Preservação Digital de Bases de Dados Relacionais

Ricardo André Pereira Freitas

uMinho | 2008

Universidade do Minho
Escola de Engenharia

Ricardo André Pereira Freitas

Preservação Digital de Bases de Dados Relacionais

Junho de 2008

Universidade do Minho
Escola de Engenharia

Ricardo André Pereira Freitas

Preservação Digital de Bases de Dados Relacionais

Tese de Mestrado
Mestrado em Informática

Trabalho efectuado sob a orientação do
Professor Doutor José Carlos Ramalho

Junho de 2008

AGRADECIMENTOS

Este trabalho foi o culminar de um ciclo, no fundo o ultrapassar de mais uma etapa na minha vida académica e pessoal. As pessoas que me rodeiam no dia-a-dia e que sempre me transmitiram energia e me ajudaram durante este longo período merecem sem dúvida o meu reconhecimento. À família em primeiro lugar, e aqui são várias as pessoas importantes, contudo destaco a Sónia pelo apoio de todos os dias; os meus pais que sempre foram exigentes, e bem, relativamente às minhas capacidades, e à minha irmãzinha pela força que também me transmitiu.

A nível profissional agradeço também aos meus colegas e amigos do CIUL, que além das ajudas mais técnicas também me ajudaram nesta fase a vários níveis.

Um agradecimento final para o meu orientador José Carlos Ramalho pela ajuda na elaboração da dissertação e pelas boas orientações que sempre me concedeu.

DEDICATÓRIA

Para o meu Pai, que com suas contínuas interpelações, sempre me incitou e incentivou à conclusão deste trabalho.

De uma outra forma dedico ao meu filho André, cujo nascimento e crescimento veio acompanhando esta etapa da minha vida.

RESUMO

O ser humano sempre teve a necessidade de exteriorizar as aprendizagens adquiridas, sendo a linguagem um dos veículos para atingir esse fim. Desde o momento em que a informação deixa de ser transmitida apenas por meio de gestos ou sons voláteis e passa a ser esculpida no mundo que nos rodeia, a informação não mais se encontra presa à mente humana.
Segundo reza a História, os primeiros registos de informação ancestral são as gravuras rupestres. Surge a forma, que ainda hoje vigora, de transmissão de conhecimento por intermédio de documentos escritos diversificados. A possibilidade de registar o conhecimento ou a informação fora de nossos cérebros é uma capacidade fundamental para a perpetuação da mesma.
Durante muitos séculos, senão milénios, a informação era registada por forma que o ser humano a pudesse entender sem a necessidade de intermediários. Actualmente e devido à recente expansão das tecnologias de informação muito do conhecimento humano passa a estar registado em suportes informáticos. Este paradigma remete-nos para um cenário bem diferente. Passa a existir a necessidade do uso de intermediários – plataformas informáticas – para que a informação nos seja inteligível. O problema prende-se exactamente com os intermediários, ou seja, com as plataformas informáticas. Estas estão em constante evolução e nada nos garante a continuidade de acesso aos artefactos digitais na sua ausência. Irrompe uma nova problemática no universo digital: a Preservação Digital.
O nosso estudo aborda esta problemática da preservação digital e centra-se numa classe de objectos digitais específicos: Base de Dados Relacionais. As bases de dados relacionais desempenham um papel relevante no contexto global da informação digital e, por conseguinte, será fundamental não comprometer a sua longevidade, integridade e autenticidade. Este tipo de arquivos são de especial importância para as organizações uma vez que justificam as suas actividades e podem também fornecer uma imagem acerca da própria organização. Os objectos digitais são estruturas complexas que implicam o uso de hardware e software específico para que os seres humanos os possam interpretar e compreender.
Das diferentes abordagens e estratégias reiteradas por diversos investigadores coube-nos escolher as que mais se adaptam ao nosso caso concreto. Um formato neutro que seja independente das plataformas de hardware e de software é a chave para alcançar um modelo normalizado a usar na preservação digital de bases de dados relacionais. O XML pela sua neutralidade foi escolhido para essa representação da base de dados. Assim sendo procede-se à migração da base de dados original para uma linguagem definida a partir do XML, o DBML (*Database Markup Language*). O DBML foi definido de forma a permitir representar tanto a estrutura como os próprios dados existentes na base de dados.
O estudo desta temática foi realizado tendo como um dos objectivos a implementação de um protótipo para preservação digital de bases de dados relacionais. O modelo conceptual para o repositório segue as orientações do modelo de referência OAIS (*Open Archival Information System*). Este modelo define uma série de recomendações e assenta quatro eixos funcionais: Ingestão, Administração, Disseminação e Preservação.
A fase final dos trabalhos passou então pela implementação de um sistema capaz de ingerir bases de dados, sob a forma de pacotes de informação (DBML + meta-informação), para uma desejada preservação. O sistema desenvolvido teve por base uma aplicação *Web* com várias interfaces de forma a possibilitar, além da ingestão da informação, a sua administração, preservação e disseminação.

ABSTRACT

The human being always had the need to exteriorize the learning acquired, and starts doing that using the language. When the information no is longer transmitted only through gestures or volatile sounds and starts being sculpted in the world around us, the information is no longer locked inside the human mind.

According to the History, the earliest records of ancestral information were rock carvings. It emerge a form of knowledge transmission through a diversity of written documents. The possibility of registering knowledge or information out of our minds is a key capability to achieve its perpetuation.

For many centuries the information was recorded so that human beings could understand it without the need of any mediators. Today, and due to the recent expansion of technologies, much of human knowledge is now recorded on computer media. This paradigm brings us to a very different scenario. We now need to use intermediaries – computing platforms – to understand the information. The main problem concerns exactly with the computing platforms. These are constantly changing and evolving and nothing can guarantee the continuity of access to digital artefacts in their absence. A new problem in the digital universe arises: Digital Preservation.

Our study addresses this issue of Digital Preservation and focuses on a specific family of digital objects: Relational Databases. Relational databases are a very important piece in the global context of digital information and therefore it is fundamental not to compromise its longevity (life cycle) and also its integrity, liability and authenticity. These kind of archives are especially important to organizations because they can justify their activities and give us a glimpse about the organization it self. 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.

Of the different approaches and strategies repeated by several researchers we have to choose those that are best suited to our case. A neutral format that is hardware and software independent is the key to achieve a standard format to use in digital preservation of relational databases. XML for its neutrality was chosen for this representation of the database. We carried a migration of the database from its original form to a new format – DBML (Database Markup Language) – which is defined from the XML. DBML was established to enable representation of both structure and data of the database.

The study was conducted having as one of the objectives the implementation of a prototype for digital preservation of relational databases. The conceptual model for the repository follows the guidelines of the reference model OAIS (Open Archival Information System). This model defines a series of recommendations and builds four functional axes: Ingestion, Administration, Dissemination and Preservation.

The final phase of our work was the implementation of a system capable of ingest databases, in the form of information packages (DBML + metadata) for preservation. The developed system is based on a Web application and has multiple interfaces that allow not only the ingestion of information, but also its administration, preservation and dissemination.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE BLOCOS DE CÓDIGO

SIGLAS

AIP – Archival Information Package

API – Application Programming Interface

BD – Base de Dados

CCSDS – Consultative Committee for Space Data Systems

CD – Compact Disk

DBML – Database Markup Language

DIP – Dissemination Information Package

DOM – Document Object Model

DSN – Database Source Names

DTD – Document Type Definition

DVD – Digital Versatile Disc / Digital Video Disc

HTTP – Hypertext Transfer Protocol

IBM – International Business Machines Corporation

ISO – International Organization for Standardization

OAIS – Open Archival Information System

ODBC – Open Database Connectivity

OO – Orientado ao Objecto / Object-oriented

PHP – Hypertext Preprocessor

SGBD – Sistema Gestor de Bases de Dados

SIP – Submission Information Package

SQL – Structured Query Language

XML – eXtensible Markup Language

XPath – XML Path Language

1

Introdução

A preocupação com a preservação da informação, ao longo da história da humanidade, foi algo que sempre nos acompanhou e que é fundamental para a nossa própria evolução. Essa preocupação mantém-se e atendendo às especificidades dos dias que correm, uma parte dela centra-se agora nos artefactos digitais.

A preservação digital é sem dúvida uma área de investigação bastante promissora, contribuindo decisivamente para a preservação da nossa História que hoje se faz no universo digital. Ao assinalarmos o grande aumento do uso de ferramentas digitais nas últimas décadas, é possível verificar que a quantidade e diversidade de formatos de ficheiros e de ferramentas associadas, que vão surgindo, é também enorme. Isto deve-se à evolução e consequente melhoramento das tecnologias ao nível do desempenho computacional e ao nível das comunicações via redes informáticas [1]. A problemática da preservação digital coloca-se essencialmente devido a esta constante e rápida evolução, caracterizadora do universo das tecnologias de informação, tanto a nível físico – hardware – como a nível lógico e conceptual – software. Esta constante evolução obriga o ser humano a preocupar-se com a preservação dos artefactos digitais, para que estes continuem inteligíveis e acessíveis.

Poderemos definir a preservação digital como o conjunto de actividades ou processos responsáveis por garantir o acesso continuado, a longo-prazo, à informação e restante herança cultural existente em formatos digitais [2]. Importa esclarecer que um objecto digital é todo e qualquer objecto de informação que possa ser representado através de uma sequência de dígitos binários (*bitstream*) [3]. Esta definição é suficientemente abrangente para acomodar tanto a informação que nasceu num contexto tecnológico digital (objectos nado-digitais), como a informação digital obtida a partir de suportes analógicos (objectos digitalizados) [4]. Documentos de texto, fotografias digitais, diagramas vectoriais, bases de dados, sequências de vídeo e áudio, modelos de realidade virtual, páginas *Web*, e jogos ou aplicações de software são apenas alguns exemplos do que pode ser considerado um objecto digital.

Nesta dissertação vamos focar o nosso trabalho apenas numa das classes de objectos digitais: as bases de dados relacionais. As bases de dados relacionais são uma peça importante do contexto global da informação digital e por conseguinte será fundamental que a sua longevidade, integridade e autenticidade não sejam comprometidas. Este tipo de arquivo é especialmente importante para as organizações, visto que neles se encontram informações, que podem justificar as actividades da organização e até mesmo fornecer uma imagem sobre a própria organização. Situações que levem à perda efectiva de informações contidas em bases de dados podem também causar elevados prejuízos às organizações. Mesmo quando determinado sistema é descontinuado a informação contida nas suas bases de dados poderá vir a ser importante no futuro: por exemplo, como prova em tribunal de determinadas acções. Mas para que este cenário seja possível terão de ser observados vários requisitos sendo um dos mais importantes o acesso continuado à informação mesmo depois da descontinuação tecnológica.

