Relational Database Preservation through XML modelling

José Carlos Ramalho [University of Minho, Department of Informatics]
Miguel Ferreira [University of Minho, Department of Information Systems]
Luís Faria [Portuguese National Archives]
Rui Castro [Portuguese National Archives]

Extreme Markup Languages®

Introduction

Archives are complex structures composed of human resources, policies and information. Due to the social responsibility that is usually associated with this type of institutions - that being, safekeeping the intellectual heritage of the human kind - it is one of their quintessential commitments to hold, preserve and make accessible to their users the information that is kept under their custody and care.

In a time and age where most informational items are produced in some type of digital form, it is important for these institutions to acquire new capabilities in what concerns managing, preserving and providing access to this information. This means that Archives are expected to be able to treat digital information with the same interest and respect as they currently do for historical records in analog formats such as paper or microfilm.

Preserving the bits

One of the indisputable truths about digital information is that it can be copied an infinite number of times without loosing any data. However, access to digital information is only possible if the appropriate technological environment is in place in order to render those bits of information into something that potential consumers of that information will be able to understand. The major problem with the long-term preservation of digital information is not keeping the data safe and stable (i.e. making sure that the bits of information do not get involuntarily altered), but providing access to that data in ways that are intelligible to its users, which can be human beings, software or any other type of contraption. In a context where hardware and software are evolving ever so rapidly (and not always in a well behaved fashion in what concerns retro-compatibility) access to this information becomes a major issue.

The set of processes that are responsible for making sure that information remains accessible and intelligible for long periods of time is generally called "digital preservation". Digital preservation, as a new branch of information science, is concerned with making sure that digital objects remain readable and interpretable for much longer periods of time than the expected lifetime of the individual hardware and software components that comprise the technological environment that is required to render those objects, as well as the formats in which the items of information are encoded **[RRLRM05][LRW03]**.

Many techniques have been introduced which aim at solving the problem of digital preservation. Among these are emulation **[Rot99]** **[Gra00]**, encapsulation **[SM98]** **[SM99]** and migration **[DPT01]** **[Whe01]**, as well as an assortment of variations and combinations from all of the above, e. g. normalization **[Thi02]** **[LSLTM02]**, migration on-request **[MWS02]** **[Rus03]** or Universal Virtual Computer (UVC) **[Lor01]** **[Lor02]**.

The migration strategy can best described as a "(...) set of organized tasks designed to achieve the periodic transfer of digital materials from one hardware/software configuration to another or from one generation of computer technology to a subsequent generation." **[TFADI96]**.

Contrary to other preservation strategies, migration-based approaches do not attempt to preserve digital objects in their original forms. In alternative, they transform objects from near obsolete formats to more up-to-date encodings that most users will be able to interpret using common software available on their personal computers. This can be regarded as a format conversion during access or dissemination, i.e. the stored items of information (generally referred to as digital objects in a digital preservation context) are transformed from their archival formats to formats more suitable for consumption by its users.

Normalization, on the other hand, consists in reducing the number of formats in the archive by converting digital objects from their original formats to a small number of normalized formats. This is usually done during the ingest stage, i.e. the act of bringing new materials into the archival environment. In this type of strategy, objects within a certain class (e.g. text documents, digital images, relational databases, etc.) are converted to a preservation format that is considered suitable for long-term preservation. The preservation format is usually a well documented and easy do decode format that will ensure readability for long periods of time. This strategy could be regarded as conversion during ingest.

RODA (Repository of Authentic Digital Objects)

The National Archives of Portugal (Direcção Geral de Arquivos) is responsible for the safekeeping of a large part of the information that is

produced by portuguese public administration institutions. However, the National Archives do not yet detain the necessary infrastructures to support the processes of managing and preserving the digital information produced by these institutions.

The current initiatives of the portuguese e-government establish the need to support public administration activities in information technologies to improve efficiency, productivity and quality of their services to the general public. In this scenario it is clear that the number of digital objects produced by these institutions will grow and that their legal value, as well as authenticity, should not be compromised. These objects will serve as the testimony of the activities carried out by public organizations and will constitute the social and patrimonial memory of our country.

Following this line of thought, the Portuguese National Archives are undertaking the endeavour of developing processes, tools and resources capable of answering the needs of the public administration in terms of preserving the digital material currently being produced.

The goal of the RODA project is to provide technical solutions to digital preservation at the national level. The first stage of this project consisted in building a prototype of a repository system capable of preserving digital information and making sure that this information remains authentic within the archival environment. This prototype will serve as a the basis for development of a fully functional solution capable of ingesting and managing large quantities and genre of digital objects.

In order to prove the concept of a repository system capable of preserving digital information, we limited the number of accepted types of digital objects to three: text documents, still images and relational databases.

Although the focus of this paper is on the database preservation component we will begin with a glimpse of the overall RODA's architecture and present the data model that supports it.

RODA's general architecture

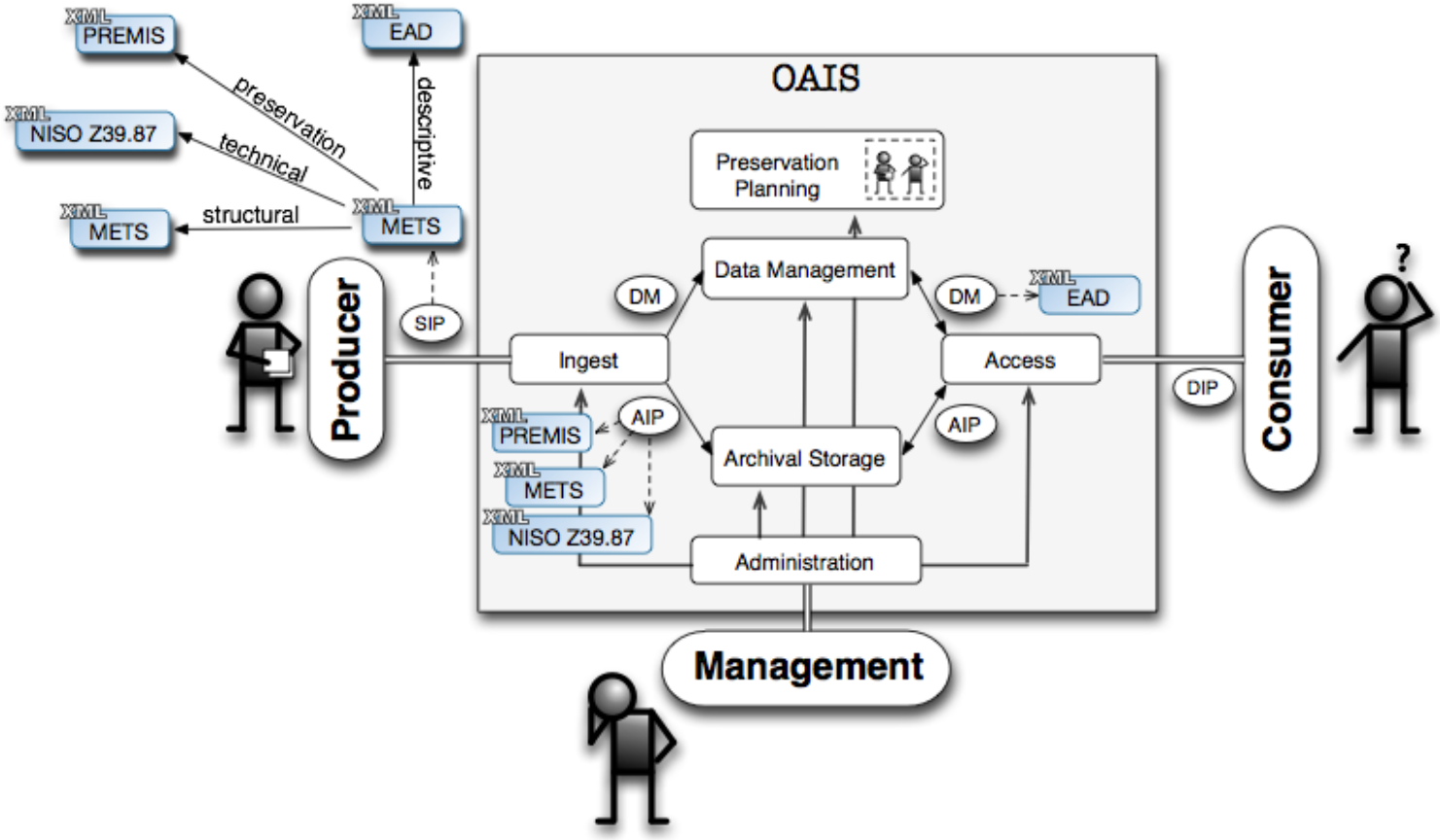
RODA follows the Open Archival Information System Reference Model (OAIS) **[CCSDS02]**. Among other things, the OAIS identifies the functional components that should be present in a digital repository system capable performing long-term preservation of digital materials. The proposed model is composed by four principal functional units: Ingest, Data Management, Archival Storage and Access. The Ingest process is responsible for accommodation of new materials in the repository and takes care of every task necessary to adequately store and preserve that information. For example, during this stage, an OAIS repository may transform the submitted objects to normalized formats adequate for long-term preservation (i.e. normalization) and request the user to add descriptive metadata to those objects to facilitate their future retrieval by search engines. It is also common practice to store the original bit-stream of the ingested material together with the normalized version (just in case a more advanced preservation strategy comes along to rescue those old bits of information).

