database data itself. The final document accommodates the structure of the database and also accommodates existing information (data) in the database. As it was mentioned before, besides the *STRUCTURE* element whose content reflects the tables structure and relationships of the database schema, there is also have the *DATA* element. This element is responsible for accommodating all the information that can be found in relations (tables as they are currently addressed). To clearly show how the data of the tables are stored in DBML, we present a short instance (case study) of two records of a table (Code Block 8).

Code Block 8 : Fragment of the element DATA

```
<DATA>
<Courses>
 <Courses-REG>
  <CodDisciplina>05327</CodDisciplina>
  <DesDisciplina>HISTRIA CONTEMPORNEA DE PORTUGAL</DesDisciplina>
  <AbrevDisciplina/>
  <DesDisciplina_ING>CONTEMPORARY PORTUGUESE HISTORY</DesDisciplina_ING>
  <DesDisciplinaOficial/>
  <SubstituirNomeQuandoEquivalenc>0</SubstituirNomeQuandoEquivalenc>
  <DesDisciplina_POR_QuandoPreenc/>
  <DesDisciplina_ING_QuandoPreenc/>
  <SiglaAreaCientifica/>
  <CodAreaCientifica/>
 </Courses-REG>
 <Courses-REG>
  <CodDisciplina>05328</CodDisciplina>
  <DesDisciplina>SEMINRIO: ARQUEOLOGIA</DesDisciplina>
  <AbrevDisciplina/>
  <DesDisciplina_ING>SEMINAR: ARCHEOLOGY</DesDisciplina_ING>
  <DesDisciplinaOficial/>
  <SubstituirNomeQuandoEquivalenc>0</SubstituirNomeQuandoEquivalenc>
  <DesDisciplina_POR_QuandoPreenc/>
  <DesDisciplina_ING_QuandoPreenc/>
  <SiglaAreaCientifica/>
  <CodAreaCientifica/>
 </Courses-REG>
</Courses>
...
<ProgramContent>
 ...
</ProgramContent>
...
</DATA>
```

The DTD and/or XMLSchema for these kind of XML structure, doesn't impose much complexity. However, a problem arises because it is not possible to know in advance what will be the schema without knowing the database to be converted. The names of tables and also its columns which determine the name of the XML elements will certainly vary according to the database under process of migration.

Given this particularity and if we want to validate the DBML files before it is necessary to prepare a DTD or an XMLSchema specific for each database. In figure 3.5 we can find the XMLSchema for the *DATA* element that reflects a database migration used in the case studies. More details related to the XMLSchemas, namely the DTDs used in the cases studies are presented in the Appendix A.

The usage of DTDs or XMLSchema enables the validation process of an XML document, such as providing a valid DBML document.

Now we can say that the structure for the file DBML is fully defined. It is noteworthy that the structure holds much of the characteristics or properties that we considered important and fundamental to the preservation of a database. Issues such as routines, users and privileges, triggers and others are not supported by the DBML format. Nevertheless, the considered critical data and structure of the database (the structure as a means pt interpreter the data and the relational model) are addressed in DBML document; the DBML schema has a concise structure, well defined and makes fully usage of the XML potentialities in terms of elements and attributes; and doesn't have a too complex structure.

To end this overview about the DBML format, we provide the reader with a diagram that intends to reflect the complete XMLSchema of a DBML document (Fig. 3.6).

3.3.3 Findings

The migration of databases to XML formats is the way to pursue RDBs preservation in the current paradigm. The current solutions differ in terms of the XML dialect adopted but it is consensual the usage of XML. Both data and structure of the database are to be archived for preservation, i.e., the database records and the RDB schema (tables, relationships and attributes). The SIARD approach even goes further by supporting also views, triggers, users privileges and routines. This fact makes the DBML approach less complex and more assertively pointed at the data and schema of the database specifically (the data and the data structure or organization – the relational model).

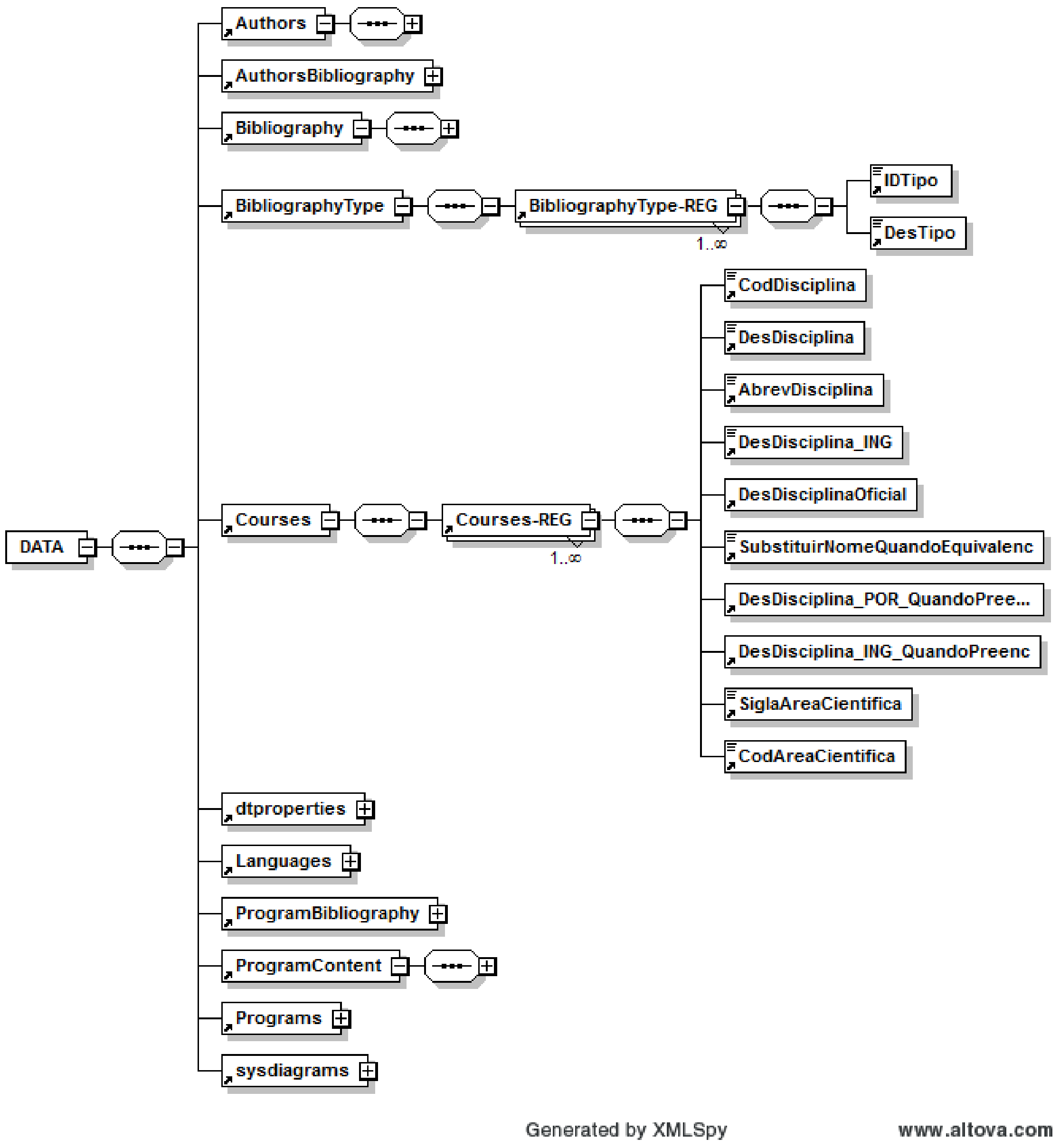


Figure 3.5: DBML - XMLSchema of the DATA element

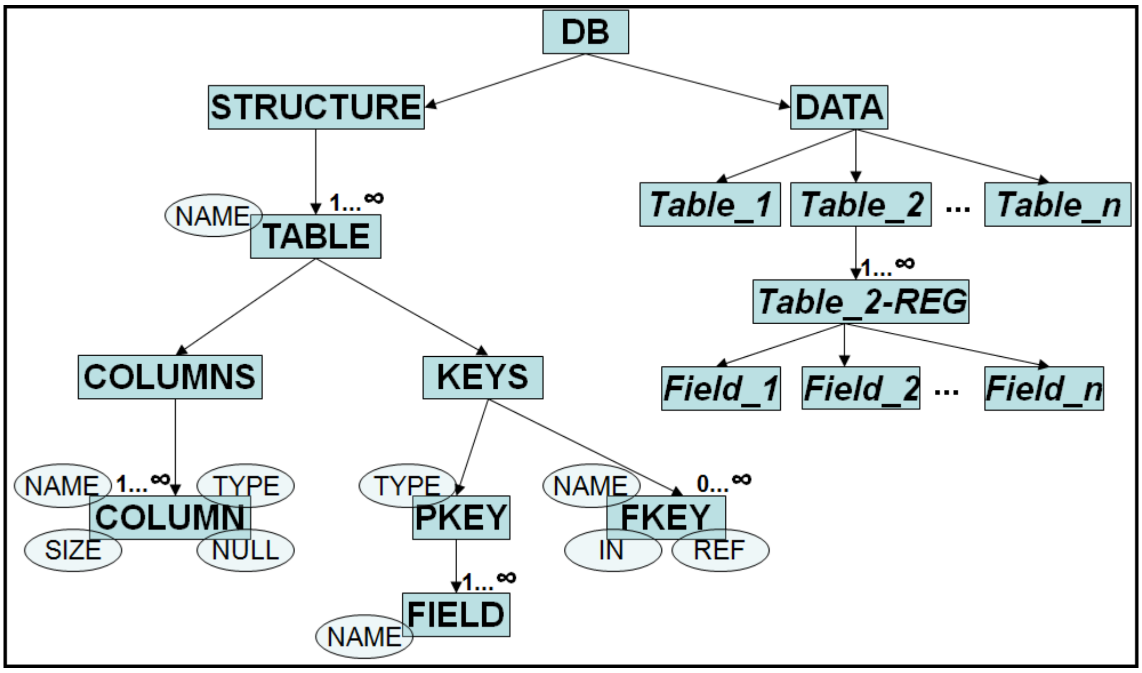


Figure 3.6: DBML Schema

Still, from the analysis of these two main approaches towards the preservation of relational databases, a fundamental question remains unanswered: what about the database semantics? Neither of the above XML formats expressly support the incorporation of the semantics of the database. The importance of this database property has to do with the preservation of the knowledge associated to the database concepts. Moreover, the semantic of any digital object can be seen as a sort of metadata about the conceptual notions of what the object was. In our particular case, the semantic can correspond to the meaning of the entities and the significance of their relationships before they were coded (modeled) into the relational model for databases

3.4 Summary

Relational Databases (RDBs) are digital objects categorized here as a class or family of objects within the whole scope of digital artifacts. The first part of this chapter was dedicated to preform an overview analysis about RDBs class of digital objects.

The necessity of structuring data to store the organizations information (operational or other) as well as the huge spread of RDBs worldwide justify gives an idea of the its impact in the digital universe. There are many different domains that relay on RDBs, such as organizations information systems, content management in the web or even to support e-commerce activities, just to name a few.

RDBs fits in the chain of abstraction levels present in all digital objects: physical, logical and conceptual levels. The physical object corresponds to the digital artifact stored in a physical digital media for hardware interpretation (not the focus of our study). At the logical level, the object consists on a set of bitstreams (or multi-bitstreams) coded specifically for a software interpretation. Some of these softwares Database Management Systems (DBMSs) are referenced and we note that their environments are quite different from each other. The conceptual level of RDBs can be materialized in the relational model with its tables, attributes and relationships. The Structured Query Language (SQL), which is standard to manipulate RDBs, also differs in some aspects between different DBMSs. It is here where the eXtensible Markup Language (XML) enters as possible format or vehicle to archive and preserve the conceptual object (the database relational model schema and data). Nevertheless, the relational model has specific technical details that are not present in reality. To achieve a normalized (third normal form at least) database reality needed to be somehow modeled into the relational model specifications. This last abstract level (to reality) detains knowledge in a way that the relations and relationships have meaning and represent concepts: the database semantics.

Because the majority of the related work points to the migration of the database into XML formats, a short background section given on XML and associated technologies, namely the XML Path Language (XPath) standard. In the background section we also perform an overview about the metadata role in digital preservation and more specifically on the preservation of RDBs.

The second part of the chapter is dedicated to detail some of the related work in the field of digital preservation of RDBs. As already mentioned the XML is widely adopted to be format used in the preservation of RDBs. In the Software Independent Archival of Relational Databases (SIARD) solution suggests a set of XML and eXtensible Markup Language Schema (XMLSchema) files into where the database can be migrated. The Database Markup Language (DBML) format used in Repository of Authentic Digital Objects (RODA) also adopts this preservation strategy of migration the database into XML. These main approaches considering the preservation of RDBs were analyzed, detailing the XML dialects (schemas) used on both of them.

After preforming the literature review and analysis of the related work (solutions and approaches) on RDBs digital preservation we came to the conclusion that the XML format, because of its platform (software/hardware) independency, has huge potentiality in this domain. It can easily be structured to store the data of the database and can also capture the schema (structure) of the database (relational model). However, there is a significant property of the database not addressed by current approaches: the database semantics. Based on this premise, our work evolved from the initial approached and leveraged the research into an higher level of abstraction.

Chapter 4

Raising the Perception Level

After all considerations and analysis concerning digital preservation and relational databases presented in the previous chapters, we are now in place to start detailing the main contribution of our research.

Databases are all about data, structured data and information. Well, information means that it was possible to extract knowledge from the structured data, i.e., perceive the database semantics. The preservation of data and structure concerning Relational Databases (RDBs) is addressed in our first approach [Freitas, 2008] and is resumed in the next section of this chapter. Nevertheless, the perspective of preserving not only i) the Data, ii) the Structure but also iii) the Semantics intrinsic to databases, leads us to address the preservation of the experienced digital object layer. Capturing this new dimension allows the preservation of concepts and their meaning concerning the information inside a database. This can be seen not as technical metadata but as semantic metadata!

Of course this idea can lead to some discussion because the concepts that we are preserving are related to a certain database and by that defined in a given context or scope. A certain concept or body of knowledge may also evolve altering its meaning through time.

Conceptual models, used for example in information systems design, consist on a sort of notation for better interpretation of reality under analysis. In practice, conceptual model are precisely the exteriorization of mental representations of a certain reality. These models play an important role in the requisites identification and on the dialog between systems (Information System (IS)) architects and stakeholders. Concepts and their relationships are indeed conceptual models.

The study led us to work with ontologies as the strategy to formalize the knowledge associated to database semantics and also to create an abstract (due to the neutrality of the eXtensible Markup Language (XML)) representation of the same knowledge. The preservation policy now focuses on the two top levels of abstraction: the database logical model and the database conceptual model.

There is a link, a close relationship between RDBs and ontologies (both represent a sort o reality [abstract or not]). Ontologies, specified with Web Ontology Language (OWL), which is the adopted format, can be a vehicle to achieve knowledge representation and also a way to enable the interpretability and interoperability between possible heterogeneous systems or platforms. Ontologies differ somehow from other conceptual models since once formalized they can be directly interpretable by machines (eg.: reasoners).

This chapter starts with an overview of the previous stages of the research where the first prototype was developed and tested, leading us to note some important findings and establish some conclusions, namely the need to raise the database representation abstraction level. Then, still in this chapter, we introduce the notion of conceptual preservation. The possible preservation of concepts, semantic and/or conceptual models in the origin of a RDBs is a important step on raising the preservation abstraction level. The last part of the chapter is dedicated to establish the methodology adopted to raise the database abstraction level. The preservation policy implemented by the developed framework is clearly detailed as well as the preservation life cycle of RDBs. A relation between RDBs and ontologies is identified and the two top abstraction levels (the logic and conceptual models) present in databases are related to Database Markup Language (DBML) and OWL, which are the target languages into which databases are mapped (converted). As background information, the Resource Description Framework (RDF) and OWL technologies are introduced and detailed with the help of some examples. Related work considering migrations or conversions between RDBs and an ontologies are also analyzed.

4.1 Initial Approach

In previous work [Freitas, 2008] we addressed the preservation of the RDBs data and structure by developing an archive prototype that uses the DBML format for preservation. Our first approach covers the preservation of the logical model of databases (tables, structure and data). However, a conclusion that grew out of this research, was that neither the DBML format approach nor others (e.g. Software Independent Archival of Relational Databases (SIARD) [SIARD, 2008]) are concerned with the database semantics.

Conceptually, the prototype is based on the Open Archival Information System (OAIS) [Consultative Committee for Space Data Systems, 2002] reference model. The OAIS model of reference does not impose rigidity with regard to implementation, rather it defines a series of recommendations. The OAIS model is accepted and referenced for digital preservation purposes since it is concerned about a number of issues related to digital artifacts preservation: the process of information Ingestion into the system, the information storage as well as its administration and preservation, and finally information access and dissemination [Day, 2006] [Lavoie, 2004]. Three information packages are the base of the archival process: Submission Information Package (SIP), Archival Information Package (AIP) and Dissemination Information Package (DIP) (these concepts were introduced in chapter 2, section 2.3).

4.1.1 Prototype Architecture

The initial prototype implementation process consisted on building a web application with multiple interfaces. These interfaces have the mission to take a certain database and ingest it into the archive. The access to the archive in order to do all the necessary interventions on the system is also done through those web interfaces. A dedicated physical machine was used with the following technologies:

a Linux distribution for the Operating System and on top of that, the Apache (Hypertext Transfer Protocol (HTTP)) server[1] plus the PHP: Hypertext Preprocessor (PHP)[2] and MySQL[3], programming language and support database respectively; the unixODBC Application Programming Interface (API) (from unixODBC Project [4]) was used for connecting to heterogeneous Database Management Systems (DBMSs).

A crucial phase of the work was indeed the implementation stage: a system capable of ingest databases, in the form of information packages (DBML + metadata) for preservation. The metadata in the first prototype consisted only on technical metadata that characterized the original database environment and metadata that document the preservation action of ingesting the database into the archive.

The various web interfaces can only be accessible through a previous authentication on the system. The administration component manages these requests, and the various privileges with regard on the handling of information in the archive. The users of the system can be divided into two types, namely two profiles of users: administrators and users. A user has at his disposal the following list of operations [Freitas and Ramalho, 2009]:

- Creation of SIPs;
- Ingestion of SIPs into the repository (AIPs);
- Repository browsing;
- Production of Structured Query Language (SQL) from AIPs;
- Dissemination of DIPs.

The administrator has at its disposal all operations available to users with the addition of operations associated with administration tasks:

- User management;

[1]http://projects.apache.org/projects/http_server.html
[2]http://www.php.net/
[3]http://www.mysql.com/
[4]http://www.unixodbc.org/

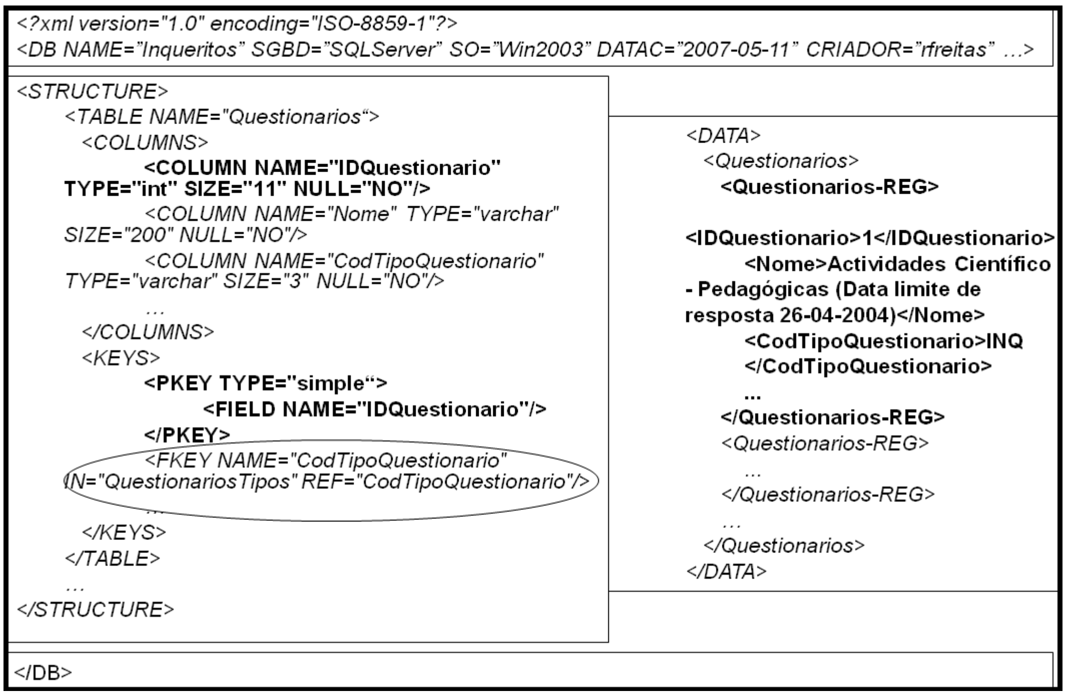
- Direct access to the repository directory;
- Manipulation of drivers (unixODBC) to manage connections with databases;
- Monitoring of the state of obsolescence of information in the repository;
- Actions of migration of the stored information whenever necessary (preservation policies).

The consumers have access to the archived DBML database (XML document) and also access to the database in a target DBMS. This was achieved because we were able to rebuild, from the DBML, the database in SQL code to reconstruct the database in MySQL. By using a DBMS in the dissemination process, consumers have a practical a standard form to navigate and access the archived database through SQL queries [Freitas and Ramalho, 2009]. This dissemination strategy consists on rebuilding the archived database in a current DBMS for user exploration and browsing. If in the future that DBMS becomes obsolete, another one must be chosen.