São estas algumas das questões que justificam o estudo desta temática e que nos levam no sentido de olhar para as bases de dados como objecto de preservação.

1.1 Repositórios Digitais

Antes de avançar mais no que diz respeito à preservação de bases de dados é importante fazer alguma referência aos Repositórios Digitais.

Genericamente, referimos que um repositório digital é o local onde os conteúdos digitais podem ser armazenados para uso futuro. Esse uso futuro passará pela pesquisa e consulta dos conteúdos. Os repositórios digitais deverão possuir mecanismos para importar, exportar, armazenar e preservar os conteúdos ou a informação a eles associada. No âmbito dos repositórios digitais verificamos que podem existir por um lado as Bibliotecas Digitais e por outro os Arquivos Digitais.

Olhando primeiro para o caso das bibliotecas digitais, podemos começar por falar nas bibliotecas tradicionais, cuja informação encontra-se guardada ou armazenada em livros, ou seja, num suporte físico que é o papel. Nas bibliotecas digitais, o paradigma passa do suporte físico para um suporte digital. As colecções estão armazenadas em formato digital e passam a ser acessíveis por um computador. Dado que a informação se encontra armazenada neste tipo de formatos (digitais), esta pode ser acedida local ou remotamente através da Internet como todo o tipo de conteúdo digital.

Os arquivos digitais são diferentes das bibliotecas digitais essencialmente devido a três aspectos: nas fontes de informação, na organização dos conteúdos e na unicidade dos conteúdos [5]. As fontes da informação nos arquivos digitais são primárias – cartas, processos judiciais, registos paroquiais, etc. No caso das fontes primárias a informação provém directamente de quem a produz, por contraponto às fontes secundárias encontradas nas bibliotecas (livros). No que diz respeito à organização dos conteúdos, nas bibliotecas estes encontram-se catalogados individualmente, ao passo que nos arquivos os conteúdos são organizados em grupos; podendo ser agrupados pela sua proveniência e pela ordem que detinham originalmente. Os arquivos também se demarcam das bibliotecas devido à unicidade dos conteúdos. Nas bibliotecas, um

livro pode ser encontrado diversas vezes, enquanto que os registos de um arquivo devem ser únicos; quer isto dizer que os registos de um determinado arquivo só podem ser encontrados ou consultados nesse mesmo arquivo (podendo esse arquivo estar acessível via *Web*).

A problemática das bases de dados e a reutilização esperada das mesmas leva-nos a querer catalogá-las registando o produtor, a proveniência, etc. Por isso, serão objecto de armazenamento num arquivo digital

1.2 A Preservação

Em termos técnicos um arquivo digital será um repositório de objectos digitais que além da sua representação física (*stream* binária) têm a eles associada meta-informação descritiva e meta-informação de preservação.

Uma das nossas preocupações consistirá em garantir o acesso continuado e a longo-prazo à informação contida nas bases de dados relacionais armazenadas. O acesso a essa informação não significa um simples acesso aos bits que constituem o objecto digital, significa sim um acesso à informação contida na base de dados mas entendendo o que lá está, isto é, a informação tem de fazer sentido para quem a procura.

Apesar de a informação digital poder ser preservada de forma exactamente igual recorrendo apenas a uma simples cópia dos bits, que a constituem, isso não significa que mais tarde seja possível percepcionar o sentido da mesma. Verificamos, como foi já referido, que a evolução na área das tecnologias digitais é enorme o que constitui um obstáculo na inteligibilidade futura. Urge realçar que normalmente as plataformas informáticas perdem a sua capacidade de auto-preservação num prazo de sensivelmente 5 anos. O que permite que os bits do objecto digital sejam transformados em algo inteligível ao ser humano são exactamente as plataformas

(Software/Hardware) informáticas que se encontram em constante evolução.

Tendo em consideração estas várias contingências associadas à constante evolução das tecnologias digitais, importa reflectir sobre as estratégias a adoptar para preservar os objectos digitais (bases de dados relacionais no nosso caso).

1.3 A Dissertação

O objectivo desta dissertação situa-se na construção de um protótipo para o armazenamento de bases de dados relacionais. Esta classe de objectos tem características próprias que irão, por um lado determinar uma série de requisitos, e por outro levantar uma série de problemas no que concerne à implementação do protótipo. Foi realizado um estudo prévio no sentido de ultrapassar os vários problemas inerentes à especificação de um arquivo digital deste tipo. Dos vários problemas com que nos deparamos, um dos mais relevantes foi sem dúvida a especificação de um formato para a representação das bases de dados arquivadas. Este formato será mais tarde objecto de uma análise cuidada (Cap.5).

Antes da implementação foi então necessário estudar e contextualizar toda esta temática da preservação digital. Começamos por contextualizar o estado actual da preservação digital e estratégias associadas. No que concerne às estratégias para a preservação digital, verificamos que existe já algum consenso devido aos vários estudos efectuados nesta área. Algumas destas estratégias abordadas no capítulo 3 são, a “preservação tecnológica”, “emulação”, “refrescamento”, “migração”, “normalização” e “encapsulamento” [1]. Para cada uma das estratégias, foram analisadas as vantagens e desvantagens, a par de outras questões que se revelaram importantes para que nos fosse possível eleger as estratégias mais adequadas ao nosso

caso – bases de dados relacionais. O nosso estudo levou-nos a optar por utilizar uma conjugação de estratégias: “refrescamento”, “migração” e “normalização”.

A normalização é crucial a fim de garantir o uso de formatos neutros, ou seja, independentes de plataformas. A migração será necessária, visto que teremos que manter de alguma forma a informação preservada em formatos considerados actuais. Os dispositivos físicos, que dão suporte à informação a preservar, devem também manter-se actualizados e portanto é usada a técnica do refrescamento. Esta última não será uma estratégia por si só, porém em articulação com outras revela-se fundamental.

Como requisito para a implementação do protótipo final, surge o modelo de referência OAIS [6] (*Open Archival Information System*). Este modelo assenta em quatro eixos funcionais: Ingestão, Administração, Disseminação e Preservação. O processo de Ingestão é responsável pela incorporação de nova informação no repositório. O componente de Administração é responsável por manipular e gerir toda a informação no interior do repositório. O componente de Disseminação é responsável pela localização, transformação e extracção de informação armazenada do repositório. O componente de Preservação trata das políticas conducentes à manutenção do acesso a longo-prazo à informação armazenada.

O modelo de referência OAIS revelou-se de especial importância ao servir como guia, esclarecendo muitas questões que foram surgindo ao longo dos trabalhos. Este modelo não é rígido, nem obriga ao uso de tecnologias específicas na elaboração do protótipo. Realce-se que o modelo aponta uma série de recomendações e define também algum vocabulário nesta área da preservação digital, mais concretamente repositórios de informação. Segundo este modelo de referência, deve ser elaborado um repositório ou arquivo que permita a ingestão de informação sob a forma de pacotes de informação por parte de um Produtor. O acesso à informação será efectuada pelo Consumidor, a quem esta será entregue também sob a forma de pacotes de informação. Aquando do processo de ingestão, os pacotes de informação passam de SIPs (*Submission Information Package*) para AIPs (*Archival Information Package*) depois de

ingeridos no sistema. Aquando do processo de disseminação, os pacotes são transformados em DIPs (*Dissemination Information Package*) prontos a ser entregues ao Consumidor [7]. Quer num quer noutro processo terá que existir uma negociação prévia entre os agentes. No capítulo 4 iremos abordar com mais detalhe o modelo de referência OAIS.

No capítulo 5, debruçar-nos-emos sobre o formato para o qual a base de dados será transformada. Uma breve abordagem às bases de dados em geral e ao modelo relacional em concreto será também neste capítulo realizada.

A ideia é normalizarmos as bases de dados para um formato que deverá ser único e neutro. Desta forma, definimos que apenas serão ingeridas no repositório as bases de dados que respeitem o formato adoptado. Isto traz imensas vantagens, uma vez que assim se garante que as bases de dados respeitem critérios determinados. O formato adoptado foi o XML [8] (*eXtensible Markup Language*). As vantagens que apresenta ao nível da neutralidade e também a aceitação que tem tido pela comunidade científica revelaram-se fundamentais na escolha do referido formato. Estando estabelecido o formato a usar, foi necessário definir um *DTD/XMLSchema*[1] que norteia a estrutura do documento XML. Esse dialecto foi denominado de DBML [9] (*Database Markup Language*). Detém uma estrutura própria e irá comportar a base de dados quer a nível dos próprios dados como a nível da estrutura da base de dados.

No que concerne à implementação prática do protótipo, salientamos o uso de uma plataforma *Web* para esse efeito. Esta plataforma foi desenvolvida de raiz e envolveu várias tecnologias. Todas as tecnologias e normas envolvidas na construção do repositório serão descritas e analisadas no próximo capítulo.

Depois de analisadas as tecnologias envolvidas, as estratégias de preservação a seguir, estabelecido o modelo que irá orientar os trabalhos na construção do repositório e estando também definido o formato no qual a base de dados deve ser preservada, chegamos à parte da implementação prática do sistema.

1 http://www.w3.org/

Será no capítulo 6 que iremos falar da arquitectura do repositório. Além da arquitectura do sistema, daremos especial atenção às características técnicas do sistema implementado. Questões como a extracção dos dados das bases de dados e posterior ingestão no repositório serão detalhadas em termos técnicos.

No último capítulo, iremos tecer algumas conclusões acerca do estudo e do protótipo desenvolvido. Iremos também ponderar os resultados obtidos em função dos testes efectuados ao sistema e, perspectivar algumas intervenções futuras.

2

Tecnologias Envolvidas

Passamos agora à análise e descrição das tecnologias envolvidas para a concretização do repositório de bases de dados relacionais. Começamos por falar do sistema operativo e servidor *Web* que dão suporte ao repositório. Em seguida, debruçar-nos-emos sobre o ODBC (*Open Data Base Connectivity*), ferramenta que nos permite estabelecer conexões às diferentes bases de dados, e também sobre o modelo relacional [10] e o SQL [11] (*Structured Query Language*). Posto isto iremos tecer algumas considerações acerca do XML e toda a sua envolvência. A linguagens de programação PHP[1] (*Hypertext Preprocessor*) também será abordada assim como o SGBD (Sistema Gestor de Bases de Dados) MySQL[2]. Terminamos este capítulo fazendo referência a duas normas importantes no contexto da meta-informação – *Dublin Core*[3] e METS[4] (*Metadata Encoding and Transmission Standard*).