New entries come in packages called SIPs ("Submission Information Packages"). When the ingestion process terminates, the SIPs are transformed into AIPs ("Archival Information Packages"), i.e. the actual packages that will be kept within the repository .

The Data Management component is responsible for providing services and functions for populating, maintaining, and accessing a wide variety of metadata that is stored by the repository. Some examples of this information are catalogs and inventories on what may be retrieved from Archival Storage, processing algorithms that may be run on retrieved data, Consumer access statistics, Consumer billing, Event Based Orders, security controls, and OAIS schedules, policies, and procedures. **[CCSDS02]**

The Access component establishes an interface between the archive and the end user (i.e. the consumer). This functional unit is able to locate an AIP by querying the Data Management component and retrieve it from the Archival Storage unit. The AIP is then transformed into a DIP (Dissemination Information Package) and delivered to the consumer. Figure 1 illustrates the OAIS model and gives a glimpse of which XML languages are being used in RODA.

Figure 1: General architecture of the repository and XML schemas in place.

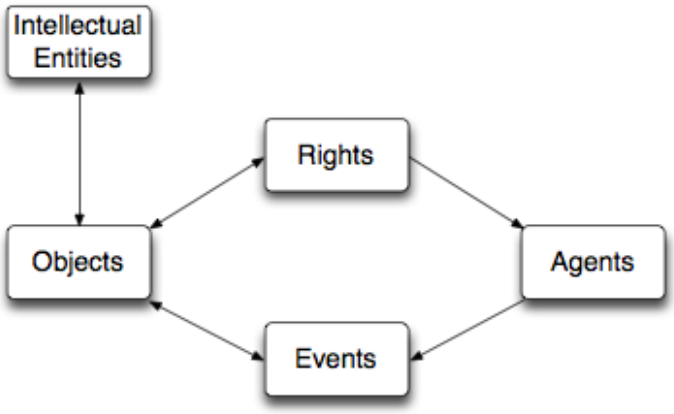


[Link to open this graphic in a separate page]

The content model

Every digital repository needs metadata. From the simplest of repositories, such as a domestic jukebox system created to store and manage a person's music collection, to a large governmental repository responsible for safekeeping the intellectual heritage of an entire country, metadata is used thoroughly to support even the most basic repository functions, e.g. providing access to stored items of information. A long-term preservation repository needs to maintain meaningful metadata of many types to be able to successfully carry out its mission. Descriptive metadata is essential to keep materials organized and easily accessible. Preservation metadata is equally important if a repository wants to be trustworthy and capable of proving evidence of the provenance and authenticity of the materials in its custody. Technical metadata are also critical for the safe-being of objects within the repository. Finally, structural metadata is essential for organising complex digital objects (e.g. a website composed of many files) and enable rendering applications to adequately process and display those objects.

Figure 2: PREMIS Data Model



[Link to open this graphic in a separate page]

One of the laying stones of preservation metadata is PREMIS (Preservation Metadata: Implementation Strategies, **[PREMIS]**), it defines a data model comprised of five entities: Intellectual Entities, Objects, Events,Rights and Agents (figure 2).

Intellectual Entity	Coherent set of content that is described as a unit, for example, a book, a map, a photograph, a database. An Intellectual Entity can include other Intellectual Entities; for example, a Web site can include a Web page, a Web page can include a photograph. An Intellectual Entity may have one or more Representations.
Object	or Digital Object, is a discrete unit of information in digital form.
Event	Action that involves at least one Digital Object and/or Agent known to the Preservation Repository.
Rights	Assertions of one or more rights or permissions pertaining to a Digital Object and/or an Agent.
Agent	Actor (human, machine, or software) associated with Events occurring over the course of a Digital Object's life cycle.

The PREMIS Data Dictionary defines semantic units for Objects, Events, Agents and Rights, but not for Intellectual Entities. This happens because the Intellectual Entity is out of scope and considered well served by descriptive metadata.

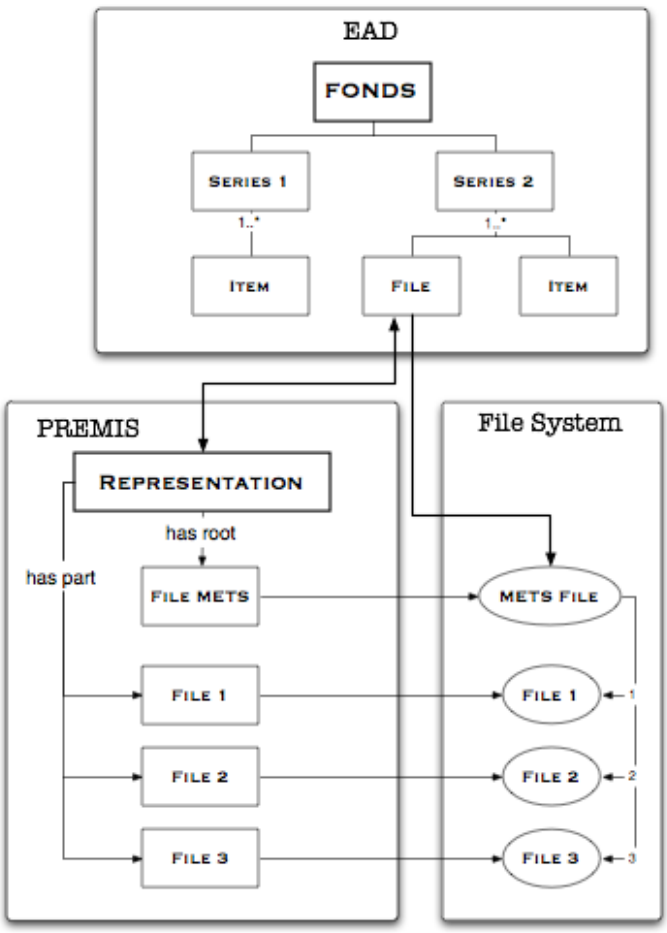
The RODA content model is based on the PREMIS data model. To suffice the data model, EAD is used as descriptive metadata to describe Entillectual Entities. Furthermore, the PREMIS Object is refined by a technical metadata schema (such as NISO Z39.87 for still images) to compensate for PREMIS generality.