4.1.2 The First Case Study

In the first case study we used a frozen database on which was no more transactions were expected from the operational point of view. In figure 4.1 a portion of code extracted from a DBML document produced by the prototype used in the case study is presented.

The case study has proven to be very significant because it was possible to obtain interesting results. After the initial assembly of all the tools necessary to perform the required tasks, we quickly started to develop the prototype. Some adjustments were made to obtain better performance of the system. It is important to refer that this work aimed to test the feasibility of relational database digital preservation using this approach. This was indeed possible, i.e., the objective of converting relational databases (different DBMSs) into DBML was achieved.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DB NAME="Inqueritos" SGBD="SQLServer" SO="Win2003" DATAC="2007-05-11" CRIADOR="rfreitas" ...>
```

```
<STRUCTURE>
    <TABLE NAME="Questionarios">
      <COLUMNS>
            <COLUMN NAME="IDQuestionario"
    TYPE="int" SIZE="11" NULL="NO"/>
            <COLUMN NAME="Nome" TYPE="varchar"
    SIZE="200" NULL="NO"/>
            <COLUMN NAME="CodTipoQuestionario"
    TYPE="varchar" SIZE="3" NULL="NO"/>
            ...
      </COLUMNS>
      <KEYS>
            <PKEY TYPE="simple">
                  <FIELD NAME="IDQuestionario"/>
            </PKEY>
            <FKEY NAME="CodTipoQuestionario"
    IN="QuestionariosTipos" REF="CodTipoQuestionario"/>
            ...
      </KEYS>
    </TABLE>
    ...
</STRUCTURE>
```

```
<DATA>
  <Questionarios>
    <Questionarios-REG>

<IDQuestionario>1</IDQuestionario>
        <Nome>Actividades Científico
- Pedagógicas (Data limite de
resposta 26-04-2004)</Nome>
        <CodTipoQuestionario>INQ
        </CodTipoQuestionario>
        ...
    </Questionarios-REG>
    <Questionarios-REG>
      ...
    </Questionarios-REG>
    ...
  </Questionarios>
</DATA>
```

```
</DB>
```

Figure 4.1: DBML portion of the document - case study

4.1.3 Initial Conclusions

This first plan of attack to the problem, provided interesting results, considering the preservation of the database data and structure (logical model) by ingesting the database in a XML based format (DBML [Jacinto, Librelotto, Ramalho, and Henriques, 2002]) into an OAIS [Consultative Committee for Space Data Systems, 2002] based archive. However, we came to the conclusion that a significant property was not being covered by this preservation policy, the above mentioned, Database Semantics. The focus of our research then turned into this problem, related to the conceptual model of the database, i.e., the information system on the top of the operational database.

4.2 Conceptual Preservation

To introduce the idea of conceptual preservation, first it is important to clarify the term concept (conceptual). The notion of "concept" can be identified as an abstract mental representation. This mental representation addresses the essential properties of a certain group of objects, or the significant properties of a class of beings – something that exists, i.e., something that is. This definitions remounts to ancient philosophy and are supported by philosophers at least to Aristotle [Pitt, 2008].

The idea behind "Conceptual Preservation" consists on preserving mental representations (knowledge) associated to the concepts or semantics intrinsic to objects or entities. In this particular study, it intends to preserve the semantic associated to the digital object (RDBs).

4.2.1 Concepts & Semantics

Things, being abstract or not, have associated representations materialized by language (pictures, symbolism) [Wittgenstein, 1922] [Allott, 2003]. A conceptual model of a certain reality is related to its mental representation – the experienced reality by a certain individual. Still, in order to exteriorized that mental representation, a language is necessary. Languages (being symbolic or other) are the vehicle to express, to communicate or to transmit mental representations associated to concepts or associated to conceptualized models.

To preserve the database semantics means precisely to preserve the language in which it is possible to express the conceptual level of RDBs, i.e., the conceptual model of the database or in other words its mental representation. The preservation of the conceptual model is the answer to our goal.

The capture of knowledge associated to the database semantics is indeed an asset in terms of preservation. Nevertheless, we must not forget that considering the main goal of this work (RDBs preservation), concepts associated with the database semantics have a meaning that normally depend on the context so this meaning may vary from one context to another. Moreover, with the passage of years (decades or more) the concepts may also evolve in terms of their meaning. So, the semantics that certain representation possess, may not be accurate if this problems occur. These issues should not be completed discard and may (in extreme situations) represent a limitation in our preservation policy. A future improvement can consist on the usage of Persistent Identifiers (PI)[5] for concepts, data and object properties.

4.2.2 Conceptual Models & Databases

Conceptual models are used in many domains, however in information systems they play a key role. The model is materialized in a language notation such as the Entity-Relationship model, Unified Modeling Language (UML)[6] or other. Ontologies are also accepted has a formal notation to represent conceptual models and have the asset of being processed (reasoned) by machines.

In several several domains ontologies are playing key roles, and the information systems field is not an exception. Knowledge engineering, database design and integration, also information retrieval and extraction are just some examples of fields in which otologies are increasingly being used [Guarino, 1998].

In the origin of the relational database (before the modeling process) the database architects have the necessity to understand the reality that they are trying to model. Then a conceptual model of that reality is created and is the starting point to the more technical process of building the specific relational database model. It is the process of moving from the conceptual model into the logical model of the database. The relational model for databases possess technical features to comply with its the mathematical/formal definitions [Codd, 1970]. It is the existence of this two layers or levels of abstraction that are in the base of the policy of preservation we are currently working with.

Moreover, there is a link between the conceptual structure of an Information System and the conceptual model of the database of support. There are several definitions for information systems that mainly point to a kind of system that integrates subsystems, inter-relationships among machines and human processes, events, attributes and information [Mora, Gelman, Cervantes, and Forgionne, 2009]. To archive also the conceptual model of the database can also mean to store the conceptual structure of the information system on its top.

[5]A persistent digital object identification for long term preservation

[6]http://www.uml.org/

4.3 Raising the Abstraction Level

Based on the first prototype we now intend to include in the information packages (SIP, AIP and DIP) an higher representation level of the database — the conceptual model of the database. Ontologies are used to address semantics and conceptual model representation.

Our hypothesis concentrates on the potentiality of reaching relevant stages of preservation by using ontologies to preserve of RDBs. This led us to the preservation of the higher abstraction level present in the digital object, which corresponds to the database conceptual model. At this level there is an inherent **Knowledge** associated to the database semantics (Tab. 4.1).

Digital Object	Preservation Levels	Relational Database
Experienced Object	**OWL**	**Conceptual Model**
Conceptual Object	DBML	Logical Model

Table 4.1: Preservation Policy - rising the abstraction level

The goal is to capture the experienced object (knowledge) through an ontology based approach. This experienced or knowledge object is the final abstraction. The ontology approach is adopted to formalize the knowledge present at the experienced object level and also a methodology to create an abstract representation of it. The system evolved into an OAIS based architecture that enables the ingestion, preservation and dissemination of relational databases at two levels of abstraction — logical and conceptual (Fig. 4.2). This approach is also an extension to previous approaches in terms of metadata since the ontology provides information about the data at a conceptual level. Figure 4.2 also shows a possible preservation "lifecycle" of RDBs.

4.3.1 Ontologies and Databases

There is a direct relation between ontologies and databases: a database has a defined scope and intends to model reality within that domain for computing (even when it is only virtual or on the web); ontology in ancient and philosophical significance means the study of being, of what exists [Berners-Lee, Hendler, and Lassila, 2001].

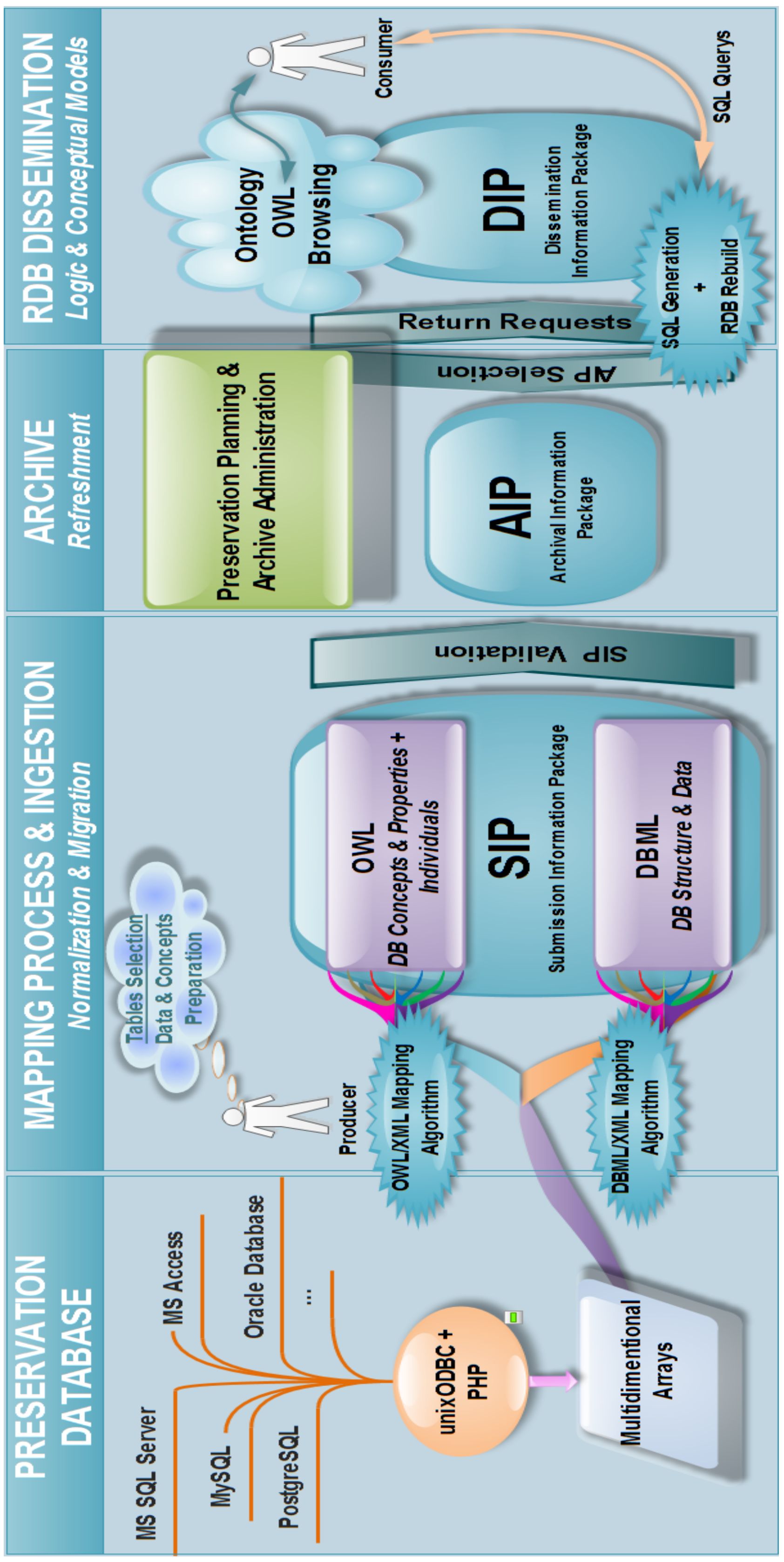

PRESERVATION DATABASE
MS SQL Server
MySQL
PostgreSQL
MS Access
Oracle Database
...
unixODBC + PHP
Multidimentional Arrays
MAPPING PROCESS & INGESTION
Normalization & Migration
Tables Selection
Data & Concepts
Preparation
Producer
OWL/XML Mapping Algorithm
DBML/XML Mapping Algorithm
OWL
DB Concepts & Properties + Individuals
SIP
Submission Information Package
DBML
DB Structure & Data
SIP Validation
ARCHIVE
Refreshment
Preservation Planning & Archive Administration
AIP
Archival Information Package
AIP Selection
RDB DISSEMINATION
Logic & Conceptual Models
Ontology
OWL
Browsing
Consumer
DIP
Dissemination Information Package
Return Requests
SQL Generation + RDB Rebuild
SQL Querys

Figure 4.2: RDBs Preservation Framework

The (strong) entities present in relational databases have an existence because they were model from the real world: they relate to each other and have associated attributes. In information society and computer science, an ontology establishes concepts, their properties and the relationships among them within a given domain [Gruber, 2009].

A database can be defined as a structured set of information. In computing, a database is supported by a particular program or software, usually called the Database Management System (DBMS), which handles the storage and management of the data. In its essence a database involves the existence of a set of records of data. Normally these records give support to the organization information system; either at an operational (transactions) level or at other levels (decision support – data warehousing systems). In particular, the relational databases model is designed to support an information system at its operational level. Thus, RDBs are complex and their data can be distributed into several entity relations that relate to each other through specific attributes (foreign to primary keys) in order to avoid redundancy and maintain consistency [Codd, 1970].

If we intend not only to preserve the data but also the structure of the (organization) information system we should endorse efforts to characterize (read) the database semantics. It is intended to raise the representation level of the database up to the conceptual model and preserve this representation. In other words, we represent the conceptual model of the database using an ontology for preservation.

An Ontology is a "formal specification of a conceptualization" [Gruber, 1993]; a very short definition of ontologies.

The study of ontologies in computer science received new impetus due to the growth of the web, their associated semantics and the possibility of extracting knowledge from it. Tim Berners-Lee realized that years ago when he first referred to the "Semantic Web", now supported by World Wide Web Consortium (W3C) [W3C, 2012] which works on establishing a technology to support the *Web of data* [Semantic Web, 2012]. Notice that a tremendous part of the web is based in (relational) databases — specially dynamic websites.

Behind the ontology there is the need of knowledge representation for machine interpretation. Two technologies: a) the RDF[7] triples give support for the meaning in simple sentences b) and XML[8] is used for structuring documents [Berners-Lee, Hendler, and Lassila, 2001].

The notion of ontology emerges from the need of expressing concepts in different domains (ontologies as collections of information). An ontology can provide readable information to machines [Santoso, Haw, and Abdul-Mehdi, 2011] at a conceptual level (higher abstraction level). They also enable the integration and interpretability of data/information between applications and platforms. Ontologies benefit from the fact that they are not platform/system dependent when compared to traditional relational databases.

4.3.2 XML: RDF & OWL

An ontology embraces the knowledge in specific domains and, in order to express that same knowledge, languages are necessary. Before moving forward into a concrete ontology language OWL, capable of establishing a body of knowledge to represent the semantics of a RDB, lets start by analyzing the RDF language.

RDF – Resource Description Framework

The RDF language was designed to represent information about resources available in the World Wide Web (WWW) [Manola and Miller, 2004]. Information about resources in the web, or in other words, metadata about those resources, like a title or a modification date are some of the features that RDF can provide as information. However, the Resource Description Framework is not girdled only to web resources. By performing a generalization of what is an web resource, it is possible to use the framework to represent information about other types of items not directly retrieved by the web [Manola and Miller, 2004]. The framework is based on a set of triples, – *object, property, value* – that we can also define as – *subject, predicate, object* [Miller, 1998] [Zarri, 2005]. Using these simple properties is possible to gather information about things.

[7]http://www.w3.org/RDF/
[8]http://www.w3.org/XML/

The main asset that the RDF language has is the common framework available for applications, since it can be processed by machines. So, we are in the presence of a framework capable of expressing information about things, giving meaning or significance to them, and with the advantage of being parsed and exchanged by and between applications.

RDF is XML based and normally addressed as RDF/XML language. Code block 9 represents a simple example, extracted from documents used during this study, presenting the *object, property, value* triples in the base of the RDF language:

Code Block 9 : RDF language excerpt.

```
<!--
http://www.semanticweb.org/ontologies/2011/10/Ontology(...).owl#Bibliography_4706
 -->
 <NamedIndividual rdf:about="&Ontology1322235056844;Bibliography_4706">
    <rdf:type rdf:resource="&Ontology1322235056844;Bibliography"/>
    ...
    <Ontology1322235056844:Bibliography_has_Titulo rdf:datatype="(...)xsd;string">
        Manual de Gesto de Pessoas e do Capital Humano
    </Ontology1322235056844:Bibliography_has_Titulo>
    <Ontology1322235056844:Bibliography_has_Editora rdf:datatype="(...)xsd;string">
        Silabo
    </Ontology1322235056844:Bibliography_has_Editora>
    <Ontology1322235056844:has_Authors rdf:resource="&Ontology(...);Authors_6776"/>
 </NamedIndividual>
```

In this example we can see a list of RDF triples associated with the *Bibliography_4706* resource (the object). For example, a triple involving a data *property* shown here is:

- **Object:** Bibliography_4706;
- **Property:** Bibliography_has_Editora;
- **Value:** Silabo;

Another example involving an object *property* instead, is the the *has_Authors* property that has as its value another resource. Resource A has an object property P which its value is another resource B (A implicates B through P). The resource B may have itself properties indicating that a *network* of relationships between resources can be established by the usage of object properties. For identification of resources, its properties and properties values RDF uses Uniform Resource Identifiers (URIs)[Klyne and Carroll, 2004][Berners-Lee, Fielding, and Masinter, 1998].

In order to consolidate the knowledge inherent to the information that RDF provides about things, an extension of the RDF was created, the RDF Schema also called the RDF vocabulary description language. This extension is indeed a semantic extension, characterized by introducing the notions of classes of resources (with domains and ranges) and also notions such as properties description and relation between them and other resources [Brickley and Guha, 2004].

OWL – Web Ontology Language

The Web Ontology Language was created based on the previous RDF and RDF Schema. It is a general purpose language developed to deal with ontologies and to be World Wide Web compliant [Mcguinness and van Harmelen, 2004]. Earlier approaches and languages were very specific, specialized in certain areas and therefore were not able to be used globally and with general purposes. Ontologies have classes, properties associated to those classes and also individuals of the classes, which OWL is capable of handle. Moreover, OWL introduces more vocabulary such as the possibility of establishing relations between classes, defining cardinality or even describing properties with functional or symmetric characteristics [Mcguinness and van Harmelen, 2004].

OWL is divided in three sublanguages: **OWL Lite** (capable of dealing with hierarchy classification and simple constrains); **OWL DL** (capable of handle maximum expressiveness with guarantee of computational completeness); and **OWL Full** (capable of handle maximum expressiveness with full syntactic freedom, like RDF, but with no computational guarantees).

A main difference between OWL DL and OWL Full is the fact that in OWL Full a class can be, for example, an instance of another class; something that in OWL DL is not possible.

An ontology is characterized by being made of *classes* and *subclasses*, namely a class hierarchy. In OWL all classes have a superclass named *Thing* which is also the class of all individuals (represents the universe). Classes have properties that can be *data properties* (establishing a relation between an individual and a data value) or *object properties* (establishing relationships between individuals). OWL also supports property hierarchies. A certain property must have a *domain* and a *range* which are related by that same property. An instance of a class is named as an *individual*. These are the main features of OWL [Mcguinness and van Harmelen, 2004]. Two portions of OWL code produced during this research, concerning the mapping from RDBs into ontologies, are presented in code block 10 and 11.

Code Block 10 : OWL code example describing only ontology structure

```
<SubClassOf>
    <Class IRI="#Authors"/>
    <Class abbreviatedIRI=":Thing"/>
</SubClassOf>

<DisjointClasses>
        <Class IRI="#Authors"/>
        <Class IRI="#Courses"/>
</DisjointClasses>

<ObjectPropertyDomain>
        <ObjectProperty IRI="#has_Authors"/>
        <Class IRI="#Bibliography"/>
</ObjectPropertyDomain>

<ObjectPropertyRange>
        <ObjectProperty IRI="#has_Authors"/>
        <Class IRI="#Authors"/>
</ObjectPropertyRange>