1 http://www.php.net/

2 http://www.mysql.com/

3 http://dublincore.org/

4 http://www.loc.gov/standards/mets/

2.1 Sistema Operativo – Linux

No que concerne ao Sistema Operativo Linux[1] é de realçar que este se baseia na arquitectura Unix[2]. Esta arquitectura tem algumas características muito interessantes nomeadamente a sua portabilidade e as suas capacidades de multitarefa e multiutilizador. O Sistema Operativo Unix foi originalmente criado por Ken Thompson já o Linux foi desenvolvido posteriormente por Linus Trovalds com a ajuda de uma comunidade na Internet que se estendeu por várias partes do globo. O seu código fonte está disponível para todos uma vez que foi desenvolvido sob o paradigma *GNU General Public License*[3]. O Linux tem na sua portabilidade uma das suas mais valias; este sistema operativo corre num vastíssimo número de plataformas, que vão desde grandes *mainframes*[4] até um simples computador portátil.

Quanto à sua arquitectura, salientamos a existência de um *kernel* que no fundo é o núcleo do sistema operativo. O *kernel* é responsável por lidar directamente com o hardware e é executada num local privilegiado da memória. O *kernel* faz então toda a gestão da memória, procede ao agendamento de processos, controla os dispositivos de hardware, enfim trabalha a um nível mais baixo. Por outro lado existem os programas ou software de sistema que se encontram separados, no que diz respeito à memória utilizada, do *kernel*. Esta característica também funciona como uma protecção para que este tipo de programas não interfira directamente com o sistema operativo. Algumas destas aplicações que fazem a ponte entre o *kernel* e o utilizador são as bibliotecas escritas na linguagem de programação C, a *Shell*[5], o Ambiente Gráfico (Servidor X) e outros programas. Saliente-se que também que o Linux existe em várias distribuições.

1 http://www.linux.org/

2 http://www.unix.org/

3 http://www.gnu.org/licenses/gpl.html

4 Mainframe é um computador de grande porte, dedicado normalmente ao processamento de um volume grande de informações.

5 Interface que permite ao utilizador interagir com o computador através da introdução de comandos

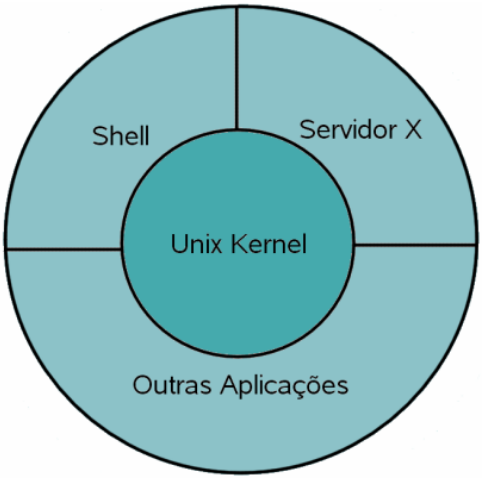

Figura 1 – Arquitectura do Sistema Linux

Muito mais haveria para dizer acerca do sistema operativo Linux, porém o que mais importa será justificar a nossa escolha. Escolhemos o Linux, primeiramente por ser um sistema aberto e de livre acesso. O que significa que podemos aceder e manipular o código fonte do sistema. Outros sistemas existem que são fechados e o acesso ao código fonte é vedado, assim como a capacidade de perceber o que se está a passar num determinado momento do seu funcionamento.

A sua portabilidade já mencionada, também se revelou importante, uma vez que olhando para a nossa área de estudo – preservação digital – será com certeza importante trabalhar com sistemas abertos e portáveis para diferentes plataformas. Quanto menos nos comprometermos com determinadas plataformas tecnológicas – hardware – mais possibilidade teremos no futuro, que o sistema continue a existir.

O sistema operativo Linux possui também primitivas que permitem interagir com um grande número de bases de dados – o tipo de objectos digitais que pretendemos preservar. Isto é

efectuado através de ligações ODBC. O projecto denominado *unixODBC*[1] *project* será utilizado para levar por diante a tarefa de ligação às diferentes bases de dados. Mais à frente também falaremos um pouco sobre o standard ODBC e também sobre o *unixODBC project*.

Com o sistema operativo a ser utilizado para o repositório definido partimos para a próxima camada que será o servidor de páginas *Web* a utilizar.

2.2 Servidor de páginas de Internet – Apache

Como já referimos, para o nosso protótipo, definimos que iríamos usar uma plataforma *Web*. Houve então a necessidade de escolher um servidor de HTTP. A nossa escolha recaiu sobre o servidor Apache[2], sendo este um servidor para páginas de Internet sobejamente usado a nível mundial. O servidor HTTP Apache foi criado em 1995 por Rob McCool. O servidor Apache é mantido pelo *HTTP Server Project*[3] e voltamos a falar em código fonte aberto.

Em termos de arquitectura, salientamos que se trata de uma arquitectura modular. Temos duas componentes principais: *core* ou núcleo e os módulos. O *core* é responsável pela implementação das funções básicas despoletando *threads*, processos. Os módulos acrescentam ou alteram funcionalidades já existentes no servidor. Os módulos podem ser criados por utilizadores uma vez que o servidor Apache disponibiliza uma API (*Application Programming Interface*) para esse efeito. Exemplificando, podemos referir o nosso caso concreto em que iremos utilizar a linguagem de programação PHP. Esta linguagem de programação, assim como outras, é adicionada/compilada como um módulo, para que o servidor a possa entender/interpretar.

1 http://www.unixodbc.org/

2 http://www.apache.org/

3 http://httpd.apache.org/

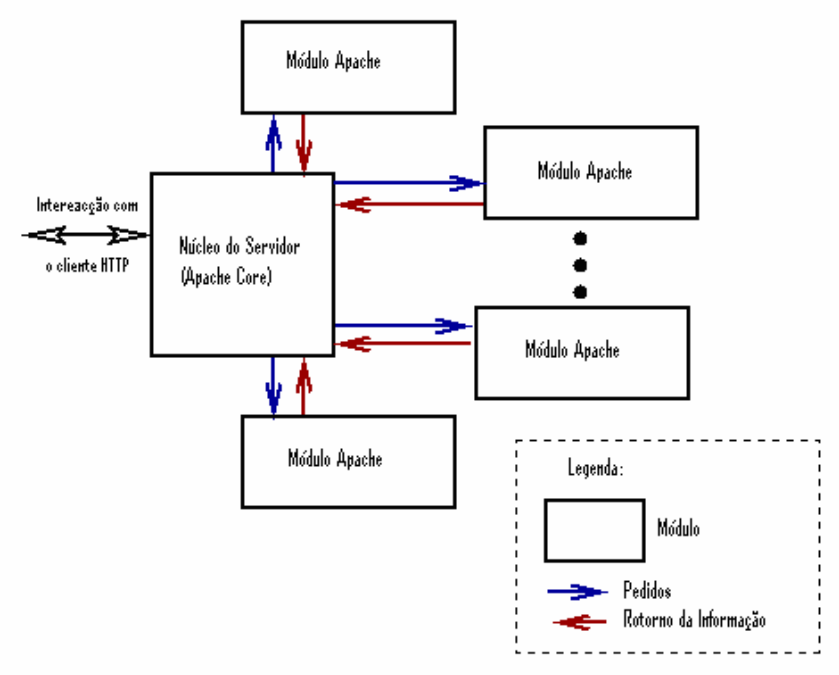

Figura 2 – Arquitectura Servidor Apache

Em termos de funcionamento falamos do paradigma cliente-servidor. Existe um fluxo de informação que parte inicialmente como um pedido – acesso a determinada informação numa qualquer página de Internet – por parte de um utilizador/cliente, recepcionada pelo servidor HTTP. Em seguida, o servidor irá responder a esse pedido enviando para o cliente a resposta, que será por exemplo a página de Internet pedida. Entre o pedido e a resposta, o servidor poderá ter que executar algo, mediante o que constar do código da página, seja simples HTML, ou *scripts* complexas em PHP, ou outra linguagem de programação.

2.3 Conexão às Bases de Dados – ODBC

Um dos pontos-chave do nosso estudo é sem dúvida o objecto digital base de dados. Sabemos

que existem diferentes Sistemas Gestores de Base de Dados (SGBD) – SQL Server, Oracle, MySQL, DB2, Microsoft Access, PostgreSQL – só para citar alguns. Rapidamente constatamos que certamente iríamos lidar com muitos destes SGBD, por isso surgiu a necessidade de encontrar uma forma que nos possibilitasse a interacção com o maior número de SGBD possível. ODBC *(Open Data Base Connectivity*) é sem dúvida a resposta para esta necessidade, uma vez que é um standard para comunicação com bases de dados. Foi desenvolvido no sentido de ser independente de linguagens de programação, de SGBD e de sistemas operativos. O ODBC proporciona uma abstracção ao SGBD, ou seja, podemos escrever o nosso código na linguagem de programação que entendermos, sem nos preocupar com o SGBD com que iremos trabalhar.

Muitas das vezes, usamos *Structured Query Language* (SQL) embebida na linguagem de programação escolhida para interagir com a base de dados. Outras vezes, apenas invocamos um procedimento da base de dados (*Store Procedure*). Algumas linguagens de programação também possuem funções específicas que, através do ODBC, conseguem obter informações específicas da base de dados, tais como nome das tabelas, domínio dos dados dos atributos, entre outros. A ponte entre a parte aplicacional e fonte de dados (*data source*) é conseguida devido à existência de *drivers*[1] específicos para os diferentes SGBD. O sistema operativo é ele próprio responsável por gerir os *drivers* que detém e que são fornecidos pelos fabricantes/vendedores dos SGBD. Não iremos analisar os *drivers* propriamente ditos uma vez que se entende que isso sai fora do âmbito do nosso estudo. Compete depois às aplicações utilizar os *drivers* que o sistema operativo tem para aceder aos diferentes SGBD.

Voltamos a referir a existência do projecto denominado *unixODBC project*, que fornece uma ferramenta para ligação às bases de dados usando exactamente ODBC. Uma vez que esta ferramenta foi desenhada para correr em vários sistemas operativos, incluindo o Linux, optámos por fazer uso dela para ligações às bases de dados.

1 Driver é um programa de software que permite interagir com um determinado dispositivo (Hardware ou Software)

Em várias distribuições do Linux, a ferramenta *unixODBC* vem já incluída, mas caso isso não aconteça podemos sempre instalar o pacote correspondente à ferramenta *à posteriori*. Esta ferramenta possui essencialmente dois ficheiros de configuração, um que irá indicar os vários caminhos para os diferentes *drivers* e outro no qual se configuram as fontes de dados. O ficheiro */etc/odbcinist.ini* é então o ficheiro que define e faz a gestão dos *drivers* do sistema operativo. O ficheiro */etc/odbc.ini* pode configurar as fontes de dados – DSN (*Data Source Name*) – indicando qual o *driver* (configurado no *odbcinist.ini*) que será usado, o servidor ou máquina na qual se encontra a base de dados, a porta *TCP/IP*[1] a usar, nome da base de dados, utilizador e palavra-chave. Nem todos estes parâmetros serão de preenchimento obrigatório, pois depende do SGBD ao qual nos pretendemos ligar. Eventualmente também podemos definir em tempo de execução alguns destes parâmetros, aquando da ligação efectuada pelo PHP. Assim sendo, o PHP irá tentar estabelecer a ligação através da função *odbc_connect,* que usa exactamente a ferramenta *unixODBC* configurada anteriormente. Se for obtido sucesso na ligação, é criado um identificador para essa mesma ligação e estamos prontos para interagir com a base de dados.