Each digital object is described by an EAD record. These records are organized hierarchically within a EAD document. Each leaf record points to a PREMIS record containing information about the digital object provenance and event history. Finally, the PREMIS record points to the files that constitute the digital object.

An additional XML language for packaging: METS (Metadata Encoding and Transmission Standard). We use it for packaging SIPs, AIPs and DIPs.

Figure 3 illustrates the content model created for the central repository of AIPs.

Figure 3: AIP Data Model



[Link to open this graphic in a separate page]

Working with Relational Databases

In March, this year, the first International Workshop on Database Preservation **[International Workshop on Database Preservation]** took place in Edinburgh. The idea was to bring together specialists from academia and industry and to begin a brainstorming session about the subject of database preservation.

The following questions were discussed during the workshop:

- How do we keep archived databases readable and usable in the long term (at acceptable cost)?
- How do we separate the data from a specific database management environment?
- How can we preserve the original data semantics and structure?
- How can we preserve authenticity and provenance of databases?
- How can we preserve data while it continues to evolve?
- How can we have efficient preservation frameworks, while retaining the ability to query different database versions?
- How can multi-user online access be provided to hundreds of archived databases containing terabytes of data?
- Can we move from a centralized model to a distributed, redundant model of database preservation?

In RODA we already provide answers to some of these questions. In order to keep archived databases readable and usable in the long-term (at an acceptable cost) one must develop a repository capable of storing abstract representations of databases.

This abstract representation of the database will enable us to separate data and structure from the specific database management environment (DBMS). We will loose all the original functionality of the DBMS but we will keep the data and the structure readable and reusable. To keep the database structure we will describe it using a declarative markup language such as XML. To preserve the authenticity and provenance of the database we will document every event that involves the database since it enters the archival environment. To fulfil the last requirement we will use PREMIS.

The data in our repository is always frozen, i.e. it is either a snapshot of data in a particular instant of time or it is composed of data that is no longer in use. This means that we won't support insertions or updates on ingested databases.

RODA will always provide access to the latest accessible version of a database.

Access to huge amounts of data (terabytes distributed across hundreds of archived databases), distributed database models and redundancy are issues that we intend to cover in our future work.

How do we preserve a database

The first step in preserving a database is to create an abstract representation of its structure. We were looking for a format that could describe as many database attributes as possible. This format should be hardware and software independent to ensure long-term access. The obvious choice was XML. However, there are some technical issues that must be addressed in order to have a fully functional solution based on this technology.

Each database engine exports a database in its own format. The problem becomes even more acute if we intend to migrate from one database format to another. If we want N database engines to dialogue with each other, we will have to develop N*N-N converters. On the other hand, if we have a neutral intermediate representation, we only need N*2 converters. This fact is not new and motivated the creation of a specific group inside the OMG (Object Management Group) to address this problem. As a result of this effort, a standard called XMI (XML Metadata Interchange) was developed.

After studying that proposal, we verified that it was too complex for our objectives. We needed a simple and clean way to describe a relational database in XML, preserving as much data and innate properties as we could.

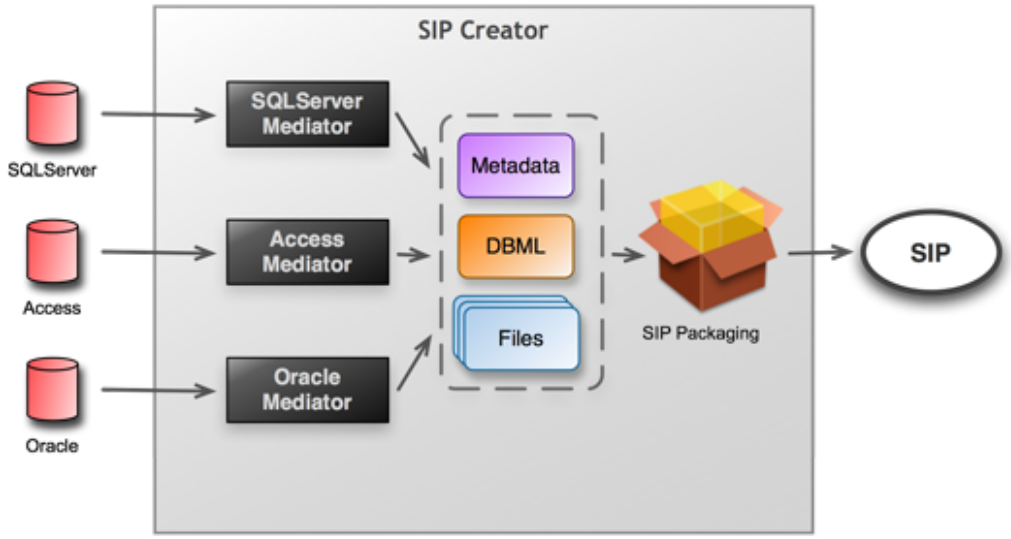
As stated before, XML was the format we wanted to use to represent any relational database. The only question remaining was: How? Develop a new language or use an existing one?

We decided to test our assumptions by building a small prototype of a database ingest module. The following sections provide a detailed view on the inner workings of this prototype.

Relational database ingest module

As in other components of RODA we are following the OAIS reference model to implement our database ingest module. This component has a structure very similar to the one presented in figure 1. The differences are in the SIP building process shown in figure 4.

Figure 4: Creating SIPs from a Database



[Link to open this graphic in a separate page]

Note that we have specific access mediators for each database engine. They all differ in the way they access content and database structure. For the moment we have created mediators for SQL Server 2000, Oracle 8 and Microsoft Access.

The SIP that contains the database is composed of:

- a METS file as a package wrapper;
- a EAD metadata record that describes the database (still under development);
- a set of DBML files that represent the original database structure and content;
- a set of binary files corresponding to blobs (binary large objects) found inside the original database (in DBML we have decided to add external pointers to these binary files instead of embedding them in the XML as base-64).

RODA is already prepared to ingest SIPs that respect this structure. Building the SIP is of the responsibility of the producer that submits information to the repository. However, we have developed an application that aids these users in this not so trivial procedure - the SIP creator.

In the next section we thoroughly describe DBML.

DBML (Database Markup Language)

As it will be shown, concerning information representation, XML is more embracing than a relational database (RDB). This means that it is always possible to represent information contained in a RDB as a XML document, but the contrary is not always possible.

To justify this point of view we will discuss and compare two important concepts: structured information and semi-structured information.

Structured Information

RDBs can be pointed out as the best example of structured information. In a RDB, the information is structured in tables which are, in turn, organized in rows (records) and columns (fields or attributes).

This rigid structure delivers some powerful advantages such as: access is made easy and efficient; the information can be validated and reused for very distinct kinds of results.