<InverseObjectProperties>
        <ObjectProperty IRI="#is_Authors_Of"/>
        <ObjectProperty IRI="#has_Authors"/>
</InverseObjectProperties>
```

In this first example (Code Block 10) the presented code describes structure (or schema) components of the ontology: the definition of the class *Authors* as subclass of *Thing*; disjoint between to distinct classes; and the definition of an object property (*has_Authors*), its domain and range as well as an inverse object property of it. In order to give a graphical overview about this structure, a image (Fig. 4.3) extracted from Protégé[9] ontology editor is also presented.

[9]http://protege.stanford.edu

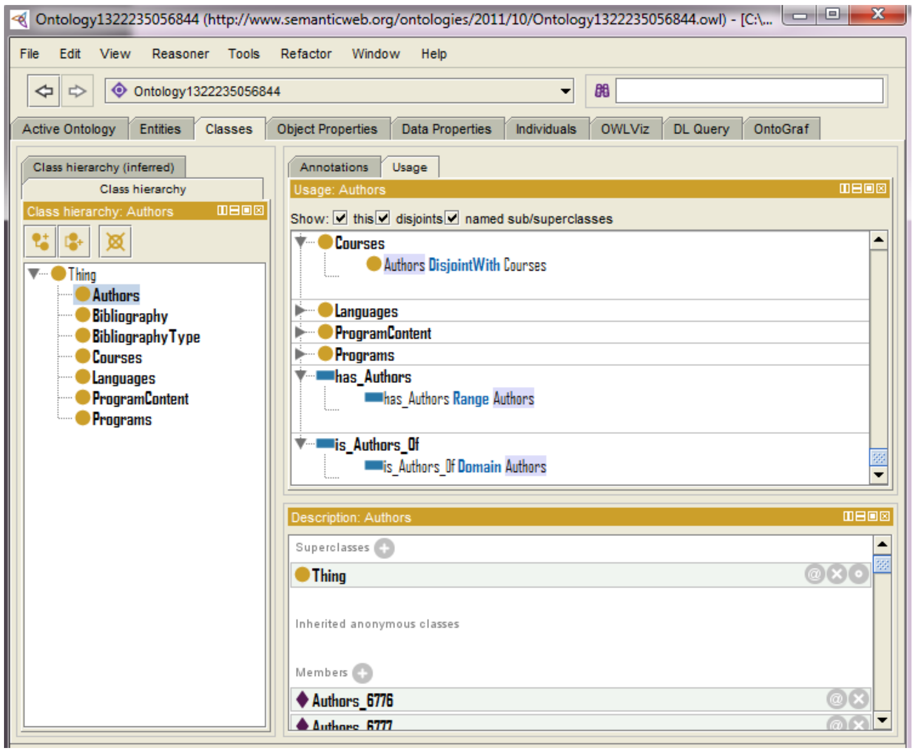

Ontology1322235056844 (http://www.semanticweb.org/ontologies/2011/10/Ontology1322235056844.owl) - [C:\...
File Edit View Reasoner Tools Refactor Window Help
Ontology1322235056844
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Class hierarchy (inferred)
Class hierarchy
Class hierarchy: Authors
Thing
Authors
Bibliography
BibliographyType
Courses
Languages
ProgramContent
Programs
Annotations Usage
Usage: Authors
Show: this disjoints named sub/superclasses
Courses
Authors DisjointWith Courses
Languages
ProgramContent
Programs
has_Authors
has_Authors Range Authors
is_Authors_Of
is_Authors_Of Domain Authors
Description: Authors
Superclasses
Thing
Inherited anonymous classes
Members
Authors_6776
Authors_6777

Figure 4.3: Ontology Classes Structure – (from Protégé)

The second example (Code Block 11 and Fig. 4.4) shows the assertion of an individual (*Authors_6931*) by its class. It is also possible to see an example of an object property assertion (*Bibliography_4813* → *has_Authors* → *Authors_6931*); and also a data property assertion (*Authors_6931* → *Authors_has_IDAutor* → *"6931"*).

In figure 4.4 we are also able to verify an inference produced when the generated ontology was classified by the reasoner. In this example the reasoner inferred that the individual *Authors_6931 is_Author_Of Bibliography_4813*. This inference is produced because the ontology defines the *has_Authors* and *is_Author_Of* has inverse properties (in chapter 5 we also refer to the inferences topic).

To end the set of OWL code samples and graphical views of the ontology classes, properties and individuals it is important to show the object properties of the ontology. Figure 4.5 shows these properties and focuses on the *has_Authors* property detailing its domain, range, individual assertions and also an inferred equivalent object property.

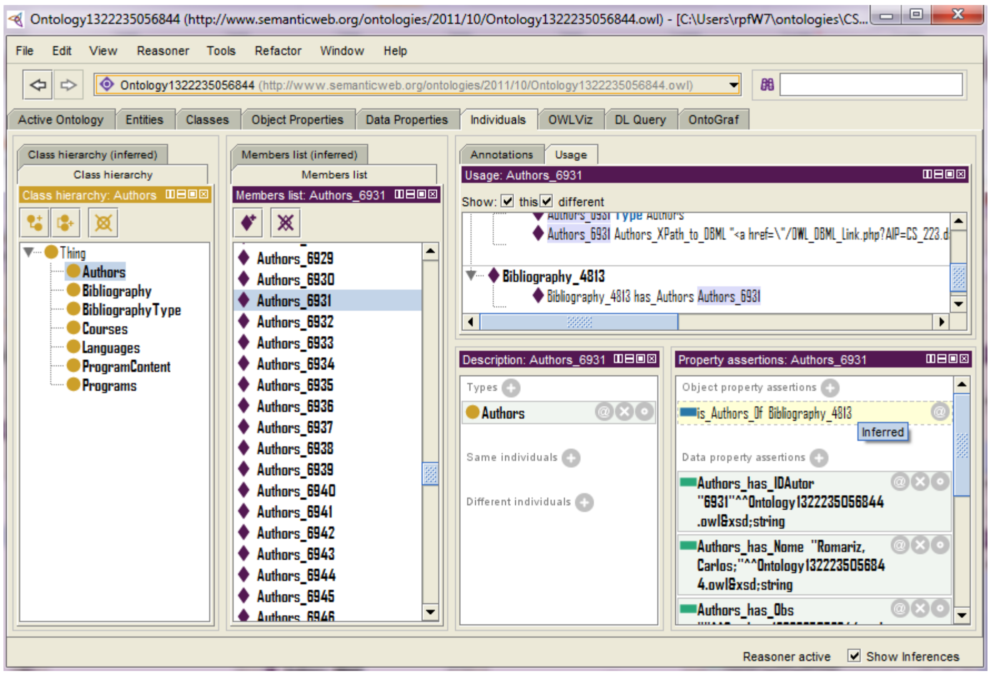
OWL also supports descriptions of equality and inequality (*equivalentClass*,

Code Block 11 : OWL code example describing an individual assertions

```
<ClassAssertion>
        <Class IRI="#Authors"/>
        <NamedIndividual IRI="#Authors_6931"/>
</ClassAssertion>

<ObjectPropertyAssertion>
        <ObjectProperty IRI="#has_Authors"/>
        <NamedIndividual IRI="#Bibliography_4813"/>
        <NamedIndividual IRI="#Authors_6931"/>
</ObjectPropertyAssertion>

<DataPropertyAssertion>
        <DataProperty IRI="#Authors_has_IDAutor"/>
        <NamedIndividual IRI="#Authors_6931"/>
        <Literal datatypeIRI="(...);xsd;string">6931</Literal>
</DataPropertyAssertion>
```



Ontology1322235056844 (http://www.semanticweb.org/ontologies/2011/10/Ontology1322235056844.owl) - [C:\Users\rpfW7\ontologies\CS...

Ontology1322235056844 (http://www.semanticweb.org/ontologies/2011/10/Ontology1322235056844.owl)

Class hierarchy: Authors

Thing
- Authors
- Bibliography
- BibliographyType
- Courses
- Languages
- ProgramContent
- Programs

Members list: Authors_6931

Authors_6929
Authors_6930
Authors_6931
Authors_6932
Authors_6933
Authors_6934
Authors_6935
Authors_6936
Authors_6937
Authors_6938
Authors_6939
Authors_6940
Authors_6941
Authors_6942
Authors_6943
Authors_6944
Authors_6945
Authors_6946

Usage: Authors_6931

Show: [✓] this [✓] different

Authors_6931 Type Authors
Authors_6931 Authors_XPath_to_DBML "<a href=\"/OWL_DBML_Link.php?AIP=CS_223.d

Bibliography_4813
Bibliography_4813 has_Authors Authors_6931

Description: Authors_6931

Types
Authors

Same individuals

Different individuals

Property assertions: Authors_6931

Object property assertions
is_Authors_Of Bibliography_4813
Inferred

Data property assertions
Authors_has_IDAutor "6931"^^Ontology1322235056844.owl&xsd;string
Authors_has_Nome "Romariz, Carlos;"^^Ontology1322235056844.owl&xsd;string
Authors_has_Obs

Figure 4.4: Ontology Individuals – (from Protégé)

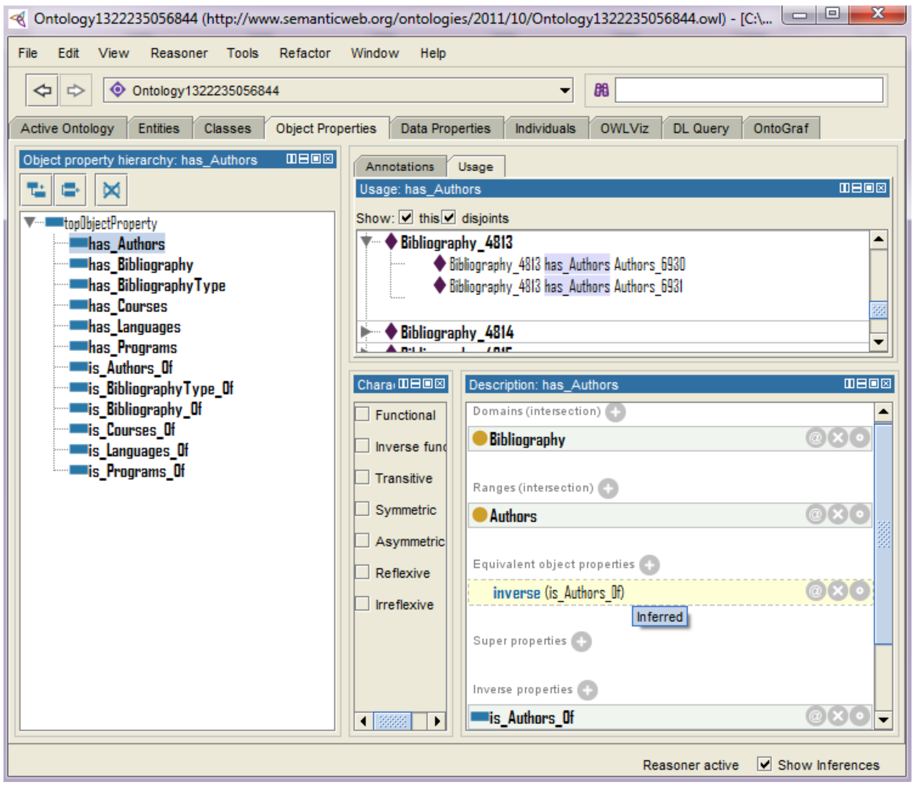


Figure 4.5: Ontology Object Properties – (from Protégé)

equivalentProperty, differentFrom, etc), descriptions for property characteristics (e.g. *inverseOf, FunctionalProperty or SymmetricProperty*), descriptions for property restriction (*allValuesFrom and someValuesFrom*), descriptions to restrict cardinality (*minCardinality, maxCardinality and cardinality*) and also a constructor for describing class intersection (*intersectionOf*) [Mcguinness and van Harmelen, 2004]. The datatypes in OWL were mostly adopted from the eXtensible Markup Language Schema (XMLSchema) datatypes[10].

Some incremental descriptions of OWL (OWL DL & OWL Full) are: *oneOf (enumerated classes)*; *hasValue (property values)*; *minCardinality, maxCardinality, cardinality (for full cardinality)*, for example. However, at this stage (in this work), concerning the conversion between relational databases and ontologies, we are using only the main (principal) feature of OWL (classes, properties, some properties characteristics and individuals).

[10]http://www.w3.org/TR/xmlschema-2/

4.3.3 Related Work

Work related to RDBs and ontologies transformations proliferate and are addressed continuously. Considering the RDF [RDF, 2004], OWL [Mcguinness and van Harmelen, 2004], ontologies and RDBs, several frameworks, mapping approaches and tools exist: Virtuoso RDF View [Several, 2009]; D2RQ [Bizer and Cyganiak, 2007]; Triplify [Auer, Dietzold, Lehmann, Hellmann, and Aumueller, 2009]; RDBToOnto [Cerbah, 2008]; R2O [Rodriguez and Gómez-Pérez, 2006]; Dartagrid Semantic Web toolkit [Chen and Wu, 2005]; SBRD Automapper [Fisher and Dean, 2007]; XTR-RTO [Xu and Li, 2007]; RDB2OWL [Būmans and Čerāns, 2010]; DB2OWL [Cullot, Ghawi, and Yetongnon, 2007]; R2RML [W3C, 2011]; OntER [Trinkunas and Vasilecas, 2007]; DM2OWL [Albarrak and Sibley, 2009]; OWLFromDB [He-ping, Lu, and Bin, 2008] and also "Concept hierarchy as background knowledge" proposal [Santoso, Haw, and Abdul-Mehdi, 2011] among others.

Several of these approaches and tools are referenced and analyzed in the W3C Incubator group survey [Sahoo, Halb, Hellmann, Idehen, Jr., Auer, Sequeda, and Ezzat, 2009] and also in [Santoso, Haw, and Abdul-Mehdi, 2011].

The conversion from databases into an ontology could be characterized as a process in the scope of reverse engineering [Trinkunas and Vasilecas, 2007]. While some approaches and works try to establish a mapping language or a mapping process [Myroshnichenko and Murphy, 2009], others use different techniques and strategies for the database translation [Albarrak and Sibley, 2009] into an ontology (e.g. OWL).

The R2RML (RDB to RDF Mapping Language) [W3C, 2011] working draft submitted to W3C is designed for mapping the data within the attributes of a **table** into pairs: property, object. Each record within a table share the same subject in this RDF triple map relation. This approach supports the input of "logical" tables from the source database, which can be an existing table, a view or a valid SQL query. Also in cases where attributes are foreign keys a pair (property, object) referencing the correspondent table is generated. The rules for this mapping are then organized in a vocabulary with several classes and subclasses (*TripleMapClass, SubjectMapClass, PredicateMapClass, ObjectMapClass, RefPredicateMapClass, etc*).

For example, R2O [Rodriguez and Gómez-Pérez, 2006] approach is based on a mapping document generation (mapping language). Virtuoso RDF View establishes a set of RDF statements by defining for each table: *primary key* (subject), *attribute* (predicate), *value* (object). In the RDB2OWL [Būmans and Čerāns, 2010] a different strategy is used since it is created a mapping RDB schema. The "Concept hierarchy as background knowledge" proposal [Santoso, Haw, and Abdul-Mehdi, 2011] gives special attention to the data preparation before conversion and to the knowledge that resides on the database.

4.4 Summary

This chapter intends to perform a bridge between the study and analysis performed in the previous chapters, concerning digital preservation and how strategies can apply to the scrutinized digital object under study (Relational Databases (RDBs)), and the framework for the preservation of relational databases, introduced in the following chapter.

From the first hypothesis, where a prototype was developed and tested for the archival (with the purpose of preservation) of databases Data and Structure, some issues questions appeared concerning the preservation of the database semantics. An overview of the first prototype was performed in the beginning of this chapter, stating the policy of preservation adopted then and detailing the prototype architecture and operation. Database Markup Language (DBML) (an eXtensible Markup Language (XML) dialect) was the language adopted to accommodate the data of the RDB and also its structure or schema. The schema that corresponds to the relational model, i.e., the logical model RDBs.

The prototype proved the feasibility of converting RDBs into an XML format (migration and normalization) and also enabled the reconstruction of RDBs from the DBML into a traditional Database Management Systems (DBMSs). Nevertheless, the database semantics, information about the database concepts that could provide knowledge about the preserved data was missing. So, a new hypothesis was established that consisted on raising the perception level of the preserved database, i.e., try to preserve an higher abstraction level of the database. That higher layer present in digital object is called the experienced object, which corresponds to the database conceptual level.

Concepts as mental representations of reality are also analyzed in this chapter, with the perspective of preserving or archiving, along with the logic model of the database (initial approach), these two levels of abstraction. The idea defended here is the preservation of what is conceptual not only what is technical (by technical we mean the technical constrains of the relational model). The conceptual model, inherent to a certain database as a knowledge base associated, this can be exteriorized by the usage of an ontology based approach. The usage of an ontology to represent the semantics of the database is an asset in terms of knowledge preservation (conceptual preservation).

The preservation life-cycle and policy for RDBs is clearly detailed in this chapter and consists on including in the information packages (Submission Information Package (SIP), Archival Information Package (AIP), Dissemination Information Package (DIP)) two levels of abstractions corresponding to the conceptual and experienced digital object. The language adopted to describe the database semantics (the database ontology) is Web Ontology Language (OWL). Technically, the packages will include DBML and OWL.

For better understanding the integration of the new abstraction level (OWL) in the preservation process a background subsection is provide that overviews Resource Description Framework (RDF) and OWL concerning the creation of ontologies. Some related work concerning the conversion from RDBs into ontologies are also addressed in the chapter.

The findings and the alteration of the preservation policy (now including two abstraction levels of the database), led to the development and implementation of new prototype for the preservation of RDBs, detailed and analyzed in the following chapter.

Chapter 5

Framework for Relational Databases Preservation

The architecture of the developed and implemented system, which we refer as the **F**ramework for **Re**lational **D**ata**B**ases **P**reservation (FrepDB), is detailed in this chapter. Conceptually, the framework stands upon the Open Archival Information System (OAIS) reference model, already introduced in chapter 2.

We planned a system based on a web platform with multiple interfaces. These interfaces support the main functions of producing, archiving and disseminating information packages that reflect a) data, b) structure (schema) and c) semantics (knowledge/information) of the database under the process of preservation. This new system was conceived following the previous prototype developed during our first approach to the problem [Freitas, 2008].

A complete platform was conceived and deployed to perform the main tasks of "capturing" a database (a frozen database or a database snapshot), process the data and the database structure, and then produce the Database Markup Language (DBML) and Web Ontology Language (OWL) code representing the original database (data, structure and semantics). The produced data is archived as an Archival Information Package (AIP). The system also provides two main forms of accessing the stored database: a) exploring the ontology through an ontology browser and b) rebuilding the database through the generation of the necessary

Structured Query Language (SQL) code (Appendix B provides a screen shot image of a FrepDB web interface – Fig. B.1).

The fact that the archived package of the database contains mainly eXtensible Markup Language (XML) code, ensures that we are adopting a format that is neutral and independent of hardware and software platforms (XML is widely used in interoperability task between heterogenous systems). Nevertheless, we also adopt (suggest the adoption) the refreshment of hardware technic. It is important to have some insurance that the hardware that supports the filesystem (physical medium) works throughout the preservation years.

We start by concentrating our efforts characterizing the preservation environment (technical details) and then on detailing the mapping process and analyzing the created algorithms. This is followed by the evaluation section where the performed tests over the case studies databases and the corresponding results are presented.

5.1 Proof of Concept

The main purpose of this implementation was the fact that we needed to test the feasibility of our hypothesis. After conceptualizing this integrated policy/strategy, where we bring to discussion a new dimension to be included in the digital preservation of Relational Databases (RDBs) (chapter 4), we felt the necessity to put those ideas in practice. So we developed a framework (prototype) to demonstrate the feasibility of the proposed preservation policy which stands as our proof of concept.

In order to verify if the proposed theoretical strategy has potential of being used in real scenarios, we developed the framework prototype capable of proving the practical application of the theory as we predicted.

5.1.1 Technical Details

To perform a technical overview of the system we use a bottom-up analysis, starting from the hardware base and operating system up to the working programming languages as well as other involved technologies.