Existe um conjunto de funções ODBC específicas do PHP, que podem devolver por exemplo o nome das tabelas da base de dados, características das tabelas e dos atributos e logicamente também permitem efectuar *querys* à base de dados usando SQL.

2.4 Modelo Relacional

O modelo relacional para bases de dados é dos mais utilizados e amplamente popular a nível global. Edgar Frank Codd foi o matemático que teorizou este modelo aquando do seu trabalho de pesquisa na IBM. Tendo por base a matemática o modelo assenta as suas propriedades e características na teoria dos conjuntos [10]. A manipulação dos dados no modelo relacional é conseguida através da álgebra relacional.

1 TCP/IP é um conjunto de protocolos de comunicação de computadores via rede

A relação é o conceito fundamental para este modelo, no entanto em termos de base de dados este conceito apresenta-se como sendo uma tabela (linhas e colunas) ou entidade. Falamos então de tabelas com duas dimensões em que numa das dimensões teremos atributos (campos) e na outra dimensão iremos ter os tuplos (registos).

Estas tabelas apresentam-se ao ser humano com os atributos correspondendo às colunas e os tuplos a corresponder às linhas da tabela; esta é a forma tabular usada no modelo relacional, no entanto o modelo pode e certamente irá conter mais que uma tabela.

Relativamente aos atributos estes possuem um domínio, ou seja, ter-se-á sempre que se definir o tipo de dados que pode ser guardado para cada atributo. Um determinado atributo pode ou não ser nulo – obrigatoriedade de preenchimento do campo. Um atributo diz respeito a uma determinada característica que incidirá sobre todos os tuplos; por outras palavras podemos dizer, usando um exemplo de uma relação de clientes, o atributo nome do cliente (corresponde a uma coluna da tabela) é um dado comum a todos os clientes ficando estes últimos distribuídos pelos tuplos/linhas da tabela.

Ainda no que diz respeito à tabela ou relação é fundamental referir que em cada uma das relações é necessário estabelecer uma chave (chave primária) que irá corresponder a um ou mais atributos que identifiquem de forma única um determinado tuplo – garantia de unicidade. Iremos ver mais adiante que existem também chaves estrangeiras que irão definir os relacionamentos entre relações.

Os relacionamentos nada têm a ver com as relações, uma relação como já vimos corresponde a uma tabela. Os relacionamentos dizem respeito às associações entre as tabelas. Um relacionamento é conseguido através da ligação entre atributos – chaves – de duas entidades; falamos da ligação entre a chave primária (pode corresponder a um ou a mais atributos) numa das entidades e a chave estrangeira na outra entidade que faz parte do relacionamento. Num relacionamento, a chave primária e a chave estrangeira têm que ser iguais, isto é, se a chave primária contém dois atributos a chave estrangeira também terá que ter; o domínio das chaves

também terá que ser o mesmo.

Os relacionamentos possíveis são os de um para um, um para muitos e muitos para muitos. No caso dos relacionamentos de muitos para muitos, procede-se a um desdobramento de forma a obter dois relacionamentos de um para muitos. Isto é conseguido através da criação de uma relação na qual participam as chaves primárias das duas tabelas que, inicialmente, sustentavam o relacionamento de muitos para muitos. Procede-se desta forma para evitar redundância nos dados.

Ao desenhar ou modelar a base de dados, temos que caminhar sempre no sentido da normalização, evitando assim redundâncias e fazendo com que os dados numa mesma relação façam sentido. A normalização dos dados consiste num processo que tem por objectivo garantir a consistência dos dados de forma a possibilitar um acesso eficiente aos mesmos. O processo de normalização passa por três estados, também chamados de formas normais. As formas normais originais são três e são cumulativas [10] (primeira forma normal, segunda forma normal e terceira forma normal).

2.5 SQL

Falemos agora um pouco do SQL, que é a linguagem comummente usada para trabalhar com bases de dados. Esta linguagem interage com as bases de dados aos mais diversos níveis, desde a criação e manipulação da estrutura da base de dados até às consultas e alterações dos próprios dados.

A linguagem SQL encontra-se normalizada e segue os *standards* ISO e ANSI, contudo existem algumas extensões adicionadas individualmente por diferentes proprietários de SGBD. A primeira versão apareceu no início dos anos 70, pela mão de Donald D. Chamberlin e Raymond F. Boyce na IBM [11]. Esta linguagem foi criada com o intuito de lidar com um sistema

proprietário da IBM denominado de *System R* [12], baseado no modelo relacional.

Voltando ao SQL, salientamos que se trata de uma linguagem muito específica, vocacionada para trabalhar com bases de dados, sendo esse o objectivo da sua criação. A linguagem encontra-se dividida em três tipos de comandos: aqueles que servem para criar, alterar ou apagar tabelas, no fundo manipular a estrutura da base de dados – *Data Definition Language* (DDL); os que são usados para inserir, actualizar, apagar ou seleccionar dados das tabelas – *Data Manipulation Language* (DML); e os comandos para definir os privilégios de acesso à base de dados, isto é, quais são os dados a que os diferentes utilizadores da base de dados podem aceder – *Data Control Language* (DCL). Estes últimos só podem ser executados pelos administradores da base de dados. Os comandos de criação e manipulação da estrutura (DDL), normalmente, também só devem ser executados pelos administradores de forma a prevenir eventuais dados, deliberados ou não, na base de dados.

2.6 XML

Falemos antes de mais do tipo de linguagem no qual o XML se insere. Não se trata de uma linguagem de programação mas sim de uma linguagem de anotação. As linguagens de anotação surgem com o aparecimento em 1986 da linguagem, também ela de anotação, *Standard Generalized Markup Language* – SGML [13]. O SGML apareceu devido à necessidade de encontrar um suporte para a informação digital que fosse neutro e independente de plataformas (Hardware e Software). Existia já na altura alguma diversidade de formatos para documentos digitais o que tornava dispendiosa a sua transferência e manipulação, entre diferentes plataformas. Alguma preocupação com a longevidade dos documentos começou de facto a surgir.

No final dos anos 80 Tim Berners-Lee dá origem ao HTML, uma derivação do SGML. Ao

contrário do SGML, que detinha grande complexidade, o HTML possui uma característica importante que é a simplicidade. Através da junção entre o poder do SGML e a simplicidade do HTML, surge o XML, desenvolvido pelo *XML Working Group*, conhecido originalmente por SGML *Editorial Review Board*. Digamos que as pessoas que estavam directamente ligadas ao SGML vieram posteriormente desenvolver o XML sob a alçada do *World Wide Web Consortium* – W3C [8]. O XML surge então em 1996 e no seu desenvolvimento estiveram presentes uma série de objectivos iniciais. Passamos a citar esses objectivos originais [14]: *"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"*.

Foi com estes objectivos que a linguagem XML cresceu e cresceu também a sua utilização. As características da linguagem revelam-se adequadas para a preservação a longo-prazo, dado que esta ao assentar nas premissas acima citadas demonstra ser uma linguagem aberta. Sabemos que hoje em dia, e cada vez mais, se utiliza a linguagem de anotação XML para proporcionar, entre outras coisas, interoperabilidade entre plataformas. A sua natureza de linguagem de anotação genérica assim como a sua neutralidade oferecem imensas vantagens. Também é importante referir que o XML não é propriamente uma linguagem mas sim uma metalinguagem, ou seja, uma linguagem com a qual podemos definir novas linguagens. Actualmente surgem constantes linguagens definidas a partir do XML – XHTML, SMDL, CML, etc. No nosso protótipo iremos usar exactamente uma linguagem XML, o DBML. O DBML surge como sendo uma linguagem para acomodar bases de dados e será a base do nosso repositório. Dedicaremos o capítulo 5 ao estudo e análise deste dialecto derivado do XML.

Realçando agora aspectos mais técnicos da linguagem salientamos a estrutura em árvore da linguagem e a sua constituição baseada em elementos, as chamadas anotações ou etiquetas.

Cada elemento pode ter um ou mais filhos e esses elementos filho também poderão eles próprios ter filhos. Não existe limite para o número de filhos pelo que a sua complexidade aumenta de forma proporcional à dimensão da árvore documental. Os elementos são identificados pelos caracteres reservados da linguagem '<' e '>'. Os elementos podem ainda conter atributos. Estes atributos visam qualificar os elementos aos quais estão associados.