Nevertheless, this excessive structuralization comes with a few drawbacks. Presently, the most relevant issue is the lack of order of the record fields; in the description of the structure of a RDB, nothing indicates their order. For the typical database applications, this aspect is not relevant; yet, when we are storing data which obeys to a linear order, and this order must be preserved, we face a serious problem.

Semi-structured Information

Usually, the concept of semi-structured information appears associated to documents. In the context of descriptive markup, of which XML is a practical application, a document has a logical structure made explicit by the inclusion of marks in the text. Although a document, as a whole, is a structured piece of information, many of its parts are semi-structured, that is, they are composed by unstructured text and tagged-elements interleaved in an almost arbitrary way.

To clarify these concepts, we present the DTD for a class of documents known as poem (Figure 5).

Figure 5: DTD for a simple poem

```
<!ELEMENT poem (title, author, body, date) >
<!ELEMENT title (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT body (quatrain|tercet)+>
<!ELEMENT quatrain (verse, verse, verse, verse)>
<!ELEMENT tercet (verse, verse, verse)>
<!ELEMENT verse (#PCDATA|name)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT date (#PCDATA)>
```

This specification describes the structure that the documents of type poem must follow: the poem has a title, followed by an author, the body and, finally, the date; the title, the author and the date are free text components; the body, however, is a structured element composed by one or more quatrains or tercets, that are made up of verses.

Apparently we are before a rigid structure. But in reality, we are not: there is an element whose definition allows for mixed-content to be inserted; a verse is made up of free text within which name elements may occur, in any number or position.

These mixed-content elements and the free text elements allow us to state that an XML document is a semi-structured information container.

At this point we may conclude that there are two impediments to the transposition of the information from any XML document to a RDB. The first one is the existence of elements with mixed-content. The other has to do with the nature of text. A text has a linear order that must be preserved so that the text can actually make sense; changing the order of elements will inevitably change its meaning.

Classical RDBs do not have innate mechanisms to deal with linear order nor with semi-structured data. On the other hand, the XML coexists with structured information, semi-structured information, and even linear orders. We are, therefore, before a strong candidate for the constitution of an information interchange platform.

In our context we are dealing with information coming from RDBs thus the semi-structured problem is not an issue.

How to transform a database into XML

Whenever we intend to represent a database entity in DBML, we must remember that, we not only want to represent the information itself but also describe its context and specific properties (the so-called meta-information). In that case, we also want to describe the way data is stored in the original RDB.

A RDB has two components: structure and information. Our final DBML document will have, accordingly, two parts, one that describes the structure of the RDB and another one to store the data contained by RDB. The XML skeleton of our final DBML document will roughly resemble the one shown in figure 6.

Figure 6: DBML skeleton - general structure

```
<?xml version="1.0" encoding="iso-8859-1"?>
<DB name="XXX" date="today">
<STRUCTURE>
...
</STRUCTURE>
<DATA>
```

```
  ...
</DATA>
</DB>
```

The conversion of a RDB to XML must be able to tackle the following:

Structure	To fill the first part of the XML document we must access the structural information of the RDB and convert that information to an adequate XML structure. Each Database Management System has a particular way of storing this structural description of a database. The specific mediators will take care of this for each type of database.
Data	The transference of information from a RDB to XML is a more generic process. Almost every Database Management System allows the user to download the information in a database to a pure text file using pre-defined field and record separators. Therefore, the conversion problem to solve is the one of converting data from those text files to the second part of the final XML document. However, at the moment we have the mediators taking care of this too using common SQL queries that return the whole data available in a table.

Lee Buck **[Buc99]** and Ronald Bourret **[Bou05]** describe two possible approaches to data codification. However, there is no information available on the description of the database structure; even in commercial projects nothing is said about the philosophy and methodology adopted. In our case, we decided to develop an XML markup language (DBML) to describe the structure. We develop these two topics in the following subsections.

Converting the structure

The conversion is done by the mediators, one for each RDBMS. Mediators use Java Database Connectivity (JDBC) API to access the database structure information. When the JDBC driver for a specific RDBMS doesn't provide all the information needed or provides wrong information, specific methods are developed to extract the information directly from system tables.

The translation scheme adopted by the mediators is the following:

Tables	Each table is mapped to an element named *TABLE* that has an attribute called *NAME*.
Columns	Each column will be mapped to a *COLUMN* element that also has a *NAME* attribute, where the column's name is saved. Other properties like the data type for the values of that column and the characteristic of being empty or not, are stored in attributes *TYPE* and *NULL* associated to the *COLUMN* element. As a table contains more than one column, it is necessary to include in the XML document another element, *COLUMNS*, to aggregate all the *COLUMN* instances.
Primary and Foreign Keys	Keys may be defined inside the table definition; so it will be described as a sub-element *TABLE*. An aggregation element, *KEYS*, has to be introduced to group the various keys in a table. The set of keys shall also be divided into Primary and Foreign Keys; so *PKEY* and *FKEY* were introduced as sub-elements of *KEYS*. Moreover, a primary key in the relational model may be single (just one column) or compound (more than one column). To distinguish between the two cases, an attribute *TYPE* was associated to *PKEY* element, see figure 7. As foreign keys (of single type) relate one table with another one, the *PKEY* element shall be associated using the attributes *IN* (identifier of the destination table) and *REF* (identifier of the linked fields in the destination table).

Figure 7: Translation skeleton

```
    <TABLE NAME="Districts">
        <COLUMNS>
                    <COLUMN NAME="code" TYPE="int" NULL="no"/>
            ...
            </COLUMNS>
            <KEYS>
            <PKEY TYPE="simple">
                    <FIELD NAME=""/>
            </PKEY>
            <PKEY TYPE="compound">
                    <FIELD NAME=""/>
                    <FIELD NAME=""/>
            </PKEY>
            <KEY NAME="" REF=""/>
            ...
            </KEYS>
    </TABLE>
```

To illustrate the translation schema just described we will use the structure of a classical Products and Suppliers database that is composed of three tables: two tables are used to represent Products and Suppliers and a third table is used to implement the N:N relation that exists between the two.

The primary key of tables Products and Suppliers is single and is stored in the column CODE, in both cases; there are no more keys. Concerning the third table, p2s, its primary key is of a composed type and the field elements are cod-p and cod-s; this table also has two foreign keys, cod-p and cod-s that establish the links to the two other tables. The result of the conversion is shown in figure 8.

Figure 8: The Products and Suppliers DB Structure translated into DBML

```
<?xml version="1.0" ?>
 <DB>
  <STRUCTURE>
   <TABLE NAME="products">
    <COLUMNS>
     <COLUMN NAME="code" TYPE="nvarchar"
      SIZE="10" NULL="no"/>
     <COLUMN NAME="description" TYPE="nvarchar"
      SIZE="50" NULL="no"/>
     ...
    </COLUMNS>
    <KEYS>
     <PKEY TYPE="simple">
      <FIELD NAME="code"/>
     </PKEY>
    </KEYS>
   </TABLE>
   <TABLE NAME="p2s">
    <COLUMNS>
     <COLUMN NAME="cod-p" TYPE="nvarchar"
      SIZE="10" NULL="no"/>
     <COLUMN NAME="cod-s" TYPE="nvarchar"
      SIZE="10" NULL="no"/>
    </COLUMNS>
    <KEYS>
     <PKEY TYPE="composite">
      <FIELD NAME="cod-p"/>
      <FIELD NAME="cod-s"/>
     </PKEY>
     <FKEY NAME="cod-p" IN="products"
      REF="code"/>
     <FKEY NAME="cod-s" IN="suppliers"
      REF="code"/>
    </KEYS>
   </TABLE>
```

```
   <TABLE NAME="suppliers">
    <COLUMNS>
     <COLUMN NAME="code" TYPE="nvarchar"
      SIZE="10" NULL="no"/>
     <COLUMN NAME="name" TYPE="nvarchar"
      SIZE="60" NULL="no"/>
     ...
    </COLUMNS>
    <KEYS>
      <PKEY TYPE="simple">
        <FIELD NAME="code"/>
      </PKEY>
    </KEYS>
   </TABLE>
  </STRUCTURE>
  <DATA>
    ...
  </DATA>
</DB>
```

Converting the data

As it was told in the beginning of this section, the transference of a DB to a DBML document has two parts: the structure description (previously described) and the data (to be discussed in this section). According to **[Buc99]** and **[Bou05]** there are two approaches to generically acomodate RDB data in XML:

Via attributes	Each table row is mapped to one element and the values of its columns (fields) are mapped to attributes of that element;
Via elements	Each table row is mapped to one element and the values of its columns (fields) are mapped to child elements, one for each value.