Hardware - Virtualization

The machine used in the project was an HP[1], model ProLiant DL380 G4. It has 2 Intel(R) Xeon(TM) CPU at 3,2GHz (2CPUs x 3,2Ghz). In terms of resources it has 3 Gigabytes of Random Access Memory (RAM) and an hard disk of 29 Gigabytes. Upon this hardware base we installed the VMware ESXi[2], 4.1.0. This virtualization product installs directly on the hardware (server), uses a Linux kernel[3] and enables the possibility to create several virtual machines sharing the physical resources of the underlying server. The decision for this option, in virtualization, was based on our experience with this virtualization environment and also because it is free to use with basic features.

On the top of the indicated hardware/software base, we created a virtual machine for our project with the following characteristics:

- 1 virtual CPU (vCPU);
- 3072 Megabytes RAM;
- 20 Gigabytes Hard Disk;

By adopting this strategy, the virtual machine supports the system (framework), and assumes a portable format since it can be moved from one location to another by only coping the virtual machine files. This is not an approach to directly implement a preservation strategy of emulation or of any other kind! However, it seems to us that considering the nature of this work (a PhD project with the associated case studies and tests over the system), the virtual machine approach gives more flexibility. We are talking about the possibility of moving the framework from one server to another without having to perform new operating system installations as

[1]Hewlett Packard
[2]http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html
[3]http://www.kernel.org/

well as all the services installation and configuration. Nevertheless, this virtualization approach creates a layer of abstraction to the hardware level: to the virtual machine, the used hardware is completely transparent.

Operating System

As for the operating system, we used the Community Enterprise Operating System (CentOS)[4], which is a linux[5] distribution (kernel 2.6.18-274.7.1.el5.).

The CentOS distribution belongs to the class of enterprise operating system, it is free, open source, has a developing team of core developers and has a huge and active community of supporters (system and network administrators, enterprise users, managers, etc) that contributes to the operating system maintenance and evolution.

Since we were planning our framework as an web application with multiple interfaces we decided to deploy an hardware/software platform that gives guarantees in terms of working as a server (in this case as an http server). The CentOS offers this guarantees because is widely use, tested and geared to act as a server machine. We decided to create this environment from scratch in order to have a dedicated platform to use during the project.

Services, Packages & Drivers

Besides the basic services that ensure the minimal conditions for the server to work, all others were stopped, some even not installed and others removed afterwards. Nevertheless, some services (daemons[6]) are essential to the framework development and implementation: network, httpd, mysqld, smb and sshd.

The web platform uses the combination Apache (Hypertext Transfer Protocol (HTTP)) server[7]+PHP[8]+MySQL[9]. The Apache HTTP server is the one responsible

[4]http://www.centos.org/
[5]http://www.linux.org/
[6]A program or process that runs in background and provides some sort o functionality/service.
[7]http://projects.apache.org/projects/http_server.html
[8]http://www.php.net/
[9]http://www.mysql.com/

for answering to the HTTP requests. The programming language for development chosen was PHP: Hypertext Preprocessor (PHP). The experience with this programming language was one of main factors that determined this choice. MySQL gives support to the web site in terms of users and access control. The MySQL engine plays also another role since it is used to accommodate the reconstruction (SQL rebuild) of databases from the DBML in the archive.

The connection and access to databases is done via Open Data Base Connectivity (ODBC). This technology offers an API to interact with several Database Management Systems (DBMSs). Specifically, we used the unixODBC[10] Project. Since the ODBC Application Programming Interface (API) was firstly designed for Microsoft Windows[11] systems the unixODBC project intends to offer a standard for other systems, namely Linux distributions. Having this tool installed on the system we then can install and enable the specific drivers for different DBMSs (eg.: MS SQL Server, MySQL, PostgreSQL, etc). To connect to Microsoft Access we also needed to install the MDB Tools[12].

Figure 5.1 tries to give a graphical overview about the system global architecture. It is possible to observe the abstraction and independency between the virtual machine and the hardware. Although, there is in fact a dependency with VMware[13] software but, has we already mentioned, this was only a strategy to obtain more flexibility in terms of moving the "server" that supports the project from one place to another.

5.1.2 OAIS Integration

After the introduction and the overview analysis preformed on chapter 2 about the OAIS reference model, we now establish some integration aspects between that reference model and the developed system. The expression "OAIS integration" means that the main ideas, concepts and environment proposed by the OAIS, are correlated to the FrepDB system.

[10]http://www.unixodbc.org/
[11]http://www.microsoft.com/
[12]A set of libraries and utilities for reading Microsoft Access Database (MDB) files.
[13]http://www.vmware.com/

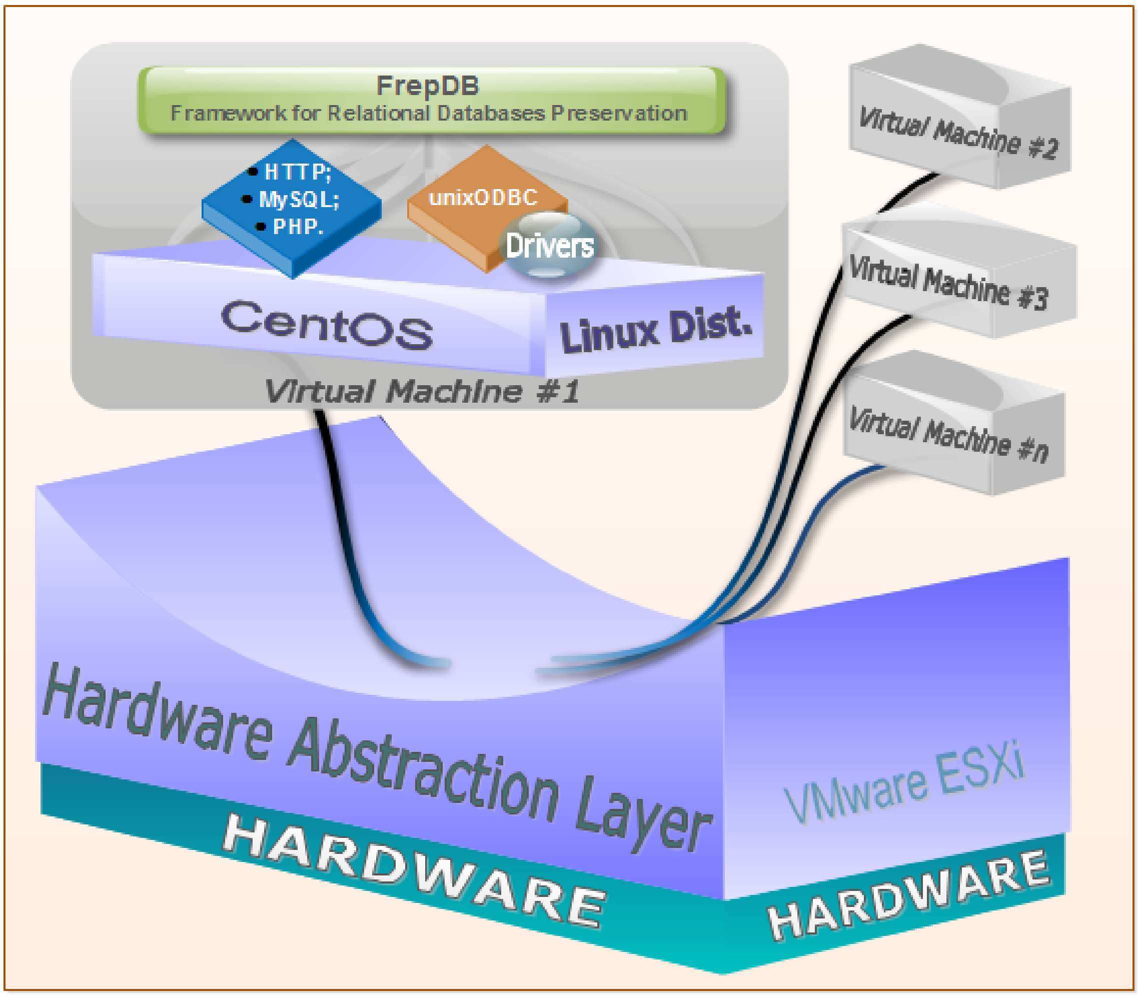


Figure 5.1: **FrepDB** – Framework for Relational Databases Preservation

Right from the start, the projected archive follows the OAIS main orientations: ingestion, archiving and dissemination. The framework was projected and developed under these main functional axes guidelines. Following what we stated in earlier chapters, the process of preservation is based in information packages as it is advocated by the reference model: Submission Information Package (SIP), Archival Information Package (AIP) and Dissemination Information Package (DIP). Focusing on the information package (P) for archiving, the OAIS states that it should be composed by the Content Information – c – plus the Preservation Description Information (PDI) – p. The conceptual structure of OAIS information packages were introduced in chapter 2, figure 2.4.

Before moving further, it is important to clarify what means the Descriptive Information – i. In the framework this is the information that describes the database that is archived under a specific package (AIP). This component consists on information that enables searching, in a collection of packages, for the one package that satisfies the consumer requests (database of interest). Technically this information is included in the ontology (OWL document) and is available to the consumer. The Descriptive Information about the whole package is included as a data property in the ontology. Thus, the composition of the whole package is as follows:

- **P={$\boldsymbol{c}$, $\boldsymbol{p}$, $\boldsymbol{i}$}**
  - $\boldsymbol{c}$ – Content Information
  - $\boldsymbol{p}$ – Preservation Description Information
  - $\boldsymbol{i}$ – Descriptive Information

Now, scrutinizing the information package, and entering inside the package, there is a division between the Content Information and the PDI, as it was explained. The PDI consists of information to identify the Content Information and the environment in which it was created. The PDI can be divided in four parts: Provenance, Context, Reference and Fixity. The OWL file supports this (technical) metadata as properties of a class in the ontology. Provenance, context and fixity are defined as data properties (dp) and the reference is defined as an object properties (*op*) of the ontology (metadata class). During the process of ingestion this structure is respected and assimilated by the framework.

Finally and establishing a relation with OAIS orientations, the Content Information in FrepDB system packages is defined as: Information Object = DBML file, *io*; Representation Information = OWL file, *ri*. The Information Object here is the relational database for preservation (archival), reflected in the DBML document (data and structure). The OWL provides the knowledge base ("Representation of Information"), also recommended by the OAIS reference model, to understand the Information Object (the relational database). By using the ontology approach to capture the database semantics, the OWL generated code tries to reflect the concepts, properties and relations that exist in the database at an higher level of abstraction, thus providing a knowledge base to understand the RDB data (information).

- $i=\{dp1\}$
- $p=\{dp2, dp3, ..., dpn\ , op1, op2, ..., opn\}$
- $c=\{io, ri\}$

The Descriptive Information corresponds to descriptive metadata and the PDI can be seen as technical metadata and are both included in the OWL document. From the composition of the Representation Information (*ri*) plus the Information Object (*io*) the Content Information is obtained. Part of the OWL file and the DBML document respectively.

The Representation Information is the knowledge base (advocated by the OAIS) to understand the Information Object which is under the process of preservation. This knowledge base is intrinsically related to the database semantics. As previously proposed, the OWL document gives support to this semantics and is adopted to formalize the knowledge associated to the conceptual model of the database. Figure 5.2 presents the package structure used in the framework (FrepDB):

- *i* – Descriptive Information

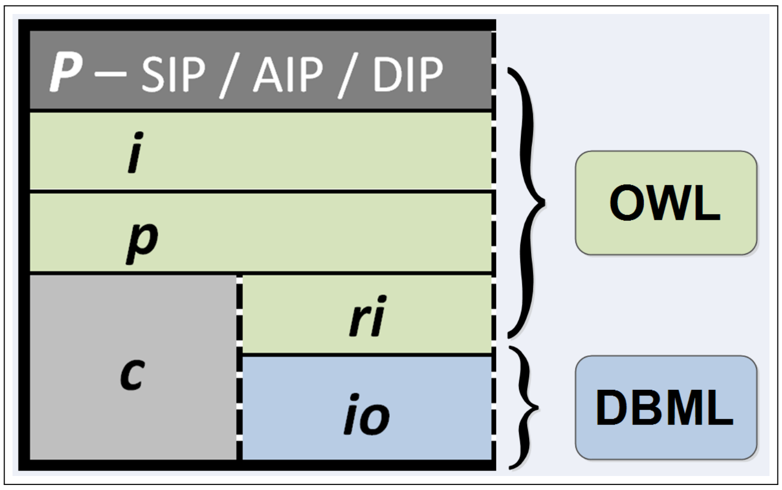


Figure 5.2: FrepDB Package

- *p* – Preservation Description Information
- *c* – Content Information
  - *ri* – Representation of Information
  - *io* – Information Object

Another important feature pointed by the reference model is that, when someone is producing data, there exists a sort of privileged access to information about that data. In the developed framework it is also possible (later detailed), a kind of data preparation. So we adopt this orientation, from the OAIS reference model, by enabling the possibility for the producer of the SIPs to, for example, alter the name for the classes in the generated OWL (ontology).

Another example is the fact that during the process of ingestion the framework can establish two stages (2 submission sessions, also covered in the OAIS recommendations): one for the DBML production and another one for the OWL generation. The combination of these XML documents is indeed the package of information to be archived (AIP).

The SIP validation phase consists on checking the schema of the XML documents. The documents must be well formed and valid against XML schema files (one for the DBML and another for the OWL). These validation processes is performed by the PHP `DOMDocument::schemaValidate` method. Nevertheless, the consistency of the generated ontology also acts as part of the validation process. If a consistent and reasonable ontology is not obtained, then the SIP does not conforms with the agreement and therefore the AIP cannot be obtained for archiving.

On the other extreme of preservation "lifecycle", is the process of dissemination of preserved databases. The consumers may request for a specific database by searching over the descriptive metadata (Descriptive Information) and by doing so select the AIP of interest which contains the desired RDB. Then, the framework deploys an ontology browser capable of navigating through the OWL document, its classes, properties, and individuals. This ontology browser was also developed in PHP and uses XML Path Language (XPath) queries, over the OWL file, to preform the browsing. Individuals have a specific property for linking a specific individual to its DBML correspondent data. This property (DBML_link) is the XPath query to the individual in the DBML document (primary key).

The dissemination process also enables the reconstruction of the database using the framework functionality of producing SQL statements. This SQL generated code can then be executed in a DBMS (tested only on MySQL) enabling the rebuild of the original RDB [Freitas, 2008].

5.1.3 Mapping Process

The primary goal is to preserve a certain database ensuring that the preserved information will be accessible and understandable for large periods of time (or even undefined time periods). We are referring to databases on which no more transactions are expected (frozen databases) or in other cases database snapshots.

The idea is to produce a package (SIP) that will be archived for preservation! To do this, as already mentioned, the package includes the structure, data and as much knowledge as possible about the conceptual model of the database. The mapping process from RDBs into DBML (Data & Structure) is detailed and analyzed in the previous work [Freitas, 2008], so now the main focus is the mapping process from RDBs into OWL.

Lets start by enumerating the properties of RDB that are mapped into the ontology (OWL):

- **Table:** names;
- **Attribute:** names and data types;
- **Keys:** primary keys, foreign keys (relationships between tables);
- **Tuples:** data;

Following the previous work, this elements are extracted from the database into multidimensional arrays. Code block 12 shows the array structure. These arrays are the starting point for both mappings (DBML & OWL). One of them (vector) has only one dimension – `tables` –, another one – `p_keys` – has two dimensions and the other three arrays – `columns, f_keys, tables_data` – have three dimensions each.

Nevertheless, the mapping process from RDBs to OWL includes a preliminary stage where it is possible to a) define which tables will be addressed by the process and b) alter the name of the class generated table, i. e., the original table name will not give name to the mapped generated class mandatorily. We offer the possibility of defining different names for the classes. This is an optional feature and was thought to provide some human intervention in order to define more accurate names to the classes in consonancy with the concepts they represent. Also in the preliminary stage the producer must include metadata to be preserved along with the information package. The technical metadata (p – PDI) and the descriptive information (i) are both established during this stage.

In the next table (Table 5.1) we summarize the mapping between the RDB and the ontology. From the conceptual mapping approach and some DBMSs heuristics we start to manually convert a relational database (first case study database – "CoursePrograms") into OWL using Protégé [Protégé Project, 2012][14]. This strategy consisted on defining, in Protégé, all the classes, data and object properties, and also some individuals to reflect the first case study database.

[14]http://protege.stanford.edu

Code Block 12 : Multidimensional Array Structure