No início do ficheiro XML deve existir a declaração XML que no fundo é uma instrução de processamento. Nos documentos XML podem ainda aparecer comentários. Em seguida temos um exemplo de um documento em XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<poema tipo="soneto">
  <titulo>Soneto Já Antigo</titulo>
  <autor>Álvaro de Campos</autor>
  <data>1922</data>
  <corpo>
    <quadra>
      <verso>Olha, <nome>Daisy</nome>: quando eu morrer tu hás-de</verso>
      <verso>dizer aos meus amigos aí de <lugar>Londres</lugar>,</verso>
      <verso>embora não o sintas, que tu escondes</verso>
      <verso>a grande dor da minha morte. Irás de</verso>
    </quadra>
    <quadra>
      <verso><lugar>Londres</lugar> p'ra <lugar>Iorque</lugar>, onde nasceste
(dizes ...</verso>
      <verso>que eu nada que tu digas acredito),</verso>
      <verso>contar àquele pobre rapazito</verso>
      <verso>que me deu tantas horas felizes,</verso>
    </quadra>
    <terno>
      <verso>embora não o saibas, que morri ...</verso>
      <verso>Mesmo ele, a quem eu tanto julguei amar,</verso>
      <verso>nada se importará... Depois vai dar</verso>
    </terno>
    <terno>
      <verso>a notícia a essa estranha <nome>Cecily</nome></verso>
      <verso>que acreditava que eu seria grande...</verso>
      <verso>Raios partam a vida e quem lá ande!</verso>
    </terno>
  </corpo>
</poema>
```

Bloco de Código 1-Poema Anotado,Retirado do Livro "XML & XSL - Da Teoria à Prática"[13]

Existem algumas regras que ditam se um documentos está bem formado ou não, uma delas é por exemplo a não possibilidade de cruzamento de anotações: *<A>olá<B>tudo bem</A>...*
Portanto numa primeira fase o documento terá que respeitar as regras da boa formação para que se pode dizer que está bem formado.

Numa segunda fase os documentos XML pode estar associados a um DTD ou a um *XMLSchema*, sendo esta a forma de os validar. Primeiro apareceram e continuam a ser utilizados, os DTDs mais recentemente, apareceram os *XMLSchemas* que têm a vantagem de ele próprios também serem escritos em XML. Todavia o objectivo quer dos DTDs quer dos *XMLSchemas* é o de validar documentos XML. Esta validação prende-se com a verificação do nome dos elementos e atributos assim como a verificação da ordem que estes ocupam no documento XML. Ora, se o documento XML respeitar o que estiver definido para a sua constituição este é considerado válido.

O dialecto que iremos utilizar para trabalhar com as bases de dados será, como já mencionamos, o DBML. Será através de um DTD ou de um *XMLSchema* que teremos a possibilidade de validar um qualquer documento XML, como sendo um documento DBML válido.

Não podemos terminar esta secção sem fazer referência ao XSL. O XSL compreende três normas, XPath, XSLT e XSLTFO. A norma XPath permite localizar e seleccionar informação em documentos XML. A norma XSLT define métodos para transformar documentos XML em outros formatos. Por fim a norma XSLFO é usada para especificar a forma pretendida para os documentos XML [13].

2.7 Linguagem de Programação – PHP

A linguagem base para o desenvolvimento do projecto foi o PHP. Esta é uma linguagem vocacionada para a programação *Web*, e amplamente divulgada. A intenção de utilizar uma plataforma *Web* para dar suporte ao nosso repositório encaminhou-nos no sentido de usar uma linguagem deste tipo. A experiência de programação, da nossa parte, com o PHP foi determinante na escolha. De referir também que, após uma breve análise, verificou-se que esta linguagem possuía o requisitos necessários para operar com as diversas tecnologias necessárias ao desenvolvimentos do sistema. Caracterizaremos em seguida a linguagem de programação PHP e as extensões usadas na implementação – *SimpleXML* e DOM –, e também o paradigma da programação orientada ao objecto.

2.7.1 PHP

Comecemos por focar algumas ideias e conceitos base do PHP de forma a enquadrarmos a linguagem. "*PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML*". De facto hoje em dia o PHP é amplamente utilizado como linguagem de programação para a Internet. O seu contributo pode ser encontrados em vastíssimos sitíos na Internet, portais, serviços *Web*, etc.

O PHP teve uma longa história que ao mesmo tempo foi curta, isto é, por um lado passou por várias fases de desenvolvimento até chegar à versão que hoje utilizamos, mas por outro lado isso aconteceu num relativo curto espaço de tempo. Foi em 1995 que Rasmus Lerdorf, por iniciativa própria e usando *scripts* em Perl, criou o PHP/FI antecessor do PHP actual. Em 1997 Andi Gutmans e Zeev Suraski rescreveram o *parser*[1] e em cooperação com Rasmus Lerdorf apresentaram o sucessor do PHP/FI, o PHP3.

1 Um Parser ou analisador sintáctico é responsável por verificar a correcta sequência dos blocos (tokens) de um programa

Depois de mais desenvolvimentos e melhorias foi criado um novo motor – *Zend Engine* – com suporte para novas funcionalidades. Algumas das funcionalidades passavam pelo suporte de sessões HTTP, suporte de mais servidores HTTP e também formas mais seguras de lidar com dados enviados pelos utilizadores. É então oficialmente lançado o PHP 4 em Maio de 2000.

Entretanto foi lançado o PHP 5 em Julho de 2004, tendo já por base tecnológica o *Zend Engine 2*, trazendo consigo um novo modelo Orientado aos Objectos (OO). Apesar de ser possível programar em PHP segundo o paradigma OO desde o PHP 3, existiam algumas limitações nos modelos de então. O PHP 5 oferece aos programadores uma forma de programar com todas as potencialidades associadas à programação OO.

Relativamente ao PHP urge salientar que é uma linguagem de *scripting*, *open-source*, que corre do lado do servidor *(server-side)* e que permite a construção de páginas *Web* dinâmicas. Esta linguagem aparece embebida no HTML e é interpretada (não compilada) pelo servidor HTTP, servidor previamente instalado com suporte PHP – módulo PHP. A linguagem permite o tratamento de formulários e ligações a diferentes bases de dados (através de ODBC ou não).

Em termos da sua sintaxe esta assemelha-se à linguagem de programação C e ao Perl, possuindo as características base de manipulação de variáveis (inteiros, *strings*, *arrays*, etc), operações aritméticas, estruturas de controlo de fluxo, ciclos, funções, etc. Como já foi dito o PHP também possui o suporte para programação orientada aos objectos o que dá aos programadores a oportunidade de criar as suas próprias classes e/ou utilizar classes já existentes criadas por outros. Existe também um conjunto de bibliotecas que disponibilizam uma série de extensões com as mais variadas funcionalidades, desde gerar documentos em PDF, criação de gráficos, manipulação de documentos em XML, etc. Em concreto para este trabalho foram criadas algumas classes para, por exemplo, extrair a informação relevante das bases de dados, gerar o respectivo DBML ou gerar SQL a partir do DBML.

2.7.2 O.O.

O paradigma da programação Orientada aos Objectos surge como consequência da evolução no campo das linguagens de programação. Inicialmente e no nível mais baixo de programação programou-se em linguagem máquina, mas como esta programação revelava-se muito difícil e complexa, surgiu numa segunda fase a programação em *assembly*. Através da programação em *assembly* é possível programar também a um nível muito baixo, praticamente igual à linguagem máquina, embora esta programação seja mais legível ao humano. Após esta fase nos anos 50 foram aparecendo as primeiras linguagens de programação consideradas de alto-nível, FORTRAN, LISP, COBOL. Posteriormente, entre 1960 e finais de 1970, teve lugar um período de refinamento no qual surgem novas linguagens (linguagens procedimentais também consideradas linguagens imperativas) [15]. Uma delas – Simula – foi a primeira desenhada para suportar programação orientada ao objecto. Mas foi nos anos 80 que o interesse por este paradigma de programação despoletou. Até determinados níveis, é possível afirmar que em muitas linguagens de programação se pode programar segundo este paradigma, no entanto algumas linguagens foram concebidas especificamente para que se possa atingir todas as potencialidades que a programação orientada ao objecto oferece – Java, C#, Delphi, PHP 5, etc.

O conceito de objecto é então fundamental. Em vez de construir um programa onde os seus procedimentos ou funções se encontram de forma dispersa, porque não agregar essas mesmas funções em torno de algo comum entre elas. Assim sendo pega-se no conceito de objecto, que oferece também a vantagem de se poder assemelhar a objectos do mundo real, e associam-se-lhe métodos (funções) que podem alterar o seu comportamento ou o seu estado mediante o envio de mensagens a esse mesmo objecto [16].

Os conceitos fundamentais desta abordagem baseiam-se na existência de classes, que representam um conjunto de objectos com características comuns e onde a instanciação de uma classe irá dar origem ao respectivo objecto. O objecto terá um conjunto de características próprias às quais chamamos de atributos. Deve ter também um conjunto de forma a permitir a alteração do seu comportamento e do seu estado. Quando existe a necessidade de invocar um

dos métodos do objecto podemos fazê-lo através do envio de uma mensagem a esse mesmo objecto.

Na programação orientada aos objectos encontramos também conceitos como a Herança, o Encapsulamento, a Abstracção e o Polimorfismo, caracterizadores deste paradigma de programação.

2.7.3 Manipulação de XML

Devido à especificidade do nosso repositório, a capacidade de manipulação de documentos XML é fundamental, e o PHP possui essa capacidade. Existem várias extensões no PHP para efectuar exactamente essa manipulação e tratamento dos documentos (dados) que sigam o formato XML. DOM (PHP 5), DOM XML (PHP 4), *libxml*, *SimpleXML*, *XMLParser*, *XMLReader* e *XMLWriter* são só algumas das extensões que o PHP coloca ao nosso dispor para trabalhar com XML.

Para o desenvolvimento do protótipo usamos essencialmente duas extensões: DOM e *SimpleXML*. A extensão *SimpleXML* oferece um conjunto de funções que permitem, de uma forma relativamente fácil e simplificada, manipular informação em XML. Esta é uma extensão que, por omissão vem activa e requer a versão 5 do PHP. Através desta extensão o PHP fica dotado de funções que lhe dão a capacidade de tratar o XML como um objecto capaz de ser processado. A extensão *SimpleXML* tem algumas limitações no que diz respeito à manipulação de informação em XML mas assegura vantagens que se prendem com a sua simplicidade e com o menor uso de recursos computacionais. Por outro lado a extensão DOM, sem dúvida mais poderosa, apresenta-se mais complexa mas também mais completa exigindo, por consequência, também mais recursos nomeadamente de memória.

Em função do tipo de manipulação que se pretende efectuar no XML, dever-se-á pesar estas

questões, por um lado uma forma de manipular mais simples, que exige menos requisitos computacionais mas com algumas limitações, por outro uma ferramenta mais poderosa e completa mas que exige mais recursos computacionais.

A extensão *SimpleXML* disponibiliza funções para adicionar atributos e elementos, identificar e encontrar atributos ou elementos filho, dá também a possibilidade de executar uma *query* XPath e obter, por exemplo, os *namespaces* usados no documento XML. Contudo, esta extensão fica-se por aqui e daí a nossa afirmação relativamente às suas limitações. Não deixa no entanto de ser verdade que se a manipulação pretendida no XML não passa destes níveis de interacção, será conveniente usar esta em detrimento da extensão DOM que, como havíamos referido, requer mais recursos computacionais nomeadamente ao nível da memória.