The two proposals might look equivalent **[Kim97]**, and they are in terms of the information representation, but concerning the processing effort involved in each of them, the options are quite different. The processing of an XML documents is structure-oriented and the structure is given by the elements. The attributes play a secondary role in this. The second approach is, therefore, much more effective. In this project we opt for the second alternative, i.e. *via elements*.

To convert the information from a RDB to DBML the following translation schema is used:

1. For each table, an element with the correspondent name is created.
2. For each row in the DB, an element with the table name and the suffix "-REG" is created.
3. For each column, an element with the name of the corresponding column is created. Its content will be the value of that field. Empty fields give origin to elements without content.

The next figure (Figure 9) illustrates the data conversion principle using the previously introduced Products/Suppliers database.

Figure 9: The Products and Suppliers DB data to DBML

```
...
  <DATA>
    <products>
      <products-REG>
        <code> a122 </code>
        <description> milk </description>
        ...
      </products-REG>
      <products-REG>
        ...
      </products-REG>
    </products>
```

```
    ...
  </DATA>
...
```

This example concludes the DBML presentation. In the next section we will discuss some DBML issues within RODA and how we are currently solving them.

Database dissemination

Up till now we have seen how we are using XML as an abstract representation for databases (data and structure). Inside RODA's repository the AIPs will be very similar to the SIPs that are ingested. However, dissemination of such AIPs raises additional issues. Common users will not be interested in a large XML file representing what they are expecting to be a relational database. In order to resolve this issue, we have developed two distinct dissemination processes: SQL and HTML.

The SQL dissemination corresponds to the generation of a single standard SQL file. Users can use this file to re-create the original database in their own state of the art database management system and then use it to access de data.

The HTML dissemination is presented as a dynamic database browser that enables users to navigate through the database. Browsing through a large DBML would require heavy processing from the XSLT processor in place. For certain sizes, processing would be impossible all together. In order to solve this problem we have created a cache representation of DBML in a temporary relational database (see figure 10.

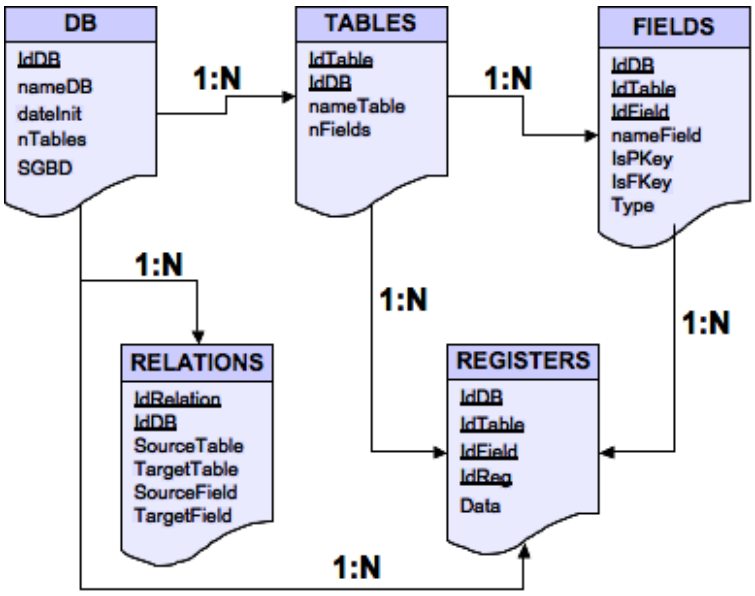
Figure 10: Database dissemination

[Link to open this graphic in a separate page]

Each AIP is translated back to SQL and this SQL description is given to a database engine (currently we are using MySQL for this purpose).

Since this cache is supposed to work with any kind of database (we are talking about a database of databases, i.e. a metadatabase) we had to create an abstract relational model suitable for that. The model shown in figure 11 is very similar to the ones used by database engines, we are just talking about one level up.

Figure 11: Database cache model



[Link to open this graphic in a separate page]

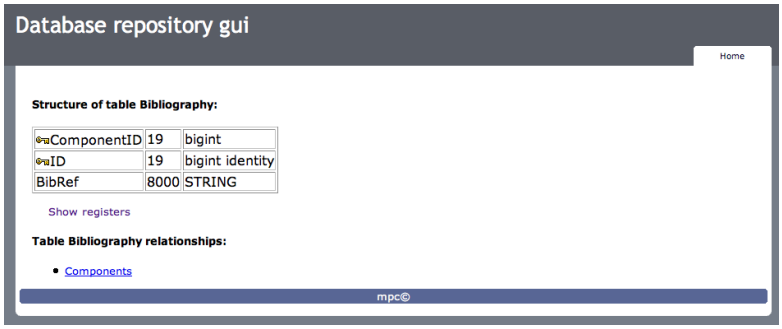
As you can see we have one table to store the database information, one to store table information, one for field information and another to store the data ("REGISTERS"). All data items are converted to a textual form in order to be stored in this table (blobs are converted to base 64). Although it may seem extremely inefficient this component works quite well in this prototype version.

We developed a navigator that enables users to browse the cached databases. Users may access and browse the database structure, access data items and navigate across relations. Figures 12, 13 and 14 present screenshots of this application.

Figure 12: RODA Database Browser - Basic Database Information

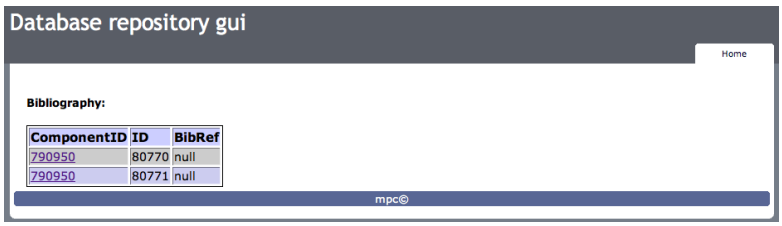
[Link to open this graphic in a separate page]

Figure 13: RODA Database Browser - Columns Information



[Link to open this graphic in a separate page]

Figure 14: RODA Database Browser - Registers Information



[Link to open this graphic in a separate page]

Final remarks

In this paper we have described one of RODA's components: the relational database ingest module. RODA will be under development for the next 15 months.

The first 12 months were used to think about the whole digital preservation problem, to specify the content model of the repository, to choose the main architecture and find preservation strategies adequate for the types digital object we have agreed to preserve.