```
tables = Array{ [1] => t1, ... , [n]=> tn }

columns = Array{
    [t1] => Array{
        [a1] => Array{ [Name] => 'a1_name', [Type] => 'a1_type' },
         ...,
    [an] => Array{ ... }},
    ...,
    [tn] => Array{...}}

p_keys = Array{
        [t1] => Array{ [a1] => 'pk_t1', ..., [an] => 'pk_t1' },
        ...,
        [tn] => Array{...}}

f_key = Array{
        [t1] => Array{
                [a1] => Array{ [pk_table] => 'tref', [pk_column] => 'tref.aref'},
                ...,
                [an] => Array{...}},
        ...,
        [tn] => Array{...}}

tables_data= Array{
        [t1] => Array{
                [1] => Array{ [a1]=> 'a1_data', ..., [an] => 'an_data'},
                ...,
                [m] => Array{...}},
        ...,
        [tn] => Array{...}}
```

The algorithms were then designed based on the mapping with the help of the code analysis (Protégé – OWL/XML format). We conceived an algorithm set capable of generating the OWL/XML code for i) classes, ii) properties and iii) individuals definition. These definitions include triples such as *subject, predicate, object* (introduced in the previous chapter, in the **XML: RDF & OWL** subsection).

RDB	OWL
Tables if(#att = #pkeys = #fkey)	→Classes →Object Property
Foreign Keys	→Object Properties
Primary Keys	→Individuals Identification
Other Attributes	→Data Properties
Tuples	→Individuals
	▶Inverse Obj. Prop. Generation ▶Functional Obj. Prop. Definition ▶Disjoin All Classes

Table 5.1: Mapping Table

Concerning the code development in PHP, an abstraction class was created for database connection and import – `db.php`. This class has the mission to establish a connection to a certain DBMS and to a specific database, extract its structure (database schema) and its data, to finally populate the arrays with the extracted information. Two other classes were developed, one for DBML generation and other for OWL conversion – `dbml.php, db_owl.php`. For the SQL reconstruction from the DBML another class was programmed – `sql.php`. Concerning the ontology browsing some individual functions were created and adapted.

5.1.4 From RDB to OWL

Now lets move specifically to the conversion between databases and ontologies, based on the mapping process (mapping algorithms).

Relegating other technological aspects, the conceived algorithms to preform the mapping process were indeed a complex stage of the implementation work. These algorithms enable the conversion from RDBs into OWL. The developed prototype enables the connection to a Data Source Name (DSN), extracts the data/information needed and gives the initial possibility of selecting the tables of interest (for conversion) and alter the names of the classes to be generated as it was already referenced.

It was assumed that the source database is normalized (Third Normal Form (3NF)) in order to produce an ontology of greater value in terms of semantic information that can be preserved. The non normalization of the source databases may result in inconsistencies on the generated ontologies. These cases were not exhaustedly tested since we focused on normalized relational databases.

For each `table` on the database we define a **class** on the ontology (Table 5.1) with the exception of those tables where all attributes constitute a composed primary key (combination of foreign keys). These link tables used in the relational model to dismount a many-to-many relationship, are not mapped to OWL classes, instead they give origin to **object properties** in the ontology. These object properties have on their domain and range the correspondent classes (database tables) involved in the relationship. For each attribute (foreign key) in the table an object property is defined. Code block 13 gives an overview of the algorithm portion that performs these tasks.

For simplicity purposes, the pseudo-code presented here shows that the algorithm crosses the columns array processing its elements with the exception of the last one (and it works perfectly for link tables with two columns), however if the link table establish a ternary (or higher) relationship[15] between three (or more) tables, the array must be crossed processing all elements. Since the last element can not point to the "next" it must point to the first array element.

Another important note is the fact that the algorithm was only tested in databases where there were no link tables with foreign keys composed by more than one attribute. The algorithm current structure does not support these particular cases.

In order to fully explain the algorithm pseudo-code, these main portions are also divided into small pieces of code and are detailed. Fragments of the OWL produced code are also presented for better understanding of conversion processes.

The first step is to determine for each table in the *tables* array which are mapped

[15]A ternary relationship, for example, means that a table possesses three attributes composing the primary key and each of them individually is a foreign key to its respective table

Code Block 13 : Algorithm – Classes and Non Classes

```
// Classes (tables) & ObjectProperties (link tables - non_classes)

FOREACH [ table ]
  IF [ ( |columns[table]| = |p_keys[table]| )
       AND ( |p_keys[table]| = |f_keys[table])| ) ] THEN
    non_class[] = table
    FOREACH [ columns[table] - 1 ]

    NEW 'ObjectProperty'
      Property_Description = 'is_' + f_keys[table][columns[table]][pk_table] + '_Of'
      Domain = f_keys[table][columns[table]][pk_table]
      Range = f_keys[table][next(columns[table])][pk_table]

    NEW 'ObjectProperty'
      Property_Description = 'has_' + f_keys[table][columns[table]][pk_table]
      Domain = f_keys[table][next(columns[table])][pk_table]
      Range = f_keys[table][columns[table]][pk_table]

    NEW 'InverseObjectProperties'
      Property_Description = 'is_' + f_keys[table][columns[table]][pk_table] + '_Of'
      Property_Description = 'has_' + f_keys[table][columns[table]][pk_table]

        END FOR
  ELSE
    class[] = table
  END IF
END FOR
```

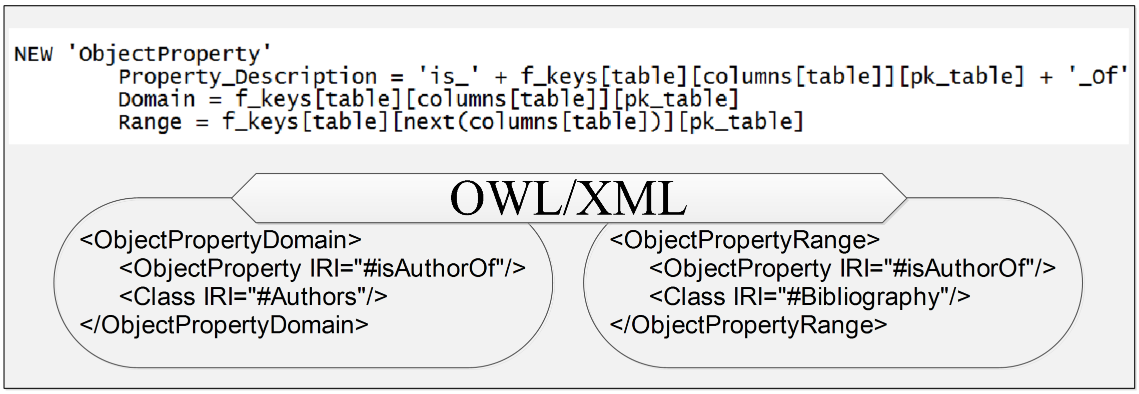
into classes and which are mapped into object properties. To do that a *foreach* loop is utilized. Then with the aid of the *columns*, *p_keys* and *f_keys* arrays, the number of attributes, for a given table, is compared with the number of primary keys of that same table. It is also verified if the number of primary keys is equal to the number of foreign keys. In case that these two comparisons are true it means that the table in question is a link table, i.e., a table that will not be mapped into class, but instead, is mapped into an object property; if one of them fails the table is mapped directly into a class in the ontology (Code Block 13). The generation of the OWL/XML code for the tables mapped into object properties is done inside the *if* statement. In the else clause it is only defined which tables are to be mapped into classes.

Code block 14 shows an example (instance) of a table *AuthorsBibliography* which is a link table and where the number of columns is equal to the number of primary keys, equals the number of foreign keys.

Code Block 14 : Arrays Instances

```
columns = array (
        ...
        [AuthorsBibliography] => Array (
            [IDAutor] => Array ( [Name] => IDAutor ... )
            [IDLivro] => Array ( [Name] => IDLivro ... )
            )
        ...
        )
p_keys = array (
        ...
        [AuthorsBibliography] => Array (
            [IDAutor] => PK_AutoresLivros
            [IDLivro] => PK_AutoresLivros )
            )
        ...
        )
f_keys = array (
        ...
        [AuthorsBibliography] => Array (
            [IDAutor] => Array ( [pk_table] => Authors [pk_column] => IDAutor )
            [IDLivro] => Array ( [pk_table] => Bibliography [pk_column] => IDLivro )
            )
        ...
        )
```

This table transforms one N-to-M (between table *Authors* and table *Bibliogra-*



```
NEW 'ObjectProperty'
       Property_Description = 'is_' + f_keys[table][columns[table]][pk_table] + '_of'
       Domain = f_keys[table][columns[table]][pk_table]
       Range = f_keys[table][next(columns[table])][pk_table]
```

```
<ObjectPropertyDomain>
    <ObjectProperty IRI="#isAuthorOf"/>
    <Class IRI="#Authors"/>
</ObjectPropertyDomain>
```