Relativamente à extensão DOM comecemos por contextualizar este modelo – *Document Object Model*[1]. O DOM é uma norma W3C que teve por base a criação de um modelo normalizado neutro e independente quer ao nível das plataformas quer ao nível das linguagens para representar, aceder e manipular formatos como o HTML ou o XML. Existem 3 níveis nos quais podemos dividir o DOM: Core DOM – um modelo normalizado para representar qualquer documento estruturado; XML DOM – um modelo normalizado para documentos XML; HTML DOM – um modelo normalizado para documentos HTML. Importa realçar, no contexto do nosso estudo, o XML DOM, uma vez que este define os objectos e as propriedades de todos os documentos XML, assim como os métodos ou formas para lhes aceder. No fundo falamos de uma norma para podermos aceder, alterar, adicionar ou apagar elementos num documento XML. Para o XML DOM todo o documento XML é constituído por nós. Um atributo é considerado um nó, um elemento é um nó, o texto é considerado um nó de texto, um comentário é um nó, e um documento completo é ele próprio considerado um nó. Esta visão encaixa numa estrutura em árvore onde podemos estabelecer um paralelismo com as relações pai e filho. Assim sendo sabemos, por exemplo, que os filhos de um mesmo nó (pai) são considerados irmãos/*sibling*. Na figura seguinte apresentamos as possíveis relações entre os nós, apenas no que diz respeito a uma parte da árvore.

1 http://www.w3.org/DOM/

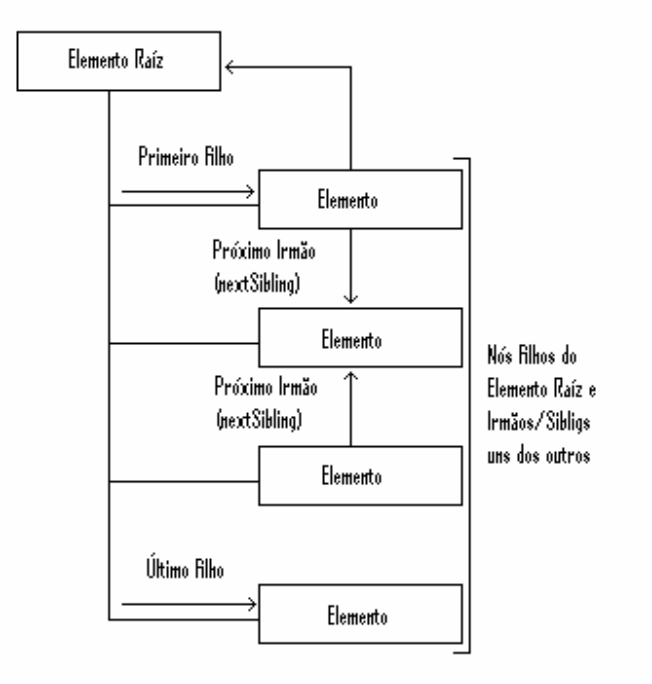

Figura 3 – Relações existentes numa parte da árvore documental (XML)

Desta forma é possível atravessar ou percorrer a estrutura/árvore do documento, sem que se saiba previamente a forma da árvore nem o tipo de dados que ele contém.
Em síntese assinalamos a forma completa como o modelo XML DOM se apresenta possibilitando as mais variadas interacções com um documento XML.

A extensão DOM do PHP assenta nestas premissas oferecendo ao programador uma API completa para lidar com documentos XML. Em concreto, esta extensão é constituída por um conjunto de classes que quando instanciadas possibilitam a manipulação, através dos métodos da classe, dos seus objectos podendo estes ser nós (classe *DOMNode*), atributos (classe *DOMAttr*), elementos (classe *DOMElement*), documentos (classe *DOMDocument*), ou outros.

2.8 MySQL

O Sistema Gestor de Bases de Dados MySQL assume um papel importante na construção do repositório devido à sua participação na fase de disseminação. Iremos ter no MySQL uma versão da base de dados preservada de forma a facilitar a disseminação dessa informação preservada através de *querys* SQL. As razões que nos levaram a optar pelo MySQL foram por um lado o seu carácter *open source* segundo os termos da licença *GNU General Public License*, e por outro a interoperabilidade que o MySQL tem com o PHP. Existe uma ferramenta de uso livre denominada de *phpMyAdmin* escrita em PHP que permite a administração do MySQL via *Web*. Isto vai de encontro aos nossos objectivos de permitir a disseminação de informação preservada através de páginas *Web*. De salientar também que o MySQL não depende de nenhum sistema operativo.

Em termos das características do SGBD este apresenta todas as funcionalidades essenciais para a administração de bases de dados: permite criar e eliminar bases de dados, criar eliminar e alterar tabelas, eliminar editar e adicionar atributos (campos) assim como as respectivas chaves, e permite também a execução de SQL. Como já referimos, podemos usar o *phpMyAdmin* na administração das bases de dados, mas essa administração pode ser também efectuada por outras interfaces gráficas (GUI) ou através da linha de comandos.
O SGBD corre como um serviço num servidor dando a possibilidade de acessos multiutilizador. O MySQL também possui comandos para definir privilégios de acesso para os possíveis utilizadores. O suporte para chaves estrangeiras ainda não se encontra disponível em todos os motores de armazenamento do MySQL mas caminhamos nesse sentido. Contudo desde a versão 3.23.44 esse suporte existe para o motor *InnoDB*, o qual foi escolhido para dar suporte à informação a ser disseminada.

2.9 Dublin Core

A norma *Dublin Core* é usada na meta-informação descritiva aquando do processo de ingestão da base de dados no repositório. Relativamente à norma evidenciamos o seu esquema de metadados que tem por objectivo fornecer informação descritiva acerca de objectos digitais (imagens, sons, bases de dados, textos, etc) de forma padronizada. O uso desta norma na informação descritiva dos objectos digitais facilita, por exemplo, a pesquisa e a partilha desses mesmos objectos.

A *Dublin Core Metadata Initiative* (DCMI) é então a organização que promove a utilização destes padrões que se destinam a descrever os recursos de informação (objectos digitais) de forma a proporcionar uma melhor percepção desses recursos. A organização DCMI define três *'I's* como sendo as suas características fundamentais: *Independent, International e Influenceable*. A norma *Dublin Core* foi definida como sendo *Standard ISO 15836* em 2003.

Normalmente as implementações *Dublin Core* usam o XML sendo disponibilizados dois níveis de padronização: Simples e Qualificado. O nível simples da norma DC comporta 15 elementos (*Title, Creator, Subject, Description, Publisher, Contributor, Data, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights*) já o nível qualificado da norma contém mais três elementos (*Audience, Provenance, RightsHolder*) e pode também conter grupos com mais elementos também chamados de qualificadores.

2.10 METS

Metadata Encoding & Transmition Standard é uma norma que tem por objectivo facultar meta-informação descritiva sobre a estrutura e organização física dos objectos digitais o que irá facilitar a exportação e intercâmbio entre repositórios dos pacotes já referidos (será usada como

envelope dos SIP e DIP. Em suma a norma METS evidencia meta-informação descritiva, administrativa e estrutural para objectos digitais. Essa meta-informação é apresentada num documento XML e detém uma determinada estrutura que iremos apresentar de seguida.

O documento METS é constituído por sete secções: *METS header*, *Descriptive Metadata*, *Administrative Metadata*, *File Section*, *Structural Map*, *Structural Links*, *Behavioral.*

2.11 Síntese

Neste capítulo procedemos à análise das várias tecnologias e normas envolvidas na construção do repositório. Salientamos que relativamente ao DBML uma análise mais exaustiva será efectuada no capítulo 5.

No próximo capítulo pretende-se abordar a temática da preservação digital, caracterizando as estratégias e consensos que actualmente vigoram.

.

3

Preservação Digital

O uso de suportes digitais para guardar informação nas instituições, quer ao nível empresarial, governamental, educacional ou outro, deve sem dúvida levar-nos à reflexão sobre a importância da preservação dessa mesma informação. A grande expansão no uso de tecnologias de informação só veio agudizar o problema [17]. Em poucos anos uma plataforma de Hardware e Software pode tornar-se obsoleta, correndo-se o risco de perder o acesso à informação lá armazenada.

Neste capítulo iremos caracterizar a visão actual sobre este problema começando por falar sobre a anatomia do objecto digital e a sua cadeia de interpretações. Estas interpretações prendem-se com os níveis em que podemos caracterizar o objecto digital – físico, lógico e conceptual. Posto isto passaremos a analisar as diferentes perspectivas relativamente à preservação digital, ou seja, passaremos em revista as várias estratégias de preservação digital. O capítulo termina com uma pequena conclusão sobre o *State of the Art* da preservação digital.

3.1 Objecto digital

Vamos antes de mais olhar um pouco para o objecto digital. Podemos fazer uma distinção entre os objectos ditos nado-digitais (aqueles que nasceram já num contexto digital) e os objectos digitalizados (oriundos de um processo de digitalização: analógico para digital). Na forma mais abrangente e que engloba ambos os casos anteriormente referidos, podemos considerar como objecto digital todo aquele que é possível representar através de um *bitstream*, ou seja, todo aquele que pode ser representado através de uma sequência de dígitos binários (zeros e uns) [4].

Em termos lógicos, o objecto digital é então caracterizado/representado por dígitos binários, contudo teremos que discutir o suporte físico sobre o qual o lógico assenta. Actualmente os suportes físicos mais comuns são o disco rígido e o CD / DVD (sabemos que também existem as *Pens/flash memory*, *tapes*, entre outros). Pegando então no exemplo do disco rígido e dos CDs / DVDs, inferimos que estes suportes físicos divergem no que concerne à tecnologia usada para guardar os códigos binários.

Neste enquadramento verificamos que a base ou estrutura física do objecto digital é fundamental até para se pensar, desde já, em possíveis estratégias de preservação. Todavia, como camada seguinte, existe a estrutura lógica ou objecto lógico; o qual corresponde à cadeia de dígitos binários. Estes detêm uma determinada disposição que irá definir o formato do objecto, dependendo do software que os irá interpretar. A interpretação por parte do software do objecto lógico irá corresponder ao aparecimento do objecto conceptual, aquele que o ser humano é capaz de entender (interpretar) e então o poder experimentar (Fig.4).

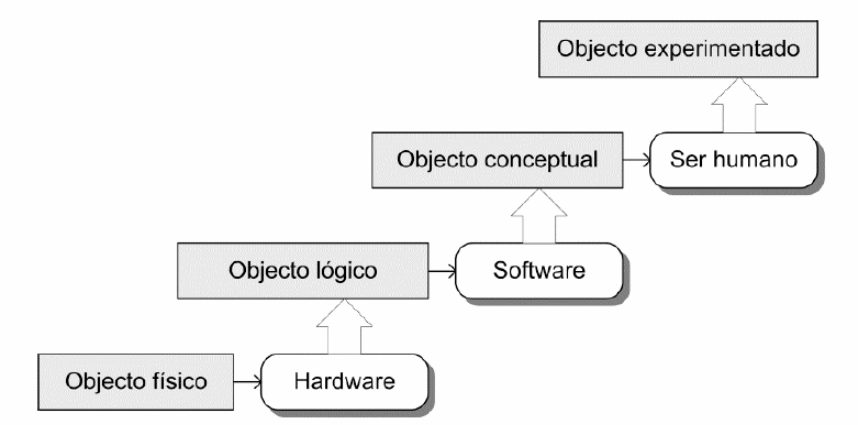

Figura 4 – Níveis de abstracção presentes num objecto digital – Retirado do Livro "Introdução à Preservação Digital" [4]

Observando toda esta cadeia de interpretações e níveis de abstracção, constatamos que a preservação digital ou a estratégia para a preservação irá de certa forma definir qual o estado do objecto a ser preservado (físico, lógico ou conceptual; o objecto experimentado depende da interpretação individual de cada ser humano, por isso não deve ser contemplado neste contexto). De certa forma, a estratégia de preservação está relacionada com o nível de abstracção considerado como importante para a preservação [3]. Numa perspectiva conceptual humana, claramente pode ser dito que o que importa preservar é o objecto conceptual, aquilo que o ser humano interpreta. Esta análise coloca em segundo plano as questões que se prendem com o que está por detrás da apresentação do objecto (suporte físico e o suporte lógico). Outras estratégias existem que defendem a preservação do *bitstream* original (objecto lógico) ou até mesmo o suporte físico original. Os vários tipos de estratégias serão analisados em seguida com mais detalhe.

O importante a reter, nesta fase em que analisamos os vários níveis de abstracção do objecto digital, é por um lado o relacionamento que estes estabelecem entre si e por outro como a existência dessa cadeia de relacionamentos é fundamental para a preservação. Quando acontecer uma quebra ou falha nessa cadeia, o objecto digital muito certamente deixará de ser inteligível, o que se pode traduzir no risco de perder o objecto para sempre [4].

3.2 Estratégias para a preservação digital