During prototype implementation some problems have emerged:

- Our first case study has a model with more than 2.000.000 nodes. RODA is performing slower than we expected and we need to find faster solutions to this problem. In the next months we will experiment with a distributed architecture and with GRID computing.
- Size is a major problem, not only the number of nodes but the size of a node itself. Consider a large database (let's say, 5 Gb) that was converted to a single DBML file . Processing such an XML file is on it self a problem.
  RODA's architecture is Service Oriented and Web Services perform poorly with large volumes of data.

In the near future we are expecting to ingest a database with 26 terabytes of data. Ingesting this amount of data will be a challenge we hope to be ready to handle.

Acknowledgments

We would like to thank Pedro Faria, Marco Palos and Maria Barreira for all the time invested in the development of the DB ingest module described in this paper.

Bibliography

[Bou05] R. Bourret. XML and Databases. http://www.rpbourret.com/xmldbms/, 2005.

[Buc99] Lee Buck. "Data models as an XML Schema development method", XML 99, Phyladelphia, Dec. 1999.

[CCSDS02] Consultative Committee for Space Data Systems, "Reference Model for an Archival Information System (OAIS)", CCSDS 650.0-B-1, Blue Book, 2002.

[DPT01] Digital Preservation Testbed, "Migration: Context and Current Status," The Hague, White Paper, 2001.

[Gra00] S. Granger, "Emulation as a Digital Preservation Strategy," D-Lib Magazine, vol. 6, no. 10, 2000.

[International Workshop on Database Preservation] A workshop held on the 23rd of March of 2007 at the National e-Science Centre (NeSC), e-Science Institute (eSI) in the city of Edinburgh, UK - http://homepages.inf.ed.ac.uk/hmueller/presdb07/

[Kim97] E. Kimber. "Designing a DTD: Elements or attributes?"; http://www.oasis-open.org/cover/attrKimber9711.html, 1997.

[Lor01] R. A. Lorie, "Long Term Preservation of Digital Information," presented at First ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'01), Roanoke, Virginia, USA, 2001.

[Lor02] R. A. Lorie, "A Methodology and System for Preserving Digital Data," presented at Second ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02), Portland, Oregon, 2002.

[LRW03] L. S. Lin, C. K. Ramaiah and P. K. Wal, "Problems in the preservation of electronic records," Library Review, vol. 52, no. 3, pp. 117-125, 2003.

[LSLTM02] K.-H. Lee, O. Slattery, R. Lu, X. Tang and V. McCrary, "The State of the Art and Practice in Digital Preservation," Journal of Research of the National Institute of Standards and Technology, vol. 107, no. 1, pp. 93-106, 2002.

[MWS02] P. Mellor, P. Wheatley and D. M. Sergeant, "Migration on Request, a Practical Technique for Preservation," presented at ECDL '02: 6th European Conference on Research and Advanced Technology for Digital Libraries, London, UK, 2002.

[PREMIS] P. Caplan, R. Guenther et al, "Data Dictionary for Preservation Metadata," OCLC and RLG, 2005

[Rot99] J. Rothenberg, Commission on Preservation and Access and Council on Library and Information Resources, Avoiding technological quicksand: finding a viable technical foundation for digital preservation: a report to the Council on Library and Information Resources. Washington, DC: Council on Library and Information Resources, 1999.

[RRLRM05] D. S. H. Rosenthal, T. Robertson, T. Lipkis, V. Reich and S. Morabito, "Requirements for Digital Preservation Systems," D-Lib Magazine, vol. 11, no. 11, 2005.

[Rus03] A. Rusbridge, "Migration on Request," University of Edinburgh - Division of Informatics, 4th Year Project Report, 2003.

[SM98] T. Shepard and D. MacCarn, "The Universal Preservation Format: Background and Fundamentals," presented at Sixth DELOS Workshop, Tomar, Portugal, 1998.

[SM99] T. Shepard and D. MacCarn, "The Universal Preservation Format: A Recommended Practice for Archiving Media and Electronic Records," Boston, 1999.

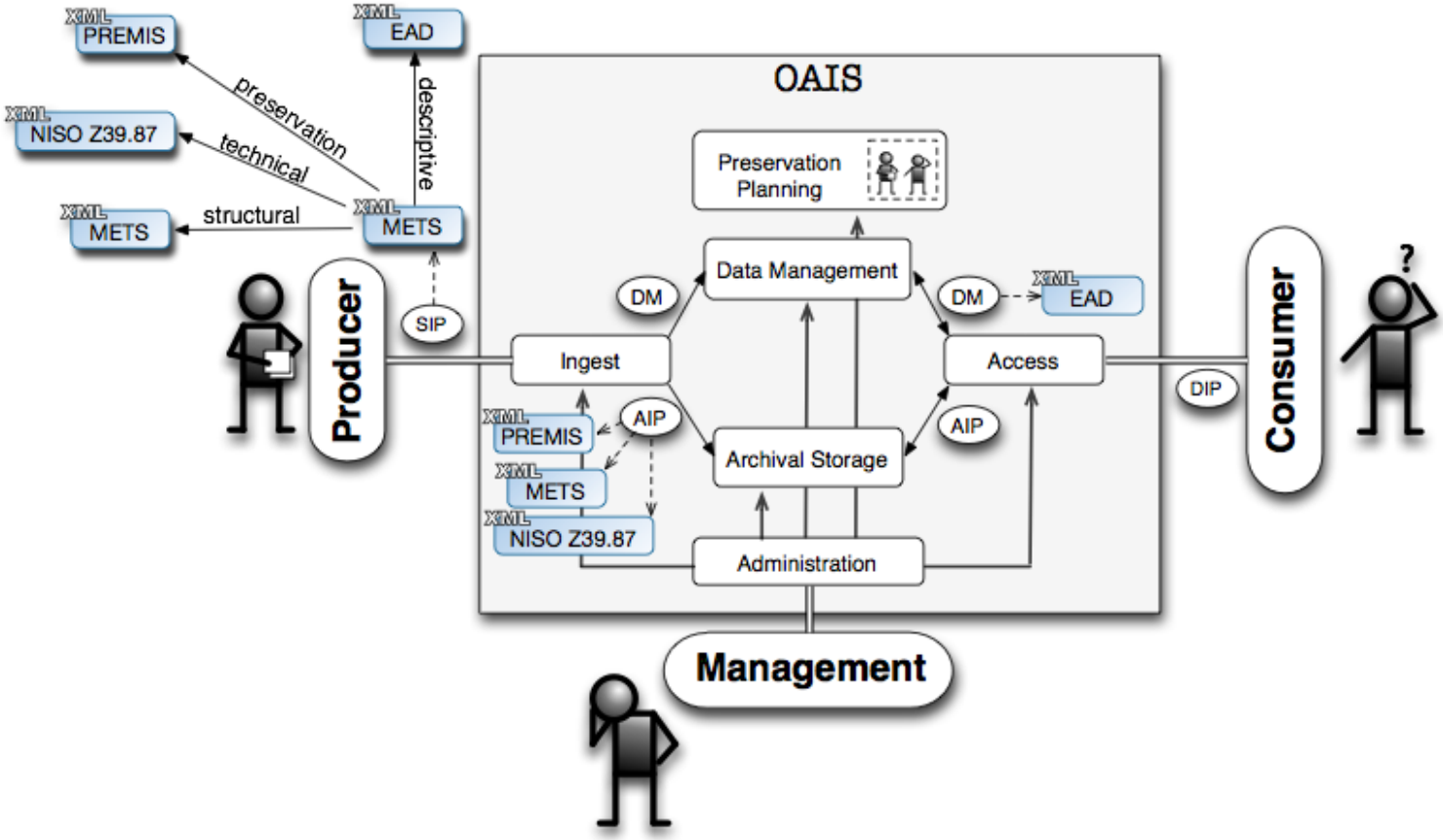
[TFADI96] Task Force on Archiving of Digital Information, Commission on Preservation and Access and Research Libraries Group, Preserving digital information: report of the Task Force on Archiving of Digital Information. Washington, D.C.: Commission on Preservation and Access, 1996.