```
<ObjectPropertyRange>
    <ObjectProperty IRI="#isAuthorOf"/>
    <Class IRI="#Bibliography"/>
</ObjectPropertyRange>
```

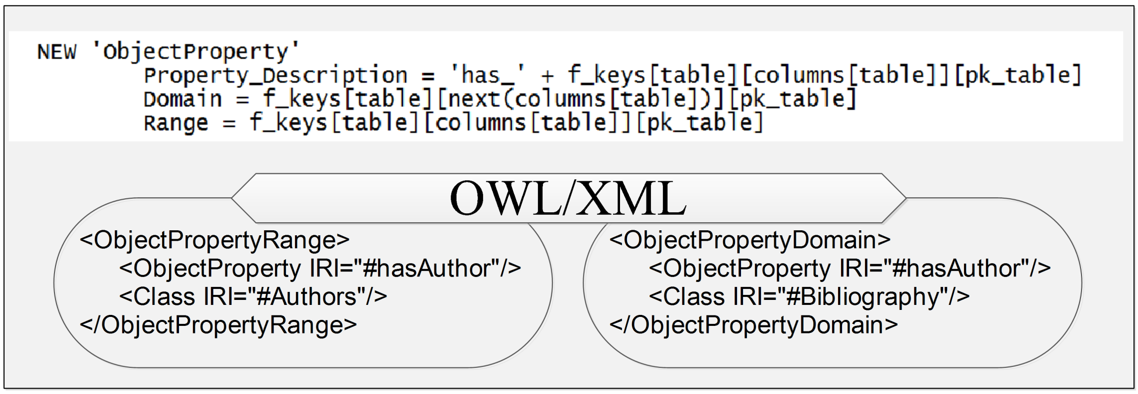
Figure 5.3: Object Property definition & OWL/XML Generated Code – Domain - Range

phy: *"(...) a book may have many authors; an author may also write many books (...)"*) relationship in two 1-to-N relationships. The relational model imposes that these N-M relationships can not exist, and therefore a table is created to ensure that the database respects the relational model. However at a semantic level this technical issues are not that relevant. The important is that the ontology reflects the relation between classes through one or more object properties.

Looking again to code block 14, it is possible to analyze the arrays behavior when instanciated. This example shows part of the instance and corresponds only to the *AuthorsBibliography*, table, columns, primary and foreign keys which are used to generate the new object properties, its domains and ranges.

The link tables are mapped, not into one, but into two object properties. These two object properties have different names and are inverse to each other since the domain of one object property *OP1* is the range of the other object property *OP2* and vice-versa. Figure 5.3 shows the code responsible for generating the "*is_..._of*" property and also presents the OWL code generated. From the combination of several portions of OWL like this one and others, the entire OWL document and the correspondent ontology are created.

The pseudo-code presented in figure 5.4 represents the nest steps needed to define another object property: the "*has_...*" property which also establishes a relation between two classes. Comparing the OWL code generated for these two object properties (*is_Author_of* and *has_Author*) it is possible to clearly see that the two

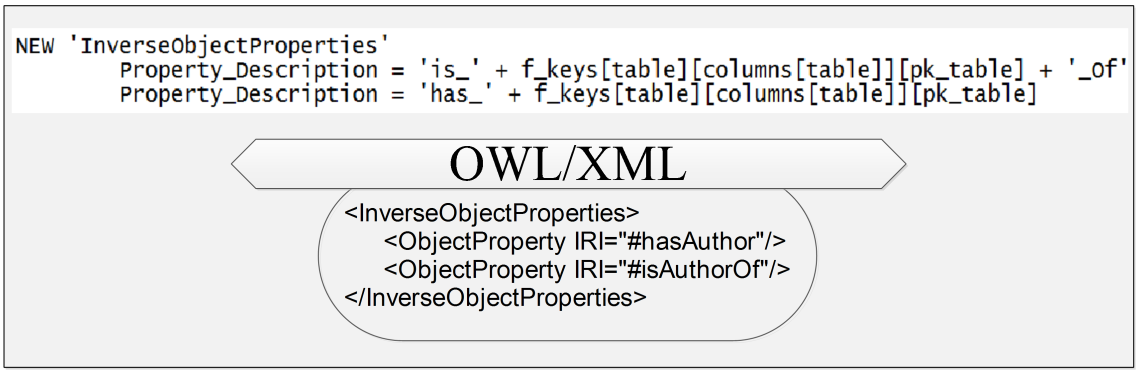


```
NEW 'ObjectProperty'
        Property_Description = 'has_' + f_keys[table][columns[table]][pk_table]
        Domain = f_keys[table][next(columns[table])][pk_table]
        Range = f_keys[table][columns[table]][pk_table]
```

```
<ObjectPropertyRange>
    <ObjectProperty IRI="#hasAuthor"/>
    <Class IRI="#Authors"/>
</ObjectPropertyRange>
```

```
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasAuthor"/>
    <Class IRI="#Bibliography"/>
</ObjectPropertyDomain>
```

Figure 5.4: Object Property definition & OWL/XML Generated Code – Range - Domain



```
NEW 'InverseObjectProperties'
        Property_Description = 'is_' + f_keys[table][columns[table]][pk_table] + '_of'
        Property_Description = 'has_' + f_keys[table][columns[table]][pk_table]
```

```
<InverseObjectProperties>
    <ObjectProperty IRI="#hasAuthor"/>
    <ObjectProperty IRI="#isAuthorOf"/>
</InverseObjectProperties>
```

Figure 5.5: Inverse Object Property definition & OWL/XML Generated Code

same classes (*Authors* and *Bibliography*) are used as domain and range for both object properties. The domain class and the range class of the first object property will be the range class and domain class of the second object property, respectively.

This indicates that we are in the presence of two inverse object properties, and therefore another portion of the algorithm is responsible the generate the necessary OWL code to define two inverse object properties in the ontology. Figure 5.5 shows the pseudo-code for the generation of inverse object properties and also presents the generated OWL code.

After processing all the link tables, generate all the associated object properties and have defined the tables to be mapped into ontology classes, the first part of the algorithm is terminated.

By extracting all the information about the schema of the database (structure), the information about the tables and the data within the tables, into arrays (multidimensional arrays), it is possible to implement these type of algorithms for the generation of the OWL (XML) code. The whole OWL generated document gives form the database ontology. The same methodology is adopted to generate the remaining parts of the ontology.

The second part of the algorithm (Code Block 15) generates the main OWL code for the ontology structure definition (classes, object and data properties). The `foreign keys` of the tables mapped directly to OWL classes also give origin to **object properties** of the correspondent OWL classes (tables). The other `attributes` of the several tables are mapped to **data properties** within the analogous OWL classes with the exception of the attributes that are foreign keys. In short, the table is mapped into an ontology class, and all their attributes are mapped into properties of that same class: a) foreign keys give origin to object properties and b) the other attributes (columns) give origin to data properties.

In this part of the algorithm (Code Block 15), inverse object properties are generated for all relationships among the classes. So, each foreign key, will have two inverse object properties. In this pair of object properties, generated directly from one-to-many relationships (not link tables), it is possible to define one of the object properties as functional (in one direction). These functional properties have the objective of maintaining the consistency of the relational database model in the generated ontology.

To conclude the conversion of the table structure into OWL, an important data property is also created. This extra data property, we called it ***XPath_to_DBML*** link. This allows the connection between a class individual and the correspondent DBML element which contains the tuple (data) within the original table. This last data property is generated to support the linking between OWL (the semantics) and DBML (the data).

Nevertheless, the `tuples` of the different tables are also mapped to **individuals** in the ontology. They are identified by the associated primary key in the correspondent tuple of the table. This means that a tuple in a database table is mapped to an individual of a class in the ontology. Code block 16 details this third part of the algorithm responsible for the generating the assertions about the ontology

Code Block 15 : Algorithm – Structure Generation

```
//   Sub Classes of Thing & Disjoint all & Object and Data Properties

class_disjoint[] = class
FOREACH [ class ]
  NEW class 'SubClassOf' owl:Thing
  FOREACH [ class_disjoint ]
    IF  [ class IN class_disjoint ] THEN
      NEW 'DisjointClasses'
        Class_Description = class
        Class_Description = class_disjoint
    END IF
  END FOR
  pop(class_disjoint)

  FOREACH [ f_keys[table] as fk ]
    NEW 'ObjectProperty'
      Property_Description = 'is_' + fk['pk_table'] + '_Of'
      Domain = fk['pk_table']
      Range = class

    NEW 'ObjectProperty'
      Property_Description = 'has_' + fk['pk_table']
      Domain = class
      Range = fk['pk_table']

    NEW 'InverseObjectProperties'
      Property_Description = 'is_' + fk['pk_table'] + '_Of'
      Property_Description = 'has_' + fk['pk_table']

    NEW 'FunctionalObjectProperty'
      Property_Description = 'is_' + fk['pk_table'] + '_Of'
  END FOR

  FOREACH [ columns[table] as table_data ]
    IF [ f_keys[table][table_data['Name']]['pk_column'] != table_data['Name'] ] THEN
      NEW 'DataProperty'
        Property_Description = 'has_' + table_data['Name']
        Domain = class
        Range = data_type
    END IF
  END FOR

  // Extra Data Property -- XPath_to_DBML link
  NEW 'DataProperty'
    Property_Description = class + '_XPath_to_DBML'
      Domain = class
      Range = xsd:string
END FOR
```

individuals.

- **Class Assertion:** In each tuple of a table mapped directly into an OWL class, a new class assertion is made about an individual, identifying him by its primary keys attributes.

- **Data Properties Assertion:** For all the attributes in the original table and associated to each tuple (individual), a new data property assertion is made, if the attribute is not a foreign key.

- **Object Properties Assertion:** If the attributes are foreign keys, then an object property assertion is established since it means that we are in the presence of an attribute that was mapped into an object property (individual A implicates an individual B through an object property P).

Concerning the data properties assertions, as we will see later, the incorporation of all the tuples data in the ontology (individuals), can be optional. In other words, all the individuals are mapped and identified but their data properties may or may not be filled (include data). This option is offered by the framework because through the *XPath_to_DBML* link data property it is possible to reach all the tuples data since this information is also stored in the DBML document. The primary keys are used to construct the XPath query to the individual in DBML document. This XPath query is stored the *XPath_to_DBML* extra data property. Therefore, a data property assertion (*XPath_to_DBML*) is performed for individuals (Code Block 16).

It should be noted that for tables not mapped into OWL classes (*non_class*), it is also necessary to perform object properties assertions (Code Block 17). It is the case already detailed where those link tables gave origin to object properties.

For all object properties, the algorithm only performs assertions for one of the two inverse properties. If in the inverse pair of object properties exists one property that is functional, this is the one to be defined; if not, the generated object property assertion is irrelevant. Later, we will see that the these not performed assertions are inferred when classifying the ontology. Some of the achieved inferences, evaluation stages and tests performed are detailed in the next section.

Code Block 16 : Algorithm – Individuals (tables mapped into classes)

```
//   tuples -> Individuals   //

FOREACH [ class ]
  FOREACH [ tables_data[table] as tuple ]
    primary_key = class
    FOREACH [ p_keys[table] as pk)
          primary_key = primary_key + pk
    END FOR

    NEW 'ClassAssertion'
      Class_Description = class
      NamedIndividual = primary_key

    FOREACH [ tuple as kt=>t ]
      IF [ NOT [ kt IN array_keys(f_keys[table]) ] ]
        NEW 'DataPropertyAssertion'
          DataProperty = class + '_has_' + kt
          NamedIndividual = primary_key
          Literal = t
      ELSE
        NEW 'ObjectPropertyAssertion'
          ObjectProperty = f_keys[table][kt]['pk_table']
          NamedIndividual = primary_key
          NamedIndividual = f_keys[table][kt]['pk_table'] + '_' + t
      END IF
    END FOR

    // XPath_to_DBML link Data Property Assertion
    xpath_to_dbml="/DB/DATA/" + class + "/" + class + "-REG[" + primary_key + "]"
    link = "&lt;a href=&quot;/OWL_DBML.php?AIP=" + DBML_FILE +
               "&amp;XPATH=" + xpath_to_dbml
    NEW 'DataPropertyAssertion'
      DataProperty = class + '_XPath_to_DBML'
      NamedIndividual = primary_key
      Literal = link

  END FOR
END FOR
```

Code Block 17 : Algorithm – Object Properties Assertions from Link Tables

```
//   tuples -> ObjectProperties (link tables)   //

FOREACH [ non_class ]
  FOREACH [ columns[table] - 1 ]
    FOREACH [ tables_data[table] as tuple ]

      NEW 'ObjectPropertyAssertion'
        ObjectProperty = f_keys[table][columns[table]]['pk_table']
        NamedIndividual = f_keys[table][next(columns[table])]['pk_table'] +
                    '_' + tuple[f_keys[table][next(columns[table])]['pk_column']]
        NamedIndividual = f_keys[table][columns[table]]['pk_table'] +
                    '_' + tuple[f_keys[table][columns[table]]['pk_column']]

    END FOR
  END FOR
END FOR
```

5.2 Evaluation

The first evaluation stages of the system started during the developing and implementation phases. There was a necessity of performing earlier evaluations because some issues had to be addressed during the development, namely, the production of consistent ontologies. Nevertheless, the first step consisted on establishing the manual definition of an ontology for the first case study database. It was possible to verify, for example, the production of inferences such as those mentioned in the case of the inverse object properties. This preliminary evaluation consisted on the first test to the system; this, allow us to determine how the system will behave concerning the generation of consistent ontologies.

The main goal was to produce a reasonable and consistent ontology capable of reflecting the database semantics. After the conceptual definition of the algorithms, they were implemented in PHP[16] programming language and tested with real databases.

[16]http://www.php.net/

5.2.1 Methodology

The evaluation methodology consisted on testing the framework with some databases (case study databases) in order to determine inconsistencies in the processes of ontology generation, SIP integration and ingestion (AIP) and also in the process of accessing the archive, browsing the ontology and obtain "preserved" information (dissemination process). However, it is important to notice that a significant part of the tests were performed over the first case study database. Only when consistent results were obtained with first case study, we move forward and test the algorithms with other databases. So, even when those results concerned only parts of the ontology (eg.: the classes or the properties), tests over different databases were also made.

Of course, before any of these stages, we perform tests on connecting (via ODBC) to different DBMSs. Different drivers (unixODBC project) to MySQL, MS SQL Server and MS Access databases were installed and tested.

A specific database on a specific server machine was selected, the connections were tested (with the necessary credentials) and then the framework deployed the running tasks of extracting the data and structure into the multidimensional arrays. This was the framework first test! Acquiring the source database data and structure.

The next stages consisted on running the algorithms to generate the DBML and OWL files, validate them and store those documents together on the file system (same directory). If desired or necessary, archive them as "`tar.gz`" files.

Figure 5.6 elucidates about ingestion, archival and dissemination processes.

If we remember the framework, more specifically the mapping process, it bases itself on the same multidimensional arrays extracted from the source RDB; so we have made some tests where we extract the data into the arrays and then serialize them (using PHP serialize function) to store them in the filesystem. With this technic the mapping process starts always from those serialized arrays (using PHP unserialize function) which reduces the DBML and OWL generation substantially. By dividing the mapping and conversion processes into these two phases, it was possible to test the algorithms directly from (serialized) arrays, not having to always

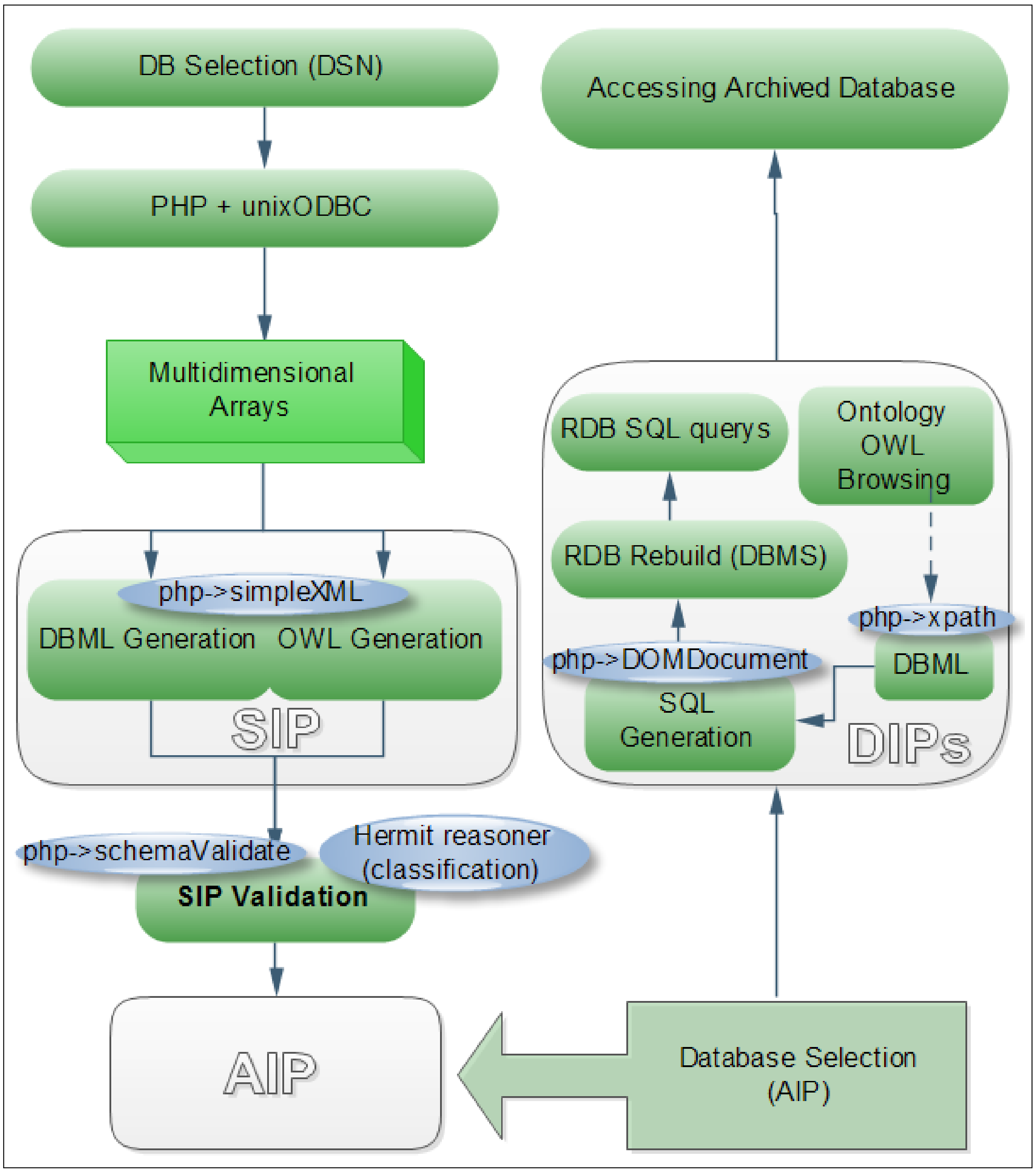


Figure 5.6: Archival and Dissemination flowchart

connect and extract the source database. However, if changes have been made to the source database or if a different database is selected, there is the necessity to connect and extract the database into the arrays again.

It was also part of the methodology the usage of the Protégé ontology editor software to load the owl generated files. With this technic it was possible to overview the ontology, its classes, properties and individuals and compare the results with the original database data and structure. XPath queries were also used to compare XML elements and attributes of the DBML files (the ones that reflect exactly the structure and data of the original database) with XML elements and attributes of the OWL files (the ontology mapped from the original database).

5.2.2 Tests and Results

As a first battery of tests, the algorithms were tested with the first case study database. As it was already mentioned, some adjustments were necessary in order to achieve a consistent ontology. After some programming refinements, we successfully use the HermiT 1.3.3 reasoner[17] to classify the ontology. The inverse "object properties assertions" that the algorithm does not generate for the individuals were inferred. Some equivalent (and inverse functionality) object properties were also inferred.

Figure 5.7 shows the database relational (logical) model and the ontology conceptual approach. It is possible to notice that the link tables in the relational model disappear in the ontology (are mapped into object properties). All the other tables are directly mapped into OWL classes.

The arrows represented in the OWL side of the figure (Fig. 5.7) correspond to the object properties that relate the different classes. There is a parallelism between these object properties and the relationships that occur in the database relational model. However, for each relationship in the RDB, two object properties are defined in the ontology. The data properties and the individuals are not graphical represented on the RDB / OWL image overview (Fig. 5.7).

[17]The Hermit OWL Reasoner; http://hermit-reasoner.com/

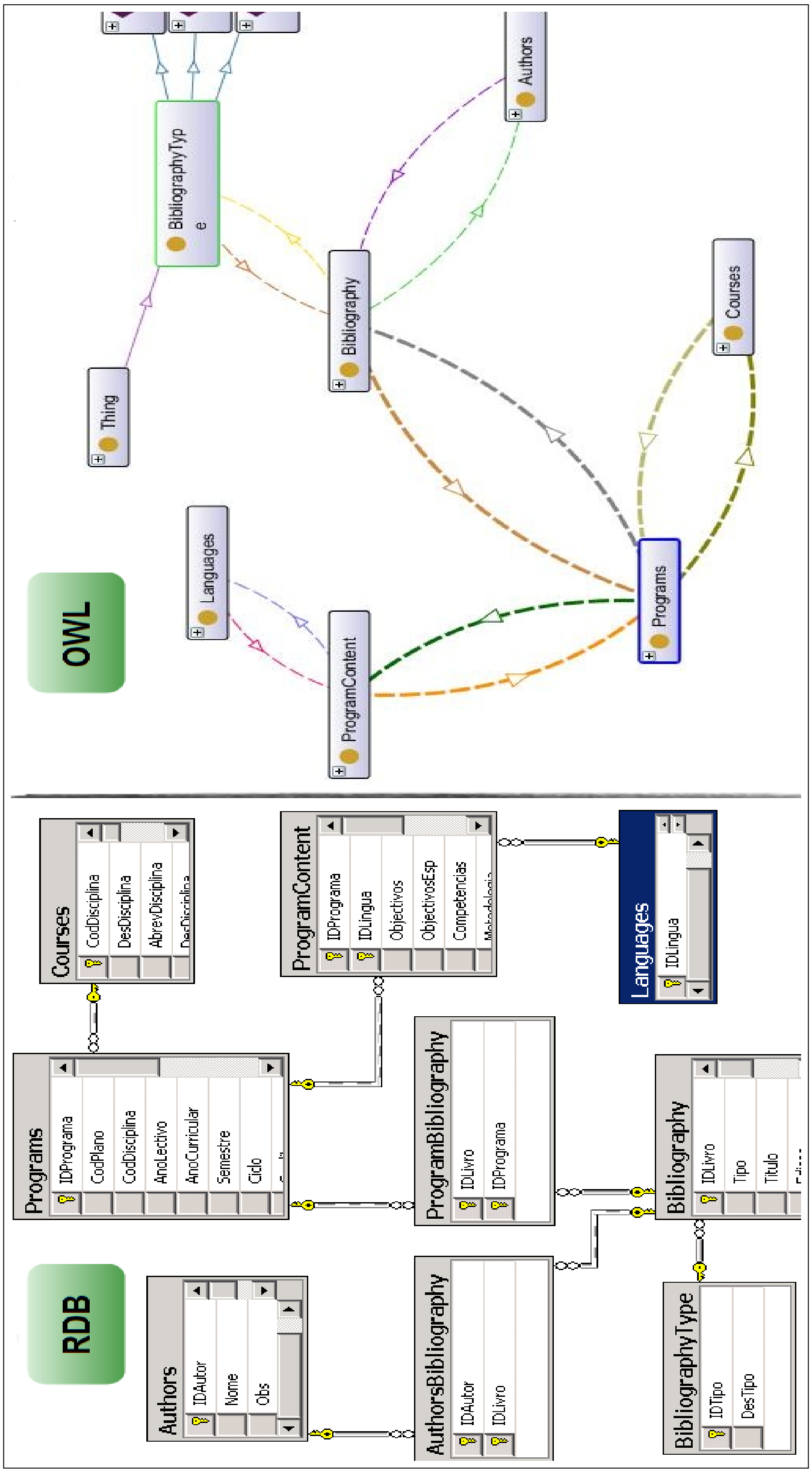


Figure 5.7: RDB Logical Model *vs* Ontology Overview

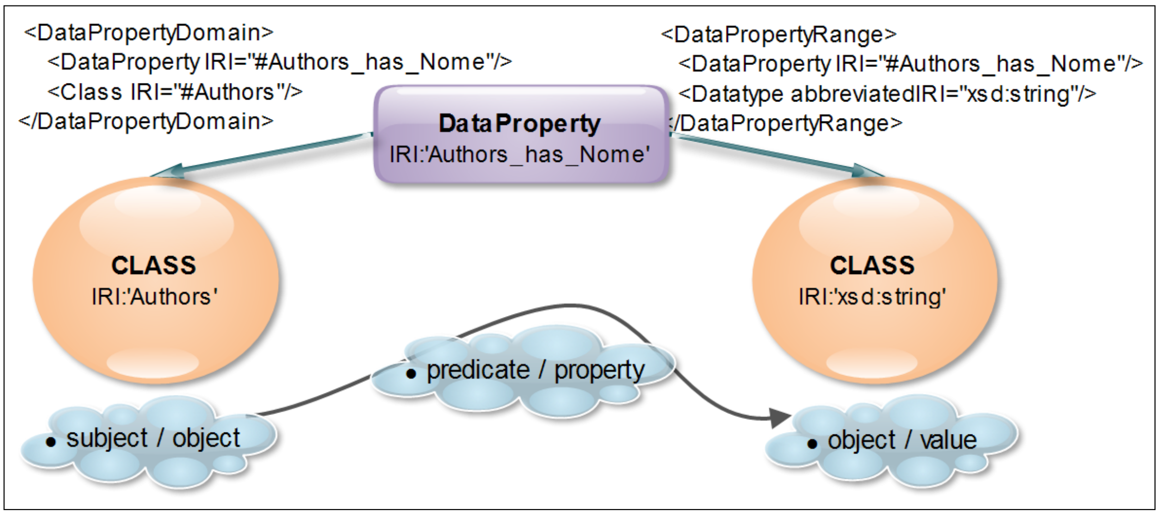
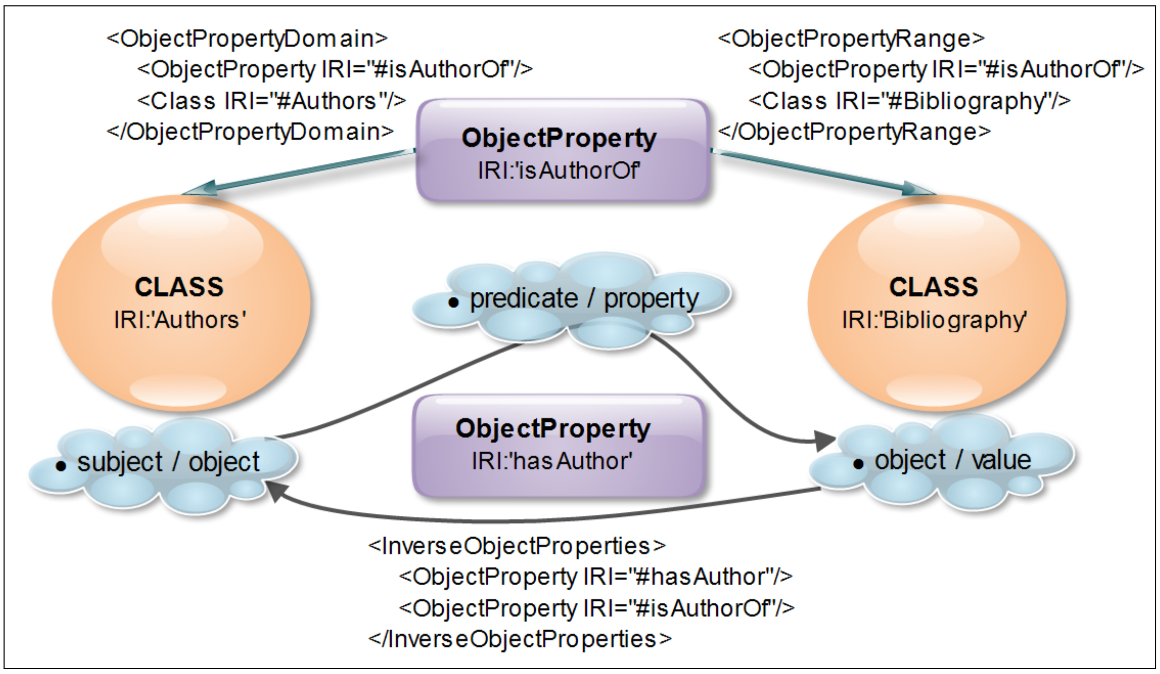


Figure 5.8: Results Portion: tables ”Authors” attribute mapping

For a more detailed representation about the object and data properties generated, two examples are presented in the following figures (one for the data properties and another for the object properties).

In Figure 5.8 the first example is presented. It shows a portion of the generated OWL document where we demonstrate the results of mapping a table attribute into a data property of a class. In this example we focus on the original table ”Authors” which is mapped to the class with the same name. The attribute of the table ”Authors” addressed here is ”Nome”. As it was possible to analyze in the algorithms (Code Block 15), the name of this data property is determined by the composition of: ’*table name*’ + ’has_’ + ’*attribute name*’. In the algorithm pseudocode, the table name is not referenced as being part of the name for the data property. Although in the PHP code it is there (interpretability purposes).

The second example focus on the relationship that exists between the two tables (”Authors” and ”Bibliography”) where the link table ”AuthorsBibliography” is mapped into an object property (and inverse object property) relating the correspondent mapped classes. Here it is clearly demonstrated that the triples are in the base of ontologies relations (Resource Description Framework (RDF) triples), and specifically, the presence of these triples as object properties in the relationship between two classes – *object, property, value* – or – *subject, predicate, object* [Zarri, 2005].