As estratégias para a preservação digital consistem nas várias abordagens reiteradas por vários investigadores, a fim de garantir que o objecto ou objectos digitais se mantenham interpretáveis ao longo dos tempos. Como já foi dito, existem preocupações diferentes no que diz respeito ao que é essencial preservar. Uns dirão que será fundamental garantir a autenticidade e então defendem que a única estratégia possível será a "Preservação de tecnologia", de forma a preservar o objecto digital na sua forma original. Outros advogam que bastará preservar o objecto conceptual, aquilo que no fundo interage com o mundo real, ou seja, aquilo que o ser humano irá experimentar. Pelo meio existem ainda outros tipos de abordagens, pelo que se deve assinalar desde já que esta não é uma problemática consensual. Tal facto deve-se essencialmente às muitas questões suscitadas e à própria experiência humana com os objectos digitais e com a problemática de preservação digital que é ainda relativamente recente. Sequentemente, abordaremos algumas estratégias de preservação digital.

3.2.1 Preservação de Tecnologia

Uma das estratégias propostas para a preservação digital intitula-se Preservação de Tecnologia. Esta abordagem procura preservar o ambiente tecnológico, tal como ele existe ou existiria. Esta estratégia passa pela conservação e manutenção de todo o hardware e software que caracterizavam e constituíam os objectos digitais originais [1].

Como previamente referido, este tipo de técnica/estratégia dá especial ênfase à preservação do objecto físico (e lógico), tendo os defensores/impulsionadores desta abordagem como justificação o facto de esta forma ser a única que fielmente poderá (de forma fidedigna)

representar/recriar o objecto digital original (autenticidade). Acontece que em termos de escala, esta estratégia revela-se muito complicada de implementar, pelo menos para todo o tipo de objectos digitais. Senão vejamos o que terá que ser feito para levar por diante tal estratégia: ter-se-ia que encontrar locais físicos e com uma determinada localização geográfica para armazenar todo o legado tecnológico (hardware e software). No fundo, acabaríamos por construir museus de tecnologia [5].

Os custos (aos mais vários níveis) envolvidos neste tipo de estratégia são a sua principal desvantagem, contudo é possível imaginar que para alguns tipos de artefactos digitais seria interessante a sua preservação segundo esta abordagem.

3.2.2 Refrescamento

Uma outra técnica usada no contexto da preservação digital é o Refrescamento. O Refrescamento consiste na passagem da informação existente num determinado suporte físico, que eventualmente possa estar a tornar-se obsoleto, para um mais actual. Como exemplo desta prática temos as cópias de informação de suportes que se estão a tornar obsoletos, como as disquetes, para suportes mais actuais, como os CDs. Podemos através desta técnica garantir que pelo menos a informação continua acessível do ponto de vista do hardware. Claro está que será necessário conjugar esta abordagem com outras estratégias, de forma a atingir uma preservação digital efectiva [18].

3.2.3 Emulação

Uma estratégia importante, tendo em conta que já se pode encontrar implementada com algum impacto no mundo real, é a Emulação. O que iremos ter sobretudo será um software capaz de emular outros softwares. Desde já, podemos constatar que este software, denominado de

emulador, poderá e virá certamente a sofrer ele próprio de obsolescência (desvantagem).

O que é então um emulador? Um software, que tenta recriar as condições tecnológicas para que uma determinada aplicação possa correr sobre ele, recriando assim o ambiente original. Esta estratégia assume principal relevância quando falamos em aplicações de software (executáveis) com aspectos dinâmicos e interactivos (por exemplo os jogos). Nestes casos assume especiais vantagens, uma vez que é capaz de atingir altos níveis de preservação no que diz respeitos às propriedades e características do objecto digital original [1].

Exemplos desta estratégia são os emuladores de consolas de jogos. Jogos antigos e que foram originalmente desenvolvidos para correr sobre um determinada consola/plataforma, podem voltar a ser executados mesmo na ausência física da consola. Isto é então possível através do uso de um emulador.

3.2.4 Migração

Outra estratégia usada para a preservação digital é a Migração. Esta estratégia consiste na conversão da informação existente num determinado formato ou numa determinada plataforma software/hardware para formatos/plataformas mais actuais. Objectivamente falando, a ideia é conseguir que a informação se mantêm sempre num estado considerado como actual e interpretável pelas tecnologias vigentes [1]. A migração é uma das estratégias para a preservação digital mais usadas actualmente e com mais provas dadas no mundo real. Não obstante, esta é uma estratégia que não poderá resolver a questão em definitivo (de tempos a tempos ter-se-á que voltar a migrar para novos sistemas tecnológicos mais actualizados).

Existem diferentes tipos de Migração, podemos ter migração para suportes analógicos que se baseia em converter os objectos para suportes analógicos. É certo que apenas se enquadram nesta abordagem os objectos digitais, cuja representação se aproxima dos formatos analógicos

como imagens, documentos textuais entre outros. O objectivo é preservar o objecto analógico, uma vez que este se encontra num estado que é inteligível ao ser humano e não necessita de intermediários para que o ser humano o possa experimentar.

A actualização de versões é uma estratégia de Migração muito utilizada senão a mais utilizada. Consiste essencialmente em importar os objectos digitais para versões mais actualizadas, visando manter a actualidade dos mesmos [3]. No entanto, a actualização de versões pode ter em si alguns problemas, uma vez que estamos dependentes da empresa que detém determinado produto. Sabemos que muitas vezes quando se passa de uma versão para outra mais actual nem sempre se garante que todos os aspectos e atributos de um determinado documento são incorporados na nova versão. Perante tais descontinuidades nos softwares é necessário por vezes transportar os documentos para formatos concorrentes que não são dependentes de qualquer fabricante.

3.2.5 Normalização

A Normalização pretende encontrar formatos que sejam amplamente conhecidos e com normas internacionais abertas. Deste modo a pode-se então migrar os objectos para esses formatos, a fim de que as estratégias de preservação se preocupem com um reduzido número de tipos de formatos a preservar. Se ao ingerir num repositório, todos os objectos (por exemplo imagens) forem convertidos para um formato único, reduziremos em muito o custo de preservação, visto que em vez de ser necessário preservar um vasto número de formatos, a preservação irá centrar-se apenas num determinado formato [19]. Por tudo isto, verificamos que a escolha do formato é muito importante (convém que agrade à comunidade de interesse e também comporte todos os aspectos relevantes de um determinado tipo de artefacto digital).

3.2.6 Migração a-Pedido

Outra abordagem em termos de migração é conhecida como Migração a-Pedido. Esta caracteriza-se por ser um tipo de migração que usa sempre o formato original para efectuar as transformações/conversões para os formatos mais actuais/pretendidos. Muitas vezes, quando temos várias migrações sobre determinado objecto digital, o que acontece é que ele pode ir perdendo características, atributos e, ao fim de uma série de migrações, pode já não representar o original de forma aceitável. A migração a-Pedido assume uma grande vantagem neste aspecto, uma vez que as migrações são sempre levadas a cabo a partir do objecto original. É importante assinalar também que uma vez construídos os conversores que descodificam o objecto original, essa informação pode ser posteriormente usada sempre que for necessária a migração.

3.2.7 Migração Distribuída

A Migração Distribuída é mais uma estratégia de migração, muito embora seja ainda relativamente recente. Podemos imaginar um conjunto de conversores existentes na Internet a funcionarem tipo *Web services* que possibilitam a conversão de diferentes formatos.

A principal potencialidade desta abordagem é a existência de diferentes conversores e rotas diversas em termos de conversão para determinados formatos. Desta forma pode-se assegurar que se por ventura algum conversor falhar ou algo se torne obsoleto temos sempre outros caminhos (*Web services*) que podemos usar para atingir o formato pretendido. Teríamos uma rede global de conversores que poderá contribuir em muito para o sucesso da preservação. Temos apenas que ter atenção à quantidade de informação tratada; se esta tiver um volume muito elevado podemos ter problemas uma vez que estamos a falar da Internet e por conseguinte da largura de banda para transferência de informação e dos seus custos inerentes.

3.2.8 Encapsulamento

Nos casos em que um objecto digital é guardado durante um período de tempo elevado sem que ninguém nem nada o altere pode surgir um situação em que a migração tenha um custo muito elevado ou até se torne completamente inviável. Para enfrentar os problemas relacionados com a preservação neste tipo de situações, existe uma estratégia que é conhecida como o Encapsulamento. Esta estratégia consiste em guardar os objectos inalterados, assim como informação (meta-informação) acerca do objecto. Desta forma, quando no futuro os artefactos forem solicitados, a meta-informação irá possibilitar a construção dos conversores necessários.

3.3 Conclusão

Verificamos que, de alguma forma, haverá ainda caminhos a desbravar no que diz respeito a esta problemática da preservação digital. Tendo em conta o estado actual das tecnologias de informação, assim como as características sob as quais o objecto digital se apresenta, são estas as estratégias consideradas válidas e utilizáveis para tentar atingir a preservação digital.

A migração é uma das estratégias que indubitavelmente assume um papel de relevância, contudo não a podemos considerar a chave do problema. Apenas através de uma visão consertada desta problemática será possível caminhar no bom sentido [20]. Em função do tipo de objectos digitais que se pretende preservar dever-se-á escolher a estratégia mais adequada [21].

Por determinação da constante revolução no mundo das tecnologias digitais é de esperar que muitos avanços sejam conseguidos nos próximos anos nesta área de investigação.

4

OAIS

Em Janeiro de 2002 um documento foi aprovado pelo *Management Council of the Consultative Committee for Space Data Systems* (CCSDS), um documento que representou o acordo levado a cabo pelos participantes do *CCSDS Member Agencies*.

O documento consiste numa série de recomendações técnicas com o objectivo de estabelecer um consenso globalizado no que diz respeito à preservação digital de informação de forma permanente, ou por longos períodos de tempo (longo prazo) – "*This document is a technical Recommendation for use in developing a broader consensus on what is required for an archive to provide permanent, or indefinite long-term, preservation of digital information*" [6]. De um modo geral podemos dizer que se procedeu à definição dos requisitos necessários de forma a obter um *Open Archival Information System* (OAIS). Vulgarmente referimo-nos ao OAIS dizendo que estamos a falar do modelo de referência OAIS, e é exactamente disso que se trata: um modelo de referência onde se estabelece uma série de recomendações, estratégias a seguir, linhas mestras a ter em conta e onde também são identificados os componentes e agentes necessários para atingir, num determinado arquivo, a preservação da informação digital.