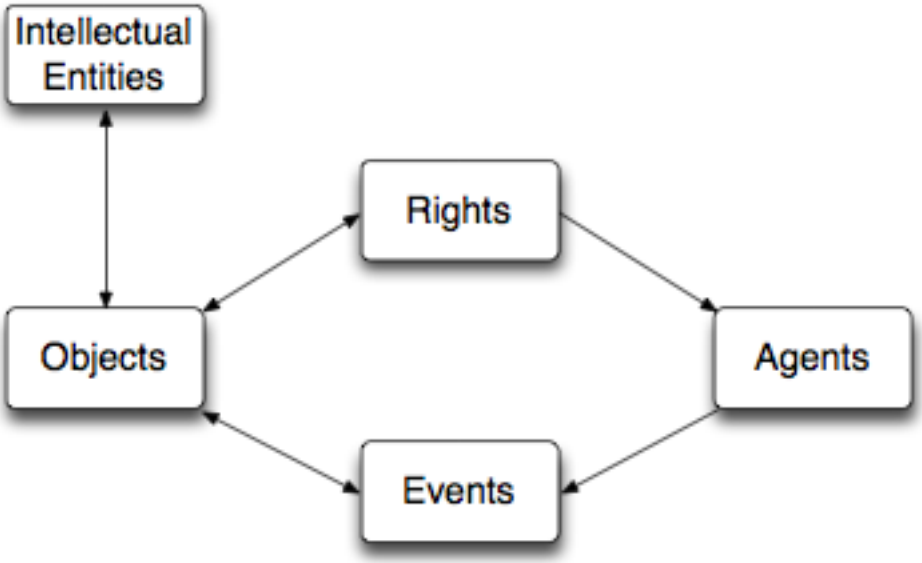
[Thi02] K. Thibodeau, "Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years," presented at The State of Digital Preservation: An International Perspective, Washington D.C., 2002.

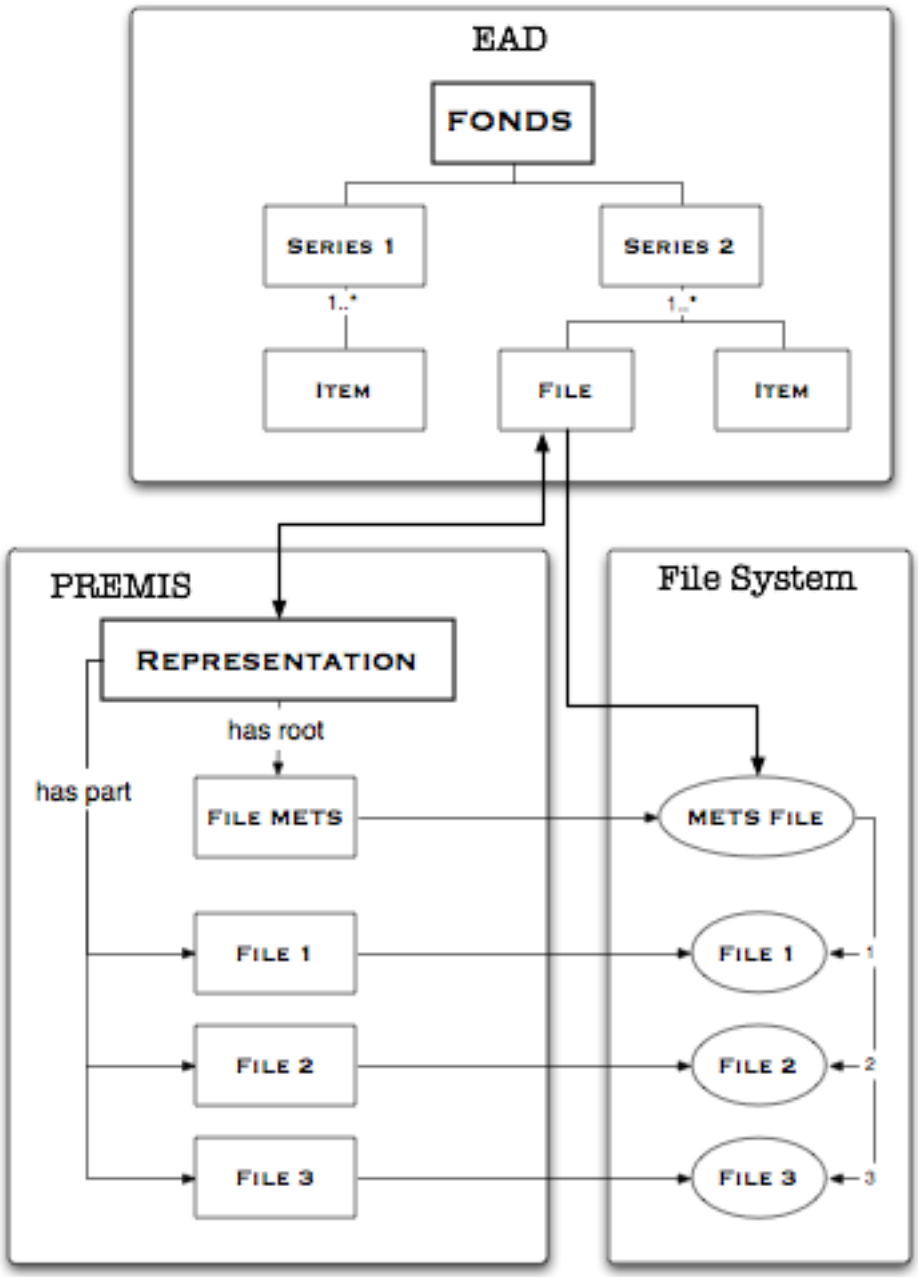
[Whe01] P. Wheatley, "Migration: a Camileon discussion paper," Ariadne, vol. 29, 2001.