```
<ObjectPropertyDomain>
  <ObjectProperty IRI="#isAuthorOf"/>
  <Class IRI="#Authors"/>
</ObjectPropertyDomain>

<ObjectPropertyRange>
  <ObjectProperty IRI="#isAuthorOf"/>
  <Class IRI="#Bibliography"/>
</ObjectPropertyRange>

<InverseObjectProperties>
  <ObjectProperty IRI="#hasAuthor"/>
  <ObjectProperty IRI="#isAuthorOf"/>
</InverseObjectProperties>
```

Figure 5.9: Results Portion: tables "Authors" and "Bibliography" object properties

Although the algorithms generate inverse object properties, in the individuals object properties assertions the object properties (relating individuals in different classes) are only defined in one direction. When classifying the ontology the not generated assertions are inferred.

In these two examples it is also possible to view the XML code "behind" the ontology that the algorithms generate.

It was easier to classify ontologies without the individuals database data because protege and the reasoner have problems dealing with some encodings and specific characters.

An interesting test was made that consisted on generating an ontology for a database with some inconsistencies concerning the referential integrity. Intentionally some foreign keys values do not have their correspondent primary keys, i.e., some tuples where deleted from the primary keys table. When classifying the ontology, those individuals where inferred and assigned to the correct class where they were missing. Although, we have tested the mapping algorithms and the classifying process with a restrict number of databases, these kind of results indicate that other possible inferences can occur accordingly to the source database state of normalization. It is not feasible to test the framework with all types of unnormalized databases, so we concentrated on our test with consistent and normalized databases.

Specific Results

The next step consisted on testing the algorithms with other databases. We use one MS Access database, one MySQL database and two MSSQL Server databases (the maximum tables size were about tens of thousands records). All databases used in this research are from the University Lusíada information system. The results were very satisfactory because the algorithm achieved similar results to the ones obtained with the first case study database only with minor inconsistencies related with naming and encoding problems.

Although the majority of tests were performed over the first case study database, a significant number of test were also made over the other databases (about 50 extraction and generation processes for each database and the triple with the first case study). Most of them, preformed in the initial phases where the algorithms still needed some adjustments. Then, the other tests aim to increase performance and also to verify the behavior of the extraction and generation processes with different "hardware" resources. The usage of a virtual machine enables the possibility of changing the resources available to the operating system (services and applications) very easily.

The processing time is an issue directly related to the dimension of the database. The number of records in the database have a substantial impact in the processing time during the conversion. The number of tables, attributes and relationships (foreign keys) also produce a lower impact in the migration process duration. The main conclusions in terms of performances based on the several tests performed are the following:

- The number of records (quantity of data) in the database is what determines the duration of the conversion process and also the necessity, or not, of more hardware resources. The schema dimension and complexity of the database structure have only small impacts in performance;

- The inclusion of the data about the individuals (filling all data properties) in the ontology is what what the most penalizes the process (the OWL file is in average seven to eight times bigger then the DBML file);
- Without the inclusion of the data about the individuals, the OWL generation takes substantial less time (the generation file size is obviously smaller), still, the generation of the DBML file is faster and smaller;
- The increase of memory – RAM[18] – to the virtual machine has an impact in the conversions processing time (increasing memory decreased conversions time);
- It is necessary to adjust the Apache and PHP configurations to memory changes and processing durations, for better performance;
- The maximum size of database that the prototype was able to convert (DBML and OWL), with the available resources, contained about 100 000 records and generated OWL files with nearly 200 Megabytes.

The following table (Table 5.2) characterizes the different databases used to perform the several test.

DBMS	Database	Tables	Link Tables	Relationships	Attributes	Rows
MSSQL	CoursePrograms	11	2	8	67	1121
MS Access	Candidates	32	0	18	344	25845
MySQL	Individuals	6	0	4	24	104885
MSSQL	Intranet	52	1	38	318	10680

Table 5.2: Case Study Databases

From the four case studies, only the ”CoursePrograms” which we refer as the first case study database is addressed by its real name. This database was also the one used, during this thesis, to present some portions of generated code (DBML and OWL) and ontology mapped structure (schema).

After characterizing the structure of the original databases (case studies) used in this project, we then try to characterize the generated ontologies. The idea was

[18]Random Access Memory, a volatile data storage in computers

to perform some analysis to the number of database tables and generated ontology classes, the number of attributes in the database and data properties in the ontology and so one, for all the mapped elements. To perform this analysis a very simple method was utilized: since the framework produces a DBML file (which reflects exactly the relational model and data of the database) and an OWL document, and they are both XML format, XPath queries were used to query both files in order to count and obtain the numbers needed for comparison.

The XPath queries are:

- **DBML**
  - Tables: */DB/STRUCTURE/TABLE/@NAME*
  - Relationships: */DB/STRUCTURE/TABLE/KEYS/FKEY/@NAME*
  - Attributes (columns): */DB/STRUCTURE/TABLE/COLUMNS/COLUMN/@NAME*
  - Tuples (rows): */DB/DATA/*/**
- **OWL**
  - Classes: */Ontology/SubClassOf/Class[@abbreviatedIRI='owl:Thing']/preceding-sibling::*/@IRI*
  - Object Properties: */Ontology/ObjectPropertyDomain/ObjectProperty/@IRI*
  - Data Properties: */Ontology/DataPropertyDomain/DataProperty/@IRI*
  - Individuals: */Ontology/ClassAssertion/NamedIndividual/@**

These queries return the XML elements/attributes themselves, it is necessary to count them to obtain the number of them for each group. The following four tables (Tables 5.3, 5.4, 5.5 and 5.6) contain the number that resulted from that counting applied to the correspondent four databases.

	DBML	OWL
Tables / Classes	11	9
Link Tables / Non Classes (Obj.P.)	2	2
Relationships / Object Properties	8	12
Attributes / Data Properties	67	68
Tuples / individuals	1121	679

Table 5.3: CoursePrograms Database – Results Comparision

Please note that in the second line of the results we compare the link tables (tables that exist in the database) which are already included in the first row of the DBML column, with the non classes (that are object properties in the ontology) which are also included in the third results row of the OWL column. That same process applies to the rest of the tables where the results comparison is preformed.

	DBML	OWL
Tables / Classes	32	32
Link Tables / Non Classes (Obj.P.)	0	0
Relationships / Object Properties	18	36
Attributes / Data Properties	344	358
Tuples / individuals	25845	25845

Table 5.4: Candidates Database – Results Comparision

	DBML	OWL
Tables / Classes	6	6
Link Tables / Non Classes (Obj.P.)	0	0
Relationships / Object Properties	4	8
Attributes / Data Properties	24	26
Tuples / individuals	104885	104885

Table 5.5: Individuals Database – Results Comparision

	DBML	OWL
Tables / Classes	52	51
Link Tables / Non Classes (Obj.P.)	1	1
Relationships / Object Properties	38	74
Attributes / Data Properties	318	331
Tuples / individuals	10680	9611

Table 5.6: Intranet Database – Results Comparision

From the empirical analysis of these results it is possible to obtain the following equations:

- LinkTables (nonClasses) = DBML_tables - OWL_Classes
- DataPropertyDBML_links = OWL_Classes * 1
- ObjectProperties = (Relationships(foreignkeys) - LinkTables) * 2
- DataProperties = DBML_columns + DBML_links - Relationships
- Individuals = Tuples - (Tuples of the link tables)

The number of *DBML_links* data properties is equal to the number of classes because for each member of a class (individual) a data property assertion is made containing the reference link to the correspondent record (data) in the DBML document. This data property is added specifically to establish a link connection between the ontology (OWL) and the data (DBML).

5.3 Limitations

Binary content in tables are not covered by this work. Nevertheless, if that content appear in table a possible solution is the conversion of it into base 64 [19] and then archive the content as separated files. Each individual binary content will have a reference in the ontology that points to the specific file.

In cases where the database is not normalized, the migration to OWL may result on lack of object properties, for instance, if foreign keys are not defined. Inconsistent ontologies may also be generated if there are inconsistencies in the database relational model.

We anticipate that performance concerning the migration process with huge databases (millions of records) may be very low, taking enormous processing time. However in machines with more powerful processing capability this limitation might be mitigated.

Another limitation is the fact the the algorithm does not support link tables with foreign keys composed by more than one attribute. The algorithm current structure does not support these particular cases.

5.4 Summary

This chapter details the main contribution of our work concerning the mapping and conversion (migration) of relational databases into eXtensible Markup Language (XML) based formats (Web Ontology Language (OWL) + Database Markup Language (DBML)). Since the migration into DBML was already detailed in a previous study [Freitas, 2008], here we focus on the migration into OWL. We conceived

[19]http://en.wikipedia.org/wiki/Base64

a **F**ramework for **Re**lational **D**ata**B**ases **P**reservation (FrepDB), based on a web application with multiple interfaces, capable of handling the archival process of this family of digital objects.

To support the development stages and test of the project we used a virtual machine approach which creates an abstract layer to the hardware underneath. However, this was not an approach to directly implement a specific preservation strategy. Moreover, this approach was adopted because of the nature of the project – a prototype development – and because of the facilities considering the possibility of testing the system in different locations and with different virtual hardware configurations.

The system intends to preserve (archive) the two top levels of the digital object (experienced and conceptual object) which correspond to the database semantics and to the database relational model (data + structure). We have performed an integration of this preservation policy with the Open Archival Information System (OAIS) reference model in which the system is based (ingestion, archival, administration and dissemination). This OAIS integration means that the packages (Submission Information Package (SIP), Archival Information Package (AIP) and Dissemination Information Package (DIP)) comply with the OAIS recommendation and are composed by a combination of OWL and DBML. Metadata and Representation Information are stored in OWL and the Information Object is stored in DBML. So, the Content Information to be preserved is supported by and between these two documents (between two abstraction levels).

The mapping process is then presented, starting with the definition of the mappings and then with presentation of the multidimensional arrays and algorithms that support the conversion. The algorithms are presented in pseudo-code and after their implementation in PHP: Hypertext Preprocessor (PHP) generate the OWL code to reflect the conceptual mapping for the database semantics representation. For better interpretation of the developed work some examples of the generated code, concerning the first case study database, are presented and detailed. It is our objective to stress out the algorithms code and the correspondent OWL generated code.

To close the chapter we talk about the whole work flow concerning the preservation process with and archival and dissemination flowchart, also present the adopted methodology and detail some of the results obtained from the several tests performed over the system (using firstly the case study database and then other case studies databases). Performance tests were made and some conclusions are presented in terms of their results, however the main results analysis are dedicated to the comparison between the elements obtained from the generated ontology and the elements in the the original database (the mapped elements). Finally, at the chapters end, a brief overview is made about some of the system limitations.

Chapter 6

Conclusion and Future Work

The preservation of digital objects is undoubtedly a challenge to several organizations and companies all over the world. The fact that humans depend on machines and technologies to access and understand digital content, put us in a not very comfortable situation. Somehow we do not have the full control on how and when it is possible to access digital information. This paradigm shift that start in the last century, and that has become more dramatic in the last years, gave origin to several studies, researches and projects that intend to face this problem. National libraries, national archives or the Consultative Committee for Space Data Systems (CCSDS) were the first to embrace the challenge. One major achievement was the compilation of a reference model entitled as the Open Archival Information System (OAIS).

Concerning the nature of digital objects, different families exist, and for each group or class, different strategies or preservation policies have to be applied. During this work, an overview about the state-of-the-art was performed in order to feel and analyze what is being done concerning the work main focus: the preservation of Relational Databases (RDBs). By focusing on this specific class of digital objects, the main adopted policy was the migration of RDBs into a normalized (standard) and platform independent format. From the study about digital preservation, databases and the preservation of RDBs, we came to the hypothesis that points to a preservation policy including two abstraction levels of the digital object.

Following, we present a full summary about the whole thesis that also summarizes the research and work during the last 4 years on this PhD project. Also in this last chapter, we discuss and draw some conclusions about the developed work. At the end, we indicate lines of possible future work.

6.1 Thesis Summary

The thesis starts with the contextualization about the problematic digital preservation and the importance of also preserving a specific type of digital object which are RDBs. Every information system stands upon a place where the data is recorded and managed. This is the place that we normally call the database and its associated Database Management System (DBMS). We stress the importance of digital preservation and how it is also important to include RDBs in this problematic, because of its impact in the digital universe (namely, Information System (IS)).

We point out the importance, that RDBs have in the global context of digital objects, as our main motivation to study and research strategies to preserve them. Therefore, since the beginning, we establish our main goal as addressing the preservation of RDBs (continuing previous work [Freitas, 2008]) by focusing now on the conceptual model of the database (the IS): raise the representation level of the database up to the conceptual model and preserve that representation. For the representation of this higher level of abstraction we propose an ontology based approach.

After establishing this main goal, we start a literature review about the "state of the art" of the problematic digital preservation, in its global sense, and then focusing on the digital preservation of RDBs, specifically.

6.1.1 Preservation of Digital Objects – RDBs – Literature Review

We analyzed the main topics related to digital preservation and present a broad overview about the technics and strategies currently used and addressed by several authors concerning the preservation of digital objects. To do so we firstly characterize the digital object and its layers of abstractions along with its chain of relationships: i) the physical object ii) the logical object iii) the conceptual object and iv) the experienced object. They relate to each other by having a) hardware to interpret the physical (to logical) object; b) the software to transform a logical into a conceptual object; and c) humans that by experimenting the conceptual object produce the experienced or knowledge object. The main strategies referenced in the literature and studied under the topic of digital objects preservationwere presented and, for each one of them, their main characteristics and there scope was analyzed. At that point in the thesis we stress the fact that the strategy or strategies to be adopted depend on the digital object under preservation and on the level of abstraction that we intend to preserve.

Then, still concerning the global problematic of digital preservation we provided (chapter 2) a detailed analysis of the OAIS [Consultative Committee for Space Data Systems, 2002] reference model. This model of reference is concerned about a number of issues related to digital preservation: the process of information ingestion into the system, the information storage as well as its administration and preservation, and finally information access and dissemination. Three information packages are the base of the archival process: Submission Information Package (SIP), Archival Information Package (AIP) and Dissemination Information Package (DIP). The packages composition, the environment and behavior of a system of this nature (OAIS) are detailed and deeply analyzed in chapter 2 of the thesis.

A fundamental issue, concerning the preservation of digital objects, are their significant properties: what are the properties that should be consider important and that should be addressed by the preservation strategy. Also in chapter 2 we focused on the importance of these properties in terms of the adopted preservation policy and in terms of their relevance for an assertive representation of the preserved object. The definition of such properties can have a major impact on the success of preservation. If the selected properties are suitable to enable a correct representation of the object, i.e., if they ensure that the preserved object will maintain its original behavior or similar to that, the preservation will probably occur successfully; otherwise, we may be able to preserve the object but without an acceptable level of authenticity when compared to the original.

From these basis we then, start to analyze these concepts and strategies relating them to the specific class of digital objects under study: RDBs. They are considered a class or family of objects within the whole scope of digital artifacts. The first thing we have done, and we are already in chapter 3, was to characterize RDBs, and establish the impact that they have in the digital universe, because of the necessity of structuring data to store the organizations information (operational or other) and because of the huge spread of RDBs worldwide.

We establish a correlation between RDBs and the chain of abstraction levels present in all digital objects: physical, logical and conceptual levels. The physical object corresponds to the digital artifact stored in a physical digital media for hardware interpretation. At the logical level, the object consists on a set of bitstreams (or multi-bitstreams) coded specifically for a software interpretation. Some of these software DBMSs are referenced and we note that their environments are quite different from each other. These two bottom levels are not directly addressed in this study.

Then, we have the conceptual level of the digital object that can be materialized in the relational model with its tables, attributes and relationships. We notice that Structured Query Language (SQL), which is the standard to manipulate RDBs, also differs in some aspects between different DBMSs. It is here where the eXtensible Markup Language (XML) enters as a possible format or vehicle to archive and preserve the conceptual object (the database relational model schema and data). We identify the relational model and his specific technical details.

On the top of these three layers is the database semantics, which detains knowledge in a way that the relations and relationships have meaning and represent concepts. These concepts try to reflect reality and it is from there that the database is normalized (in conformity with the relational model). Reality needs to be somehow modeled into the relational model specifications. Considering the inverse path (reverse engineering), we have the experienced object which corresponds precisely to the database conceptual model, last abstract level (to reality).

As we noticed XML is widely adopted to be the format used in the preservation of RDBs. In the studied Software Independent Archival of Relational Databases (SIARD) solution it is suggested a set of XML and eXtensible Markup Language Schema (XMLSchema) files into where the database can be migrated. We also analyzed the Database Markup Language (DBML) format which was used in Repository of Authentic Digital Objects (RODA) and that also adopts this preservation strategy of migration the database into XML. These main approaches considering the preservation of RDBs were analyzed, detailing the XML dialects (schemas) used on both of them.

After preforming the literature review and analysis of the related work (solutions and approaches) on RDBs digital preservation we came to the conclusion that the XML format, because of its platform (software/hardware) independency, has an huge potentiality in this domain. It can easily be structured to store the data of the database and can also capture the schema (structure) of the database (relational model). However, as we highlight in the findings sub section (chapter 3), there is a significant property of the database not addressed by current approaches: the database semantics. Based on this premise, our work evolved from the initial approach and leveraged the research into an higher level of abstraction.

6.1.2 Conceptual Preservation – Rising the Perception Level

The necessity of rising the perception level of archived database emerged because, in the first approach (hypothesis), we were not considering and including the database semantics in the process.

A prototype was indeed developed and tested for the archival (with the purpose of preservation) of databases Data and Structure and an overview of the first prototype is performed in the beginning of chapter 4, stating the policy of preservation adopted then and detailing the prototype architecture and operation. DBML (an XML dialect) was the language adopted to accommodate the data of the RDB and also its structure or schema. The schema that corresponds to the relational model, i.e., the logical model RDBs. The prototype proved the feasibility of converting RDBs into an XML format (migration and normalization) and also enabled the reconstruction of RDBs from the DBML into a traditional DBMSs.

The fact that the concepts associated with data (of RDB) are not covered by preservation strategies, nor by our first approach, nor by related work approaches, leveraged the study to include a higher level of abstraction in the preservation policy. We are talking about the database semantics, information about the database concepts that could provide knowledge about the preserved data. So, a new hypothesis was established that consisted on raising the perception level of the preserved database, i.e., try to preserve an higher abstraction level of the database. That higher layer present in digital object is called the experienced object, which corresponds to the database conceptual level.

Considering this new abstraction level we introduce and analyze, in chapter 4 of the thesis, concepts as mental representations of reality. This, with the perspective of preserving or archiving, along with the logic model of the database (initial approach), these two levels of abstraction. The idea defended is the preservation of what is conceptual not only what is technical (by technical we mean the technical specifications and constrains of the relational model). The conceptual model, inherent to a certain database as a knowledge base associated and this can be exteriorized by the usage of an ontology based approach. The usage of an ontology to represent the semantics of the database is an asset in terms of knowledge preservation (conceptual preservation). The preservation life-cycle and policy for RDBs is clearly detailed in chapter 4 and consists on including in the information packages (SIP, AIP, DIP) two levels of abstractions corresponding to the conceptual and experienced digital object. At that point of the research we also established the adopted language to describe the database semantics (the database ontology) which is Web Ontology Language (OWL). Technically, the packages now include to dialects DBML and OWL.

There is a link, a close relationship between RDBs and ontologies (both represent a sort o reality [abstract or not]). Ontologies, specified with OWL, which is the adopted format, can be a vehicle to achieve knowledge representation and also a way to enable the interpretability and interoperability between possible heterogeneous systems or platforms. Ontologies differ somehow from other conceptual models since once formalized they can be directly interpretable by machines (eg.: reasoners). Thus, there are a considerable amount of related work concerning the conversion from RDBs into ontologies that we also present in chapter 4. A background subsection is also provided to overview Resource Description Framework (RDF) and OWL concerning the creation of ontologies.

We were then in position, considering the new preservation policy (now including two abstraction levels of the database), to detail and analyze the developed and implemented new prototype for the preservation of RDBs.

6.1.3 The System Architecture – FrepDB

Finally, one of the final stages of this Doctoral Program in Computer Science (before the thesis writing): the main contribution of our work, the mapping and conversion (migration) of relational databases into XML based formats (OWL + DBML). Because the migration into DBML was already detailed in a previous work [Freitas, 2008], here we focus on the migration into OWL. In chapter 5, we detail and analyze the conceived system: **F**ramework for **Re**lational **D**ata**B**ases **P**reservation (FrepDB). This framework is based on a web application with multiple interfaces, capable of handling the archival process of RDBs digital objects.

We start the system analysis by characterizing the created environment to support the project (Operating System, Services, Packages & Drivers). We used a virtual machine approach to create a abstract layer to the hardware underneath. This approach was not adopted to directly implement a specific preservation strategy. Moreover, this approach was adopted because of the nature of the project – a prototype development – and because of the facilities considering the possibility of testing the system in different locations and with different virtual hardware configurations.

The purpose of the framework (FrepDB) is to preserve (archive) the two top levels of the digital object (experienced and conceptual object) which correspond to the database semantics and to the database relational model (data + structure). We have performed an integration of this preservation policy with the OAIS reference model in which the system is based (ingestion, archival, administration and dissemination). This OAIS integration means that the packages (SIP, AIP and

DIP) comply with the OAIS recommendation and are composed by a combination of OWL and DBML. The OWL provides the knowledge base ("Representation of Information"), recommended by the OAIS reference model, to understand the Information Object (the relational database). By using the ontology approach to capture the database semantics, the OWL generated code tries to reflect the concepts, properties and relations that exist in the database at an higher level of abstraction, thus providing a knowledge base to understand the RDB data (information). A Representation of Information (metadata) is stored in OWL and the Information Object is stored in DBML. So, the Content Information to be preserved is supported by and between these two documents (between two abstraction levels).

The mapping process is then presented in chapter 5, starting with the definition of the mappings (Table 5.1) and with the presentation of the multidimensional arrays and algorithms that support the conversion. The algorithms are presented in pseudo-code and are the core of the developed work.

The implementation was done in PHP: Hypertext Preprocessor (PHP), which enabled the generation of OWL code that reflects the conceptual mapping for the database semantics representation. For better interpretation of the developed work some examples of the generated code, concerning the first case study database, are presented and detailed. It was our objective to stress out the algorithms code and the correspondent OWL generated code.

At the end we talk about the whole work flow concerning the preservation process with and archival and dissemination flowchart (Fig. 5.6), and also present the adopted methodology and detail some of the results obtained from the several tests performed over the system (using firstly the case study database and then other case studies). The tests phase was important to refine the algorithms and to achieve some conclusions in terms of results. The main results analysis were dedicated to the comparison between the elements obtained from the generated ontology and the elements in the the original database (the mapped elements). The system limitations are also analyzed before the end of chapter 5.

The summary of the developed work is present in this final chapter, and the thesis main conclusions and future work are detailed in the following sections.

6.2 Conclusions

Arriving to this point, it is important to underline the fact that the work to be done to determine accurate methods or strategies for preservation of digital objects is undoubtedly closer. It was the new paradigm where more and more data (information) is stored in digital artifacts that boosted researches, studies, or even empirical achievements in order to establish methods and approaches to keep information available for the later accesses or (long-term scope) for future generations.

An important fact is that different types (formats) of digital objects normally require different preservation strategies. In our research we explore possible preservation strategies of an well known "hard format": RDBs.

It is unquestionable that the database data is the main subject of preservation, however to understand the data and how it is related, the structure of the database also needs to be preserved. But, this might not be enough. What about the the meaning of the database elements? That, the database semantics, is also a "significant property" present in these digital objects.

So we establish as one of the main objectives of this work the conversion/migration of RDBs into ontologies (OWL), based on a mapping process (mapping algorithm), with the purpose of its preservation. The migration of the Data and Structure to DBML was covered in our initial approach. Therefore, here we give special emphasis to the conversion between RDBs and OWL. Globally, the main contribution of our work consisted on the mapping and conversion (migration) of relational databases into XML based formats (OWL + DBML).

The main objective of establishing and implementing a new preservation policy, through the FrepDB system, was achieved therefore proving that the proposed hypothesis is feasible. Nevertheless, there are some limitations, namely the problem of scalability, database dimensions can be a problem to our approach. Both DBML and OWL documents are composed by an unique file each. This means that for very large databases the system needs to generate huge XML files, something that implicates huge processing capabilities and therefore the consumption of a lot of hardware resources. A possible solution, as we will point in the future work section, may be to apply the scalability strategies adopted by the SIARD solution.

The archival of two representations of the database enables a more complete overview about the whole database, something that current approaches do not support. The ontology representation of the database is an advantage to current known approaches concerning RDBs preservation, since it facilitates its interpretation and access. We created a new form of accessing the database through the navigation in the mapped ontology (OWL).The ontology approach also relegates to a second plan the technical transformation necessary and associated to the relational model, thus providing a simpler representation model of the database. With these new preservation policy it is possible to abstract the database relational model, i.e., the database normalization.

In one hand, we have the advantage that the ontology brings in terms of data interpretation, and on the other hand the benefit (if desired) of not dealing with the technical transformations associated to the database relational model. Furthermore, the ontology approach, incorporated in the preservation policy, also enables machine interpretation of the database (formalized) knowledge.

Without the presumption of having found "the ultimate solution", our proposal and tested hypothesis, embraces the archival of the artifact under preservation (RDBs) in a neutral format (DBML) attached with an ontology representation of the object semantics (OWL), namely its concepts and properties. This combined strategy is indeed an asset to future interpretations of structured data (RDBs or others). We also anticipate that ontologies may be adopted by others and play an important role in digital preservation (not only in the preservation of RDBs)

6.3 Future Work

In terms os future work, we identify two main paths that can be explored: a) the framework improvement and b) study and research how can ontologies have more impact in the preservation of structured data.

Starting with the framework improvement, there are some issues related to the system limitations that can be further developed. For instance, the support of binary content that might exist in the RDB or the improvement of the mapping algorithms to support more complex relational models.

Another important improvement, or upgrade, might be the adoption of the SIARD format to reflect the database relational model (instead of the DBML). This possible change is justified by the fact that the SIARD format is more scalable than the DBML format. Moreover, it should also be considered the possibility of adopting the same principle (division of the XML generated code into several files) to the OWL document, thus making the system more scalable.

We also anticipate the possibility of integration between OWL and Semantic Web Rule Language (SWRL) [Horrocks et al., 2004] to consolidate the asserted and inferred knowledge about the database and its information system. The integration with SWRL offers the possibility of imposing more restrictions to the ontology model through their semantic rules.

Future work can also consist on studying the possible extension of the framework to support the archival of other types of structured data, like data sets for instance. Here, we are already making the bridge to the second main path, pointed in the beginning of this section, as future work. The relational model possesses a formal specification, which we study and present a possible mapping into ontologies. Nevertheless, RDBs are not the only means to store, manage and administrate structured data. We already mention the data sets (collection of records) which are used in several fields, namely in scientific researches, but there are several others, such as the dimensional model used in dataware systems. Our main perspective is that every structured information might be somehow mapped into a specific ontology or ontologies.

Other possible fields of research are related to potentialities that ontologies possess in terms of interoperability,, when formalized they can be interpreted by machines. Another important area that can be deeply explored is related to metadata, which we also intend to also address in future work. Ontologies can provide answers to questions that other metadata standards are unable to give (they normally possess rigid structures).

The number of fields in which ontologies may impact is very diversified, opening in this way a panoply of possible future researches related to digital preservation but not limited to it.

The study surrounding the preservation of RDBs is not closed, however with this work, we hope to have contributed positively towards its solution and provided a better understanding of the problem.

Appendix A

XMLSchemas Details & DTDs

Code block 18 shows the DTD corresponding to the STRUCTURE element definition in the DBML document.

Code Block 18 : DTD for the STRUCTURE element