Em traços gerais, o modelo de referência OAIS, assenta em três operações fundamentais: a ingestão da informação, a administração do arquivo e a disseminação da informação. Em seguida iremos descrever com algum detalhe os componentes de um sistema orientado segundo o modelo de referência OAIS, e também a forma como o sistema se deve comportar. Todas as características assim como todas as recomendações presentes no modelo de referência apontam no sentido de uma preservação digital da informação, contudo não se limitam apenas à preservação simples mas também à obrigatoriedade da informação permanecer acessível e inteligível por longos períodos de tempo.

O propósito do modelo de referência OAIS é o de definir um *standard* ISO[1] – *International Organization for Standardization* – que aponte no sentido de preservar informação digital por longos períodos de tempo, podendo estes ser indefinidos. Isto implica a ocorrência de mudanças ao nível das tecnologias envolvidas, dos formatos dos documentos ou até da comunidade de utilizadores (interesse). O modelo em si passa por um conjunto ou organização de pessoas e sistemas que aceitam a responsabilidade de manter a informação digital "sempre" disponível para a sua comunidade de interesse. O modelo diz-se Aberto (*Open* A. I. S.) pois as recomendações para o seu desenvolvimento devem ser mantidas em fóruns abertos e disponíveis a todos.

O modelo de referência OAIS coloca ao nosso dispor as ferramentas necessárias para o entendimento e sensibilização de conceitos relacionados com arquivos digitais e preservação a longo-prazo. O modelo também oferece os conceitos necessários para que as organizações, que nada têm a ver com a preservação, possam participar de forma activa neste processo. O modelo oferece ainda mais conceitos para uso futuro no que diz respeito à comparação dos sistemas e das suas arquitecturas. Questões relacionadas com estratégias, técnicas de preservação digital e alterações que ao longo do tempo os modelos adoptados para preservação podem sofrer são também abordadas por este modelo de referência.

1 http://www.iso.org/

O modelo também disponibiliza uma base para tratar questões relacionadas com a preservação de informação que não se encontre em formatos digitas. O modelo refere alguns consensos relacionados com todo o processo da preservação que possibilitarão ao nível do mercado um aumento deste tipo de produtos. Este modelo de referência participa também na identificação e na orientação da produção de modelos relacionados – *OAIS-related Standards*.

Como iremos ver mais à frente este modelo de referência preocupa-se com uma série de questões inerentes à preservação digital: o processo de ingestão de informação no sistema, o armazenamento da informação assim como a sua administração e preservação, e finalmente com o acesso e disseminação da informação. Questões relacionadas com a troca de informação entre arquivos também são abordadas pelo modelo assim como o papel que o software assume na problemática da preservação digital.

4.1 O modelo

O modelo OAIS, por definição, pode ser aplicado a qualquer tipo de arquivo, no entanto ele é mais direccionado para organizações que necessitem de ver a sua informação preservada por longos períodos de tempo. O modelo também irá ter interesse para aqueles que pretendam extrair informação de arquivos orientados segundo o mesmo modelo (OAIS).

O rápido crescimento no mundo da informática e das comunicações levou a um aumento nas transacções digitais entre empresas/organizações. A informação segue agora cada vez mais caminhos e meios digitais em detrimento dos meios mais tradicionais como o papel.

As próprias organizações entraram ou foram levadas a entrar neste novo paradigma de funcionamento sem prestarem atenção ao problema que estavam a criar. Possivelmente muitas ou pelo menos algumas das organizações nunca pensaram na possibilidade de enfrentarem esta

questão de preservação. O modelo OAIS vem definir um conjunto de requisitos e recomendações que as organizações devem seguir para amenizar ou combater esta problemática da preservação da informação digital por longos períodos de tempo (a longo-prazo ou de forma permanente).

Todavia o modelo OAIS não refere nem especifica plataformas computacionais, linguagens de desenvolvimento, sistemas gestores de bases de dados, interfaces, enfim, não condiciona o desenvolvimento do sistema ao nível da tecnologia envolvida. O modelo servirá então como um guia para quem pretender desenvolver um arquivo digital [6].

4.2 Conceitos OAIS

As organizações começam a tomar consciência da necessidade de implementar políticas de preservação da informação digital. Sabe-se que a informação digital pode muito facilmente perder-se ou ser corrompida. No momento da produção da informação digital temos um acesso privilegiado aos dados sobre como essa informação é produzida (metadados). Ora se as organizações tiverem um papel activo relativamente à preservação irão mais tarde recolher benefícios.

Devido a vários factores, alguns deles já referidos, a produção de informação digital aumenta a cada dia, o que faz com que em muitos casos o papel do produtor da informação e do arquivador da mesma informação se funda. As questões, que a preservação digital levanta e impõe, são muitas vezes relegadas para segundo plano, acarretando custos que só mais tarde se sentiram. Quem constrói e desenha os sistemas deve ter em atenção a extrema importância em documentar todo tipo de informação gerada; contudo facilmente nos apercebemos que isto muitas das vezes colide com objectivos de mercado de uma rápida produção e disseminação de produtos para os seus consumidores [6].

O modelo de referência OAIS surge então para facultar um conhecimento e uma percepção do efectivamente necessário de forma a se obter ou poder atingir a preservação digital (por longos períodos de tempo). Este modelo encaminha-nos no sentido da criação de um sistema de arquivo de informação aberto, assim como no sentido de criar as funções necessárias para aceder a essa mesma informação [6,22].

Revela-se fundamental o acesso à informação pela respectiva comunidade de interesse; um sistema OAIS assenta de forma sólida sobre esta premissa.

O sistema pode ser actualizado com informação de forma regular ou não; por outro lado o sistema poderá também ter que dar respostas mais ou menos complexas dependendo dos casos. É um dos objectivos, não de um sistema OAIS em si mas do modelo de referência OAIS, proporcionar termos e conceitos que ajudem as organizações a lidarem com a preservação.

4.3 Ambiente OAIS

Um sistema de arquivo OAIS, visto do exterior, é constituído pelos responsáveis pela produção de informação a ingerir no arquivo OAIS (produtores), pelos responsáveis pela manutenção/gestão/administração da informação existente no sistema (gestores) e, finalmente, por responsáveis pelo consumo da informação (consumidores) ou seja aqueles que extraem informação do sistema/arquivo.

O produtor tem por função fornecer informação ao sistema com o objectivo da preservação da mesma. Os responsáveis pela manutenção da informação no sistema levarão a cabo políticas e actividades no sentido de promover a preservação. O consumidor por sua vez interage com o sistema no sentido de extrair informação de interesse. A comunidade de interesse é um

determinado grupo de consumidores capaz de compreender a informação preservada.

Existirão casos em que o sistema OAIS não se apresenta desta forma tão explícita; podemos ter relacionamentos a vários níveis entre arquivos/sistemas. Um determinado OAIS pode assumir o papel de produtor para outro OAIS, por exemplo, quando a responsabilidade de preservar determinado tipo de informação é passar essa informação para o outro OAIS. O contrário também se pode verificar quando um OAIS assume o papel de consumidor entendendo que um tipo de informação pode perfeitamente ficar arquivada num outro OAIS sendo esta consumida quando necessário. Estas interacções entre arquivos deverão assentar em bases formais que garantam as comunicações mesmo quando estas sofram alterações nas especificações.

4.4 A Informação no OAIS

Antes de mais é fulcral perceber o que é informação. Um OAIS terá que identificar e caracterizar a informação que se propõe preservar. A informação só é útil se puder ser entendida, ora para entender a informação arquivada no sistema é necessário possuir uma Base de Conhecimento. Esta Base de Conhecimento irá permitir entender e perceber a informação que está arquivada; se não possuímos esta Base, e como falamos em preservação por longos períodos de tempo isso pode perfeitamente vir a verificar-se, será necessário criar uma "Representação da Informação" (*Representation Information*) que irá funcionar como uma espécie de manual para entender o que está arquivado – um dicionário para entender uma nova língua, por exemplo. O sistema deve preocupar-se com a preservação do objecto de dados (*data object*) e também com o dito manual para entender esses mesmos dados; desta forma será possível obter o Objecto de Informação (*Information Object*). A partir do Objecto de Informação deverá ser possível extrair conhecimento.

Uma complicação que advém desta visão é que a "Representação da Informação" poderá ser

recursiva; isto normalmente leva à constituição de uma rede deste tipo de objectos. No caso da informação digital é necessário uma Representação que dê a possibilidade ao sistema de identificar e perceber os bits fazendo com que possamos vir a extrair informação inteligível.

Voltando à Comunidade de Interesse, o OAIS deverá estar a par da Base de Conhecimento desta mesma comunidade para que possa definir a Representação que será necessária arquivar junto com os dados. A decisão dessa Representação ser maior ou menor irá ter impacto na comunidade que poderá vir a entender o que está arquivado. Poderão haver actualizações à Representação para que o que está arquivado permaneça inteligível.

Sabemos que, na prática, usamos um software para aceder ao Objecto Informação, mas é necessário ter algum cuidado para que o software não tome um papel fundamental pois tornar-se-ia mais complicado preservar o software do que a própria informação.

Algumas questões que se prendem com a forma como a informação é apresentada, podem ser também pertinentes muito embora o modelo de referência OAIS se preocupe mais com a informação propriamente dita e não com a forma como esta se apresenta. Claramente se observa que ao atender a estas questões iríamos tornar o sistema mais complexo. Em certos casos não seria mesmo possível satisfazer este requisito devido, por exemplo, aos softwares proprietários. As técnicas ou estratégias para esta preservação do ambiente original são abordadas no capítulo “Preservação Digital”.

Falemos agora da definição de Pacote de Informação (*Package Information*).

4.5 Pacotes de Informação

Cada transmissão de informação num OAIS quer seja nos processos de submissão ou

disseminação ocorrerá como sendo uma operação ou um conjunto de operações discretas. Surge então o conceito de Pacote de Informação, associado a essas transmissões.

Conceptualmente um Pacote de