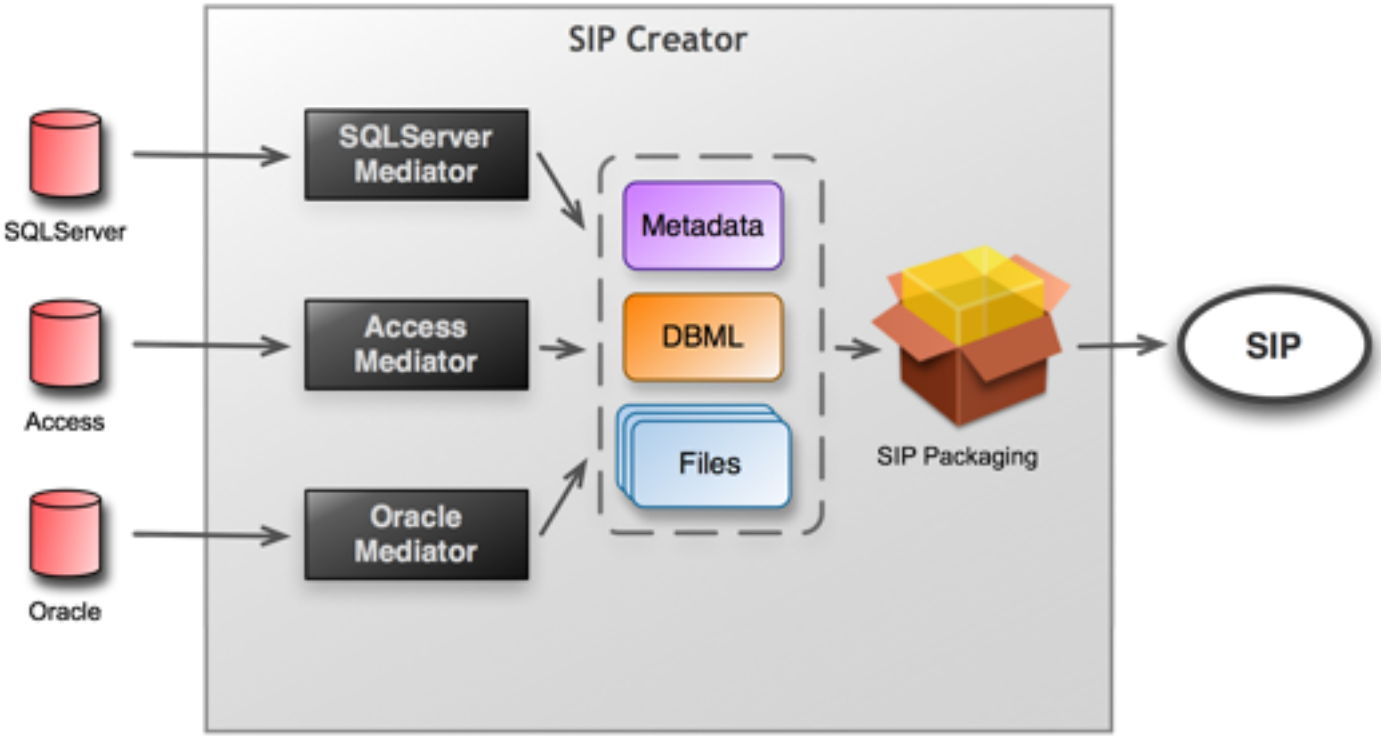
Relational Database Preservation through XML modelling

José Carlos Ramalho [University of Minho, Department of Informatics]
jcr@di.uminho.pt
Miguel Ferreira [University of Minho, Department of Information Systems]
mferreira@dsi.uminho.pt
Luís Faria [Portuguese National Archives]
lfaria@iantt.pt
Rui Castro [Portuguese National Archives]
rcastro@iantt.pt

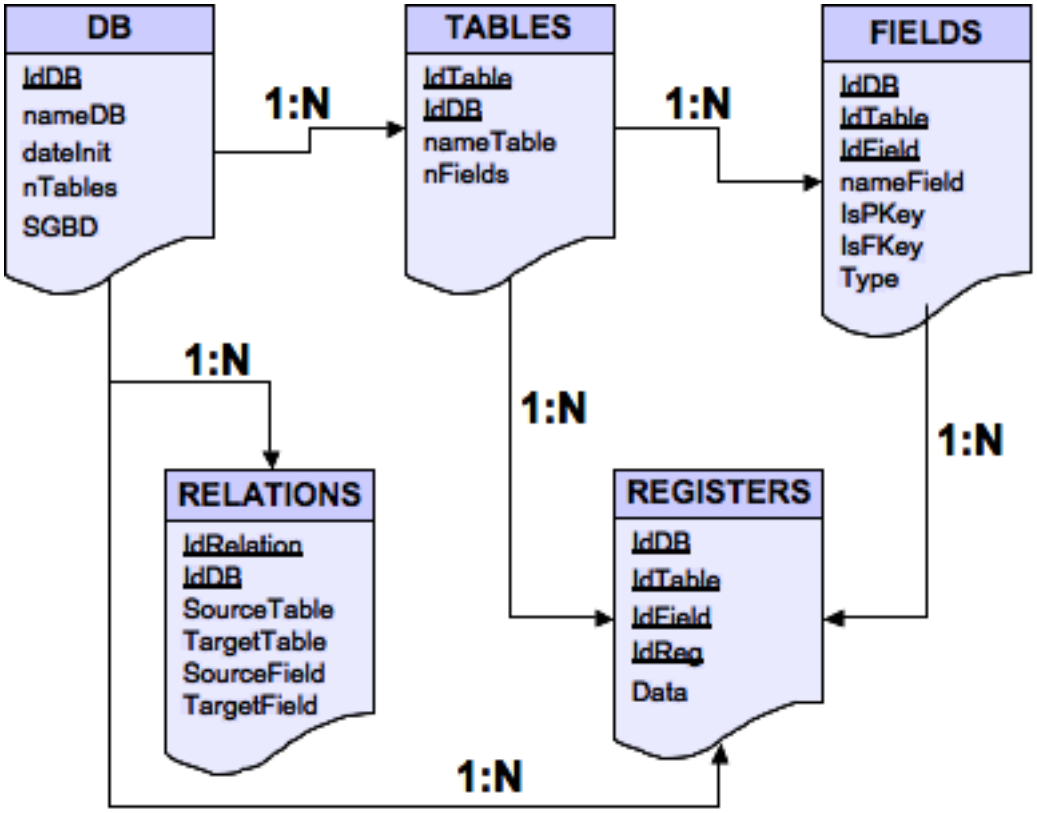

XML PREMIS
XML EAD
XML NISO Z39.87
XML METS
XML METS
preservation
technical
structural
descriptive
OAIS
Preservation Planning
Data Management
DM
DM
XML EAD
Producer
SIP
Ingest
Access
DIP
Consumer
XML PREMIS
AIP
AIP
Archival Storage
XML METS
XML NISO Z39.87
Administration
Management


Intellectual Entities
Rights
Objects
Agents
Events


EAD
FONDS
SERIES 1
SERIES 2
1..*
1..*
ITEM
FILE
ITEM
PREMIS
REPRESENTATION
has root
has part
FILE METS
FILE 1
FILE 2
FILE 3
File System
METS FILE
FILE 1
FILE 2
FILE 3
1
2
3


SIP Creator
SQLServer
Access
Oracle
SQLServer Mediator
Access Mediator
Oracle Mediator
Metadata
DBML
Files
SIP Packaging
SIP


AIP
Metadata
DBML
Files
Access
DIP
SQL
MySQL
Database
Browser


DB
IdDB
nameDB
dateInit
nTables
SGBD
1:N
TABLES
IdTable
IdDB
nameTable
nFields
1:N
FIELDS
IdDB
IdTable
IdField
nameField
IsPKey
IsFKey
Type
1:N
1:N
1:N
RELATIONS
IdRelation
IdDB
SourceTable
TargetTable
SourceField
TargetField
REGISTERS
IdDB
IdTable
IdField
IdReg
Data
1:N

Database repository gui

Home

Database details:

Name	DigitArq
Creation date	null
Export date	2007-04-18 17:27:15
Product name	Microsoft SQL Server
Product version	8.00.2039
Number of tables	19
SQL to restore original DB	download file

Show tables
Show relationships

Database repository gui

Home

Structure of table Bibliography:

ComponentID	19	bigint
ID	19	bigint identity
BibRef	8000	STRING

Show registers

Table Bibliography relationships:

- Components

Database repository gui

Home

Bibliography:

ComponentID	ID	BibRef
790950	80770	null
790950	80771	null