```
<! ELEMENT STRUCTURE (TABLE +)>
<! ELEMENT TABLE (COLUMNS, KEYS)>
<! ATTLIST TABLE
NAME CDATA # REQUIRED>
<! ELEMENT COLUMNS (COLUMN +)>
<! COLUMN ELEMENT EMPTY>
<! ATTLIST COLUMN
NAME CDATA # REQUIRED
TYPE CDATA # REQUIRED
SIZE CDATA # REQUIRED
NULL CDATA # REQUIRED>
<! ELEMENT FIELD EMPTY>
<! ATTLIST FIELD
NAME CDATA # REQUIRED>
<! ELEMENT KEYS (PKEY, FKEY *)>
<! ELEMENT PKEY (FIELD +)>
<! ATTLIST PKEY
TYPE CDATA # REQUIRED>
<! FKEY ELEMENT EMPTY>
<! ATTLIST FKEY
NAME CDATA # REQUIRED
IN CDATA # REQUIRED
REF CDATA # REQUIRED>
```

In code block 19 we can find a portion of a DTD of the DATA element in a DBML document.

Code Block 19 : DTD Fragment for the Element DATA

```
<!ELEMENT DATA ((Authors, AuthorsBibliography, Bibliography,
 BibliographyType, Courses, dtproperties, Languages,
 ProgramBibliography, ProgramContent, Programs, sysdiagrams))>
<!ELEMENT Courses ((Courses-REG+))>
<!ELEMENT Courses-REG ((CodDisciplina, DesDisciplina, AbrevDisciplina,
 DesDisciplina_ING, DesDisciplinaOficial, SubstituirNomeQuandoEquivalenc,
 DesDisciplina_POR_QuandoPreenc, DesDisciplina_ING_QuandoPreenc,
 SiglaAreaCientifica, CodAreaCientifica))>
<!ELEMENT Coordenador (#PCDATA)>
<!ELEMENT Competencias (#PCDATA)>
<!ELEMENT CodPlano (#PCDATA)>
<!ELEMENT CodDocente (#PCDATA)>
<!ELEMENT CodDisciplina (#PCDATA)>
<!ELEMENT CodAreaCientifica (#PCDATA)>
<!ELEMENT Ciclo (#PCDATA)>
<!ELEMENT BibliographyType-REG ((IDTipo, DesTipo))>
<!ELEMENT BibliographyType ((BibliographyType-REG+))>
<!ELEMENT Bibliography-REG ((IDLivro, Tipo, Titulo, Edicao, LLocal,
 Editora, Ano, ISSN, ISBN))>
<!ELEMENT Bibliography ((Bibliography-REG+))>
<!ELEMENT Avaliacao (#PCDATA)>
<!ELEMENT AuthorsBibliography-REG ((IDAutor, IDLivro))>
<!ELEMENT AuthorsBibliography ((AuthorsBibliography-REG+))>
<!ELEMENT Authors-REG ((IDAutor, Nome, Obs))>
<!ELEMENT Authors ((Authors-REG+))>
<!ELEMENT Assistente (#PCDATA)>
<!ELEMENT AnoLectivo (#PCDATA)>
<!ELEMENT AnoCurricular (#PCDATA)>
<!ELEMENT Ano (#PCDATA)>
<!ELEMENT AbrevDisciplina (#PCDATA)>
```

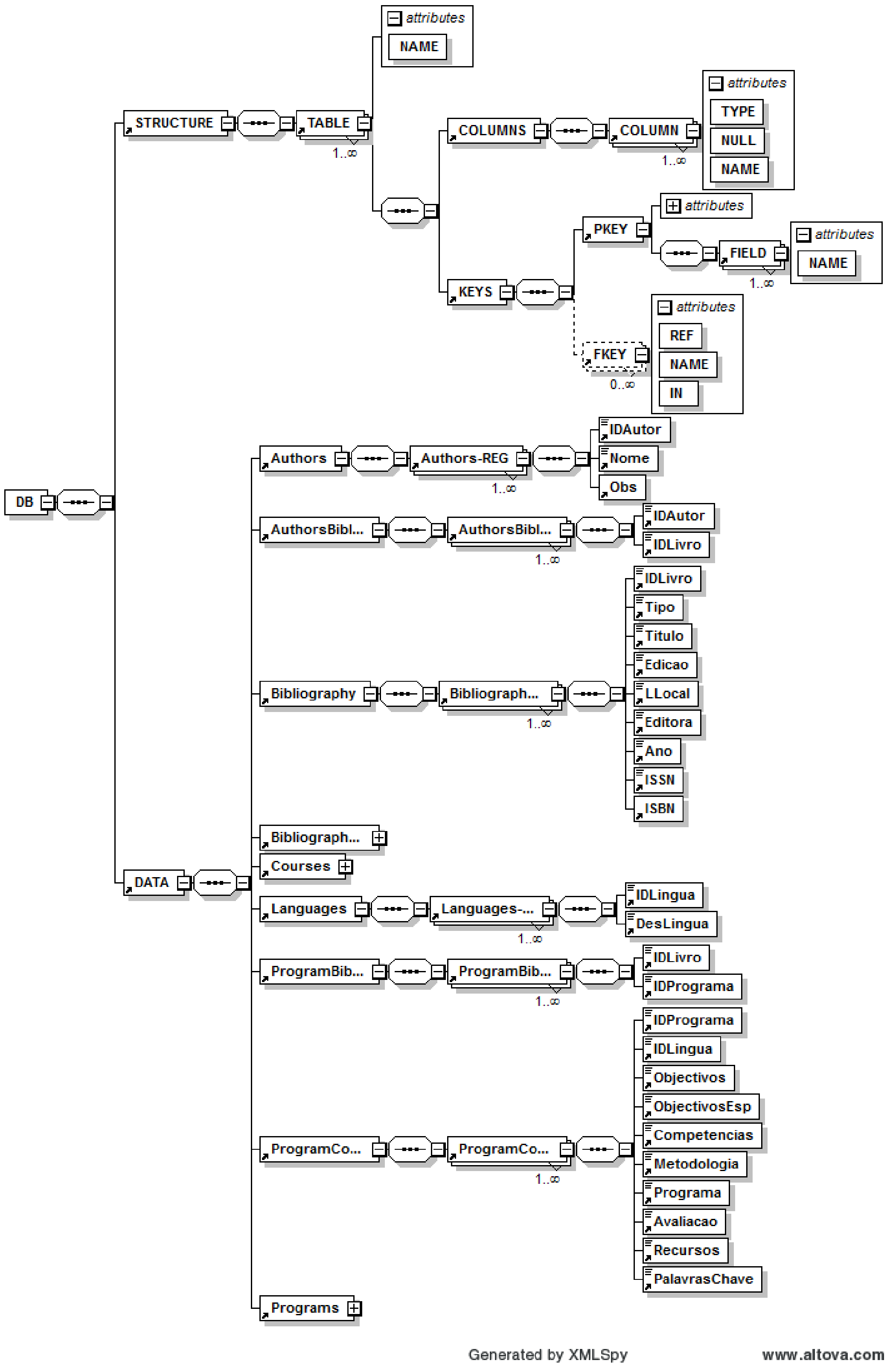
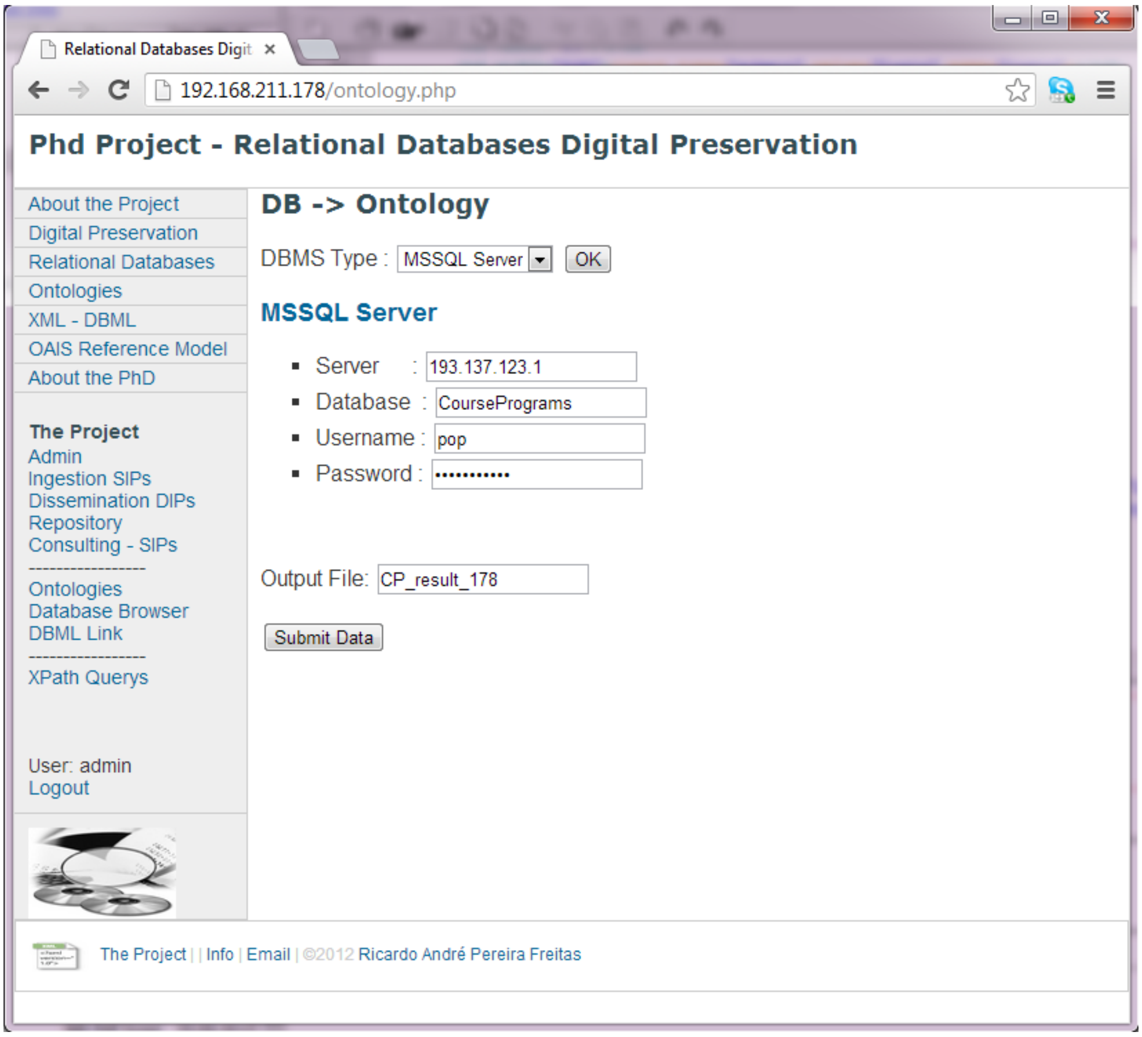


Figure A.1: A global overview of the DBML document structure (XMLSchema)

Appendix B

FrepDB – Web Interface



Phd Project - Relational Databases Digital Preservation

User: admin
Logout

DB -> Ontology

DBMS Type : MSSQL Server OK

MSSQL Server

- Server : 193.137.123.1
- Database : CoursePrograms
- Username : pop
- Password : •••••••••••

Output File: CP_result_178

Submit Data

The Project | | Info | Email | ©2012 Ricardo André Pereira Freitas

Figure B.1: FrepDB – Web Interface

Appendix C

Conferences and Publications by the Author During the PhD

2012

- Ricardo André Pereira Freitas, José Carlos Ramalho; New Dimension in Relational Database Preservation: Using Ontologies. In *Innovations in XML Applications and Metadata Management: Advancing Technologies*, chapter 9, IGI-Global, 2012.

2011

- Ricardo André Pereira Freitas, José Carlos Ramalho; "*Significant Properties in the Preservation of Relational Databases*"; IEEE-TCDL. Bulletin of IEEE Technical Committee on Digital Libraries; Volume 7 Isseu 1. Sep. 2011; URI no RepositoriUM: http://hdl.handle.net/1822/16188
- Ricardo André Pereira Freitas, José Carlos Ramalho; "*New Dimension in Relational Database Preservation: rising the abstraction level*"; iPRES 2011 - 8th International Conference on Preservation of Digital Objects; Singapore; 1 - 4 Nov; 2011; URI no RepositoriUM: http://hdl.handle.net/1822/15859
- Ricardo André Pereira Freitas, José Carlos Ramalho; " *Using Ontologies to Abstract Relational Databases Conceptual Model*"; EPIA2011 - 15th Portuguese

Conference on Artificial Intelligence; Lisboa, Portugal; 10 - 13 Oct; 2011; URI no RepositoriUM: http://hdl.handle.net/1822/16113

- Ricardo André Pereira Freitas, José Carlos Ramalho; "*Relational Databases Conceptual Preservation*"; VLDL2011 - Fourth Workshop on Very Large Digital Libraries (In conjunction with the International Conference on Theory and Practice of Digital Libraries (TPDL2011)); Berlin, Germany; 09.29; 2011; URI no RepositoriUM: http://hdl.handle.net/1822/16019

- Ricardo André Pereira Freitas, José Carlos Ramalho; "*Preservation of Relational Databases*"; MSKE2011- International Conference on Managing Services in the Knowledge Economy; Universidade Lusíada, Vila Nova de Famalicão, Portugal; 07.13; 2011; URI no RepositoriUM: http://hdl.handle.net/1822/13704

- Ricardo André Pereira Freitas, José Carlos Ramalho; "*Using Ontologies in Database Preservation*"; 9th Conference on XML: Aplicaes e Tecnologias Associadas, XATA 2011, ESEIG -IPP, Vila do Conde - Portugal, 2011; URI no RepositoriUM: http://hdl.handle.net/1822/13703;

2010

- Ricardo André Pereira Freitas, José Carlos Ramalho; "*Significant Properties in the Preservation of Relational Databases*"; Research and Advanced Technology for Digital Libraries, 14th European Conference, ECDL2010, Glasgow, UK, September 6-10, 2010; isbn: 978-3-642-15463-8; URI no RepositoriUM: http://hdl.handle.net/1822/13702

2009

- Ricardo André Pereira Freitas, José Carlos Ramalho; "*Relational databases digital preservation*"; INFORUM : Simpósio de Informática, Lisboa – Portugal, 2009; isbn:978-972-9348-18-1; URI no RepositoriUM: http://hdl.handle.net/1822/9740

Bibliography

Pat Manson. Digital preservation research: An evolving landscape. 2010. URL `http://ercim-news.ercim.eu/en80/keynote`.

Kyong-Ho Lee, Oliver Slattery, Richang Lu, Xiao Tang, and Victor Mccrary. The State of the Art and Practice in Digital Preservation. *Journal of Research of the National Institute of Standards and Technology*, 107(1):93–106, January 2002.

National Library of Australia. Guidelines for the preservation of digital heritage. Technical report, United Nations Educational, Scientific and Cultural Organisation (UNESCO) (Prepared by National Library of Australia), March 2002. URL `http://unesdoc.unesco.org/images/0013/001300/130071e.pdf`.

K. Thibodeau. Overview of technological approaches to digital preservation and challenges in coming years. *The state of digital preservation: an international perspective*, 2002.

Miguel Ferreira. *Introdução à preservacao digital – Conceitos, estratégias e actuais consensos.* Escola de Engenharia da Universidade do Minho, Guimarães, Portugal, 2006.

Ricardo André Pereira Freitas. *Preservação Digital de Bases de Dados Relacionais.* MSc Thesis, Escola de Engenharia, Universidade do Minho, Portugal, 2008.

Edgar Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 1970.

Ricardo André Pereira Freitas and José Carlos Ramalho. Relational databases digital preservation. In *Inforum: Simpósio de Informática*, pages 491–494, Lisboa, Portugal, 2009. ISBN 978-972-9348-18-1.

M.H. Jacinto, G.R. Librelotto, J.C. Ramalho, and P.R. Henriques. Bidirectional conversion between xml documents and relational databases. In *Computer Supported Cooperative Work in Design, 2002. The 7th International Conference on*, pages 437 – 443, 2002. doi: 10.1109/CSCWD.2002.1047728.

Consultative Committee for Space Data Systems. Reference model for an open archival information system (OAIS) : Recommendation for space data system standards : CCSDS 650.0-B-1. Technical report, January 2002. URL `http://public.ccsds.org/publications/archive/650x0b1.pdf`.

SIARD. SIARD – Format Description. Technical report, Swiss Federal Archives - SFA, Berne, September 2008.

Deborah L. Mcguinness and Frank van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, W3C, February 2004.

Gail M Hodge. Best Practices for Digital Archiving: An Information Life Cycle Approach. *DLib Magazine Volume*, 6(1):1–13, 2000. ISSN 10829873. URL `http://www.dlib.org/dlib/january00/01hodge.html`.

José Carlos Ramalho, Miguel Ferreira, Luís Faria, Rui Castro, Francisco Barbedo, and Luís Corujo. RODA and CRiB a service-oriented digital repository. In *International Conference on Preservation of Digital Objects - iPRESS 2008*, London, 2008.

Adam Farquhar and Helen Hockx-Yu. Planets: integrated services for digital preservation. *The International Journal of Digital Curation*, 2, 2007. URL `http://www.ijdc.net/index.php/ijdc/article/view/45/31`.

Ross King, Rainer Schmidt, Christoph Becker, and Sven Schlarb. Scape: Big data meets digital preservation. *ERCIM News*, 2012(89), 2012.

ExLibris. The ExLibris Group. Technical report, ExLibris, 2012. URL `http://www.exlibrisgroup.com/`. Accessed: 01/06/2012.

DANS. Data Archiving and Networked Services (DANS) Institute, 2012. URL `http://www.dans.knaw.nl/`. Accessed: 01/06/2012.

DELOS. Network of Excellence on Digital Libraries, 2009. URL http://delos.info/. Accessed: 04/06/2012.

Christoph Becker, Miguel Ferreira, Michael Kraxner, Andreas Rauber, Ana Alice Baptista, and Jos Carlos Ramalho. Distributed preservation services: Integrating planning and actions. In *Research and advanced technology for digital libraries : proceedings of the 12th European Conference on Digital Libraries (ECDL'08)*, pages 25–36. Heidelberg : Springer-Verlag, 2008. ISBN 978-3-540-87598-7. URL http://doi.acm.org/10.1145/1526709.1526793.

Howard Besser. Digital Preservation of Moving Image Material. 2001.

Phil Mellor, Paul Wheatley, and Derek M. Sergeant. Migration on request, a practical technique for preservation. In *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '02, pages 516–526, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44178-6. URL http://dl.acm.org/citation.cfm?id=646635.700203.

Jane Hunter and Sharmin Choudhury. A semi-automated digital preservation system based on semantic web services. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '04, pages 269–278, New York, NY, USA, 2004. ACM. ISBN 1-58113-832-6. doi: 10.1145/996350.996415. URL http://doi.acm.org/10.1145/996350.996415.

J. Ramalho, M. Ferreira, L. Faria, and R. Castro. Relational database preservation through xml modelling. In *Extreme Markup Languages*, Montréal, Québec, 2007.

Miguel Ferreira. *Preservação de Longa Duração de Informação Digital no Contexto de um Arquivo Histrico.* PhD thesis, Escola de Engenharia, Universidade do Minho, Portugal, Julho 2009.

Alan R. Heminger and Don M. Kelley. A delphi assessment of the digital rosetta stone model. *Hawaii International Conference on System Sciences*, 4:40104b, 2004. ISSN 1530-1605. doi: http://doi.ieeecomputersociety.org/10.1109/HICSS.2004.1265277.

Reagan Moore, Chaitan Baru, Arcot Rajasekar, Bertram Ludaescher, Richard Marciano, Michael Wan, Wayne Schroeder, and Amarnath Gupta. Collection-Based persistent digital archives - part 1. *D-Lib Magazine*, 6(3), March 2000. ISSN 1082-9873. URL http://www.dlib.org/dlib/march00/moore/03moore-pt1.html.

Claire Eager. The state of preservation metadata practices in north carolina repositories. Technical report, Chapel Hill, North Carolina, 2003.

D. Waters. Good archives make good scholars: Reflections on recent steps toward the archiving of digital information. Number Washington, D.C. in Documentation Abstracts, Inc., pages 78–95. Council on Library and Information Resources, February 2002. ISBN 1-887334-92-0. URL http://mixed.dans.knaw.nl/files/file/The%20State%20of%20Digital%20Preservation.pdf#page=84.

Brian F. Lavoie. The Open Archival Information System Reference Model: Introductory Guide. Technical report, Digital Preservation Coalition/OCLC, January 2004. URL http://www.dpconline.org/docs/lavoie_OAIS.pdf.

A. Wilson. Significant properties report. Technical report, InSPECT Project, April 2007. URL http://www.significantproperties.org.uk/wp22_significant_properties.pdf.

Angela Dappert and Adam Farquhar. Significance is in the eye of the stakeholder. In *Proceedings of the 13th European conference on Research and advanced technology for digital libraries*, ECDL'09, pages 297–308, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 3-642-04345-3, 978-3-642-04345-1. URL http://dl.acm.org/citation.cfm?id=1812799.1812838.

PLANETS. Planets – preservation and long-term access through networked services. Technical report, 2010. URL http://www.planets-project.eu/.

Tim Bray, Jean Paoli, C. M. Sperberg-Mcqueen, Eve, and François Yergeau, editors. *Extensible Markup Language (XML) 1.0.* W3C Recommendation. W3C, fourth edition, August 2003. URL http://www.w3.org/TR/REC-xml/.

P. Henriques J. Ramalho. *XML and XSL - Da Teoria à Prática*. FCA - Editora Informática, 2002.

Dirk Roorda Ren van Horik. Migration to intermediate xml for electronic data (mixed): Repository of durable file format conversions. *The International Journal of Digital Curation*, Issue 2, Volume 6, Last updated September 2011.

Bram Van Der Werf. Open Source Software and Digital Preservation: An Interview with Bram van der Werf of the Open Planets Foundation, 2012. URL `http://blogs.loc.gov/digitalpreservation/2012/04/open-source-software-and-digital-preservation-an-interview-with-bram-van-der-werf-of-the-open-planets-foundation/`.

Andrew Eisenberg and Jim Melton. Sql: 1999, formerly known as sql3. *SIGMOD Rec.*, 28(1):131–138, March 1999. ISSN 0163-5808. doi: 10.1145/309844.310075. URL `http://doi.acm.org/10.1145/309844.310075`.

Donald D. Chamberlin and Raymond F. Boyce. Sequel: A structured english query language. In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, SIGFIDET '74, pages 249–264, New York, NY, USA, 1974. ACM. doi: 10.1145/800296.811515. URL `http://doi.acm.org/10.1145/800296.811515`.

D. Chamberlin, M. Astrahan, M. Blasgen, J. Gray, W. King, B. Lindsay, R. Lorie, J. Mehi, T. Price, F. Putzolu, P. Selinger, M. Schkolnick, D. Slutz, I. Traiger, B. Wade, and R. Yost. A History and Evaluation of System R. Technical report, IBM Research Laboratory, San Jose, California, 1981.

Jim Melton and Alan R. Simon. *SQL: 1999 - Understanding Relational Language Components*. Morgan Kaufmann, May 2001. ISBN 1558604561.

Lee Buck. Data models as an xml schema development method. In *XML 99*, Phyladelphia, december 1999.

Ronald Bourret. Xml and databases. *Copyright 1999-2005 by Ronald Bourret*, Last updated September 2005.

W3C. World wide web consortium. Technical report, 2012. URL http://www.w3.org/.

John Cowan. extensible markup language a basic overview. 1998.

XPath W3C Recommendation. XML Path Language (XPath), 1999. URL http://www.w3.org/TR/xpath/.

B. Lavoie and R. Gartner. Preservation Metadata. Technical report, Digital Preservation Coalition/OCLC, Oxford, September 2005.

Premis. PREMIS Data Dictionary for Preservation Metadata, version 2.0. Technical report, PREMIS Editorial Committee, March 2008. URL http://www.loc.gov/standards/premis/v2/premis-2-0.pdf.

OCLC/RLG Preservation Metadata Working Group. A Metadata Framework to Support the Preservation of Digital Objects. Technical report, OCLC Online Computer Library Center, Dublin, USA, mar 2002. URL http://www.loc.gov/standards/premis/v2/premis-2-0.pdf.

Michael Day. The oais reference model. Technical report, Digital Curation Centre UKOLN, University of Bath, 2006.

David Pitt. Mental representation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

Ludwig Wittgenstein. *Tractatus Logico-Philosophicus*. K. Paul, Trench, Trubner & Co. Ltd. ; Harcourt, Brace and Co., London; New York, 1922.

Robin Allott. Language as a mirro of the world: Reconciling picture theory and language games, July 2003. URL http://cogprints.org/3110/. presented at 30th LACUS, Victoria University, Canada.

N. Guarino. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998. ISBN 9051993994.

M. Mora, O. Gelman, F. Cervantes, and G. Forgionne. *A Formal Definition of Information Systems.* Encyclopedia of Information Science and Technology. In M. Khosrow-Pour (Ed.), second edition edition, 2009.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 2001.

Thomas R. Gruber. Ontology. In *Encyclopedia of Database Systems*, pages 1963–1965. 2009.

Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993. ISSN 1042-8143. doi: 10.1006/knac.1993.1008. URL `http://dx.doi.org/10.1006/knac.1993.1008`.

Semantic Web. Semantic web. Technical report, 2012. URL `http://www.w3.org/standards/semanticweb/`.

Heru Agus Santoso, Su-Cheng Haw, and Ziyad T Abdul-Mehdi. Ontology extraction from relational database: Concept hierarchy as background knowledge. *Knowledge-Based Systems*, 24(3):457–464, 2011. ISSN 09507051. doi: 10.1016/j.knosys.2010.11.003. URL `http://linkinghub.elsevier.com/retrieve/pii/S0950705110001693`.

Frank Manola and Eric Miller, editors. *RDF Primer.* W3C Recommendation. World Wide Web Consortium, February 2004. URL `http://www.w3.org/TR/rdf-primer/`.

Eric Miller. An Introduction to the Resource Description Framework. *D-Lib Magazine*, May 1998. URL `http://www.dlib.org/dlib/may98/miller/05miller.html`.

Gian Piero Zarri. *Encyclopedia of Knowledge Management.* IGI Global, September 2005. ISBN 9781591405733. doi: 10.4018/978-1-59140-573-3. URL `http://www.igi-global.com/chapter/encyclopedia-knowledge-management/17026/`.

Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax, 1998.

Dan Brickley and R. V. Guha, editors. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. World Wide Web Consortium, February 2004. URL `http://www.w3.org/TR/rdf-schema/`.

RDF. Resource description framework. Technical report, 2004. URL `http://www.w3.org/RDF/`.

Several. Automated Generation of RDF Views over Relational Data Sources with Virtuoso, 2009. URL `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSSQL2RDF`.

C. Bizer and R. Cyganiak. D2rq – lessons learned. In *W3C Workshop on RDF Access to Relational Databases*, Cambridge, USA, 2007.

Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: light-weight linked data publication from relational databases. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 621–630, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526793. URL `http://doi.acm.org/10.1145/1526709.1526793`.

Farid Cerbah. Learning highly structured semantic repositories from relational databases: The rdbtoonto tool. In *In Proc. of ESWC 2008*, 2008.

Jesús Barrasa Rodriguez and Asunción Gómez-Pérez. Upgrading relational legacy data to the semantic web. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 1069–1070, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: 10.1145/1135777.1136019. URL `http://doi.acm.org/10.1145/1135777.1136019`.

H. Chen and Z Wu. Dartgrid iii: A semantic grid toolkit for data integration. In *Proceedings of the First International Conference on Semantics, Knowledge, and Grid*, 2005.

Matthew Fisher and Mike Dean. Automapper: Relational Database Semantic Translation using OWL and SWRL. *Architecture*, pages 1–6, 2007. URL http://www.progeny.net/People/MattFisher/files/papers/AutomapperIASKEALT07.pdf.

Jiuyun Xu and Weichong Li. Using relational database to build owl ontology from xml data sources. In *Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops*, CISW '07, pages 124–127, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3073-7. doi: 10.1109/CIS.Workshops.2007.139. URL http://dx.doi.org/10.1109/CIS.Workshops.2007.139.

Guntars Būmans and Kārlis Čerāns. Rdb2owl: a practical approach for transforming rdb data into rdf/owl. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 25:1–25:3, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0014-8. doi: 10.1145/1839707.1839739. URL http://doi.acm.org/10.1145/1839707.1839739.

N. Cullot, R. Ghawi, and K. Yetongnon. Db2owl: A tool for automatic database-to-ontology mapping. In *Proceedings of 15th Italian Symposium on Advanced Database Systems (SEBD 2007)*, pages 491–494, Torre Canne, Italy, June 2007.

W3C. R2RML: RDB to RDF mapping language. Technical report, 2011. URL http://www.w3.org/TR/r2rml/.

Justas Trinkunas and Olegas Vasilecas. Building ontologies from relational databases using reverse engineering methods. In *Proceedings of the 2007 international conference on Computer systems and technologies*, CompSysTech '07, pages 13:1–13:6, New York, NY, USA, 2007. ACM. ISBN 978-954-9641-50-9. doi: 10.1145/1330598.1330614. URL http://doi.acm.org/10.1145/1330598.1330614.

Khalid M. Albarrak and Edgar H. Sibley. Translating relational & object-relational database models into owl models. In *Proceedings of the 10th IEEE international conference on Information Reuse & Integration*, IRI'09, pages 336–341, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-4114-3. URL `http://dl.acm.org/citation.cfm?id=1689250.1689311`.

Chen He-ping, He Lu, and Chen Bin. Research and implementation of ontology automatic construction based on relational database. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 05*, CSSE '08, pages 1078–1081, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3336-0. doi: 10.1109/CSSE.2008.1427. URL `http://dx.doi.org/10.1109/CSSE.2008.1427`.

Satya S Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr., Sören Auer, Juan Sequeda, and Ahmed Ezzat. A Survey of Current Approaches for Mapping of Relational Databases to RDF. *w3org*, 2009. URL `http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf`.

I. Myroshnichenko and M.C. Murphy. Mapping er schemas to owl ontologies. In *Semantic Computing, 2009. ICSC '09. IEEE International Conference on*, pages 324 –329, sept. 2009. doi: 10.1109/ICSC.2009.61.

Protégé Project. The Protégé Ontology Editor and Knowledge Acquisition System. Technical report, 2012. URL `http://protege.stanford.edu`.

I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Technical report, W3C Member submission, 2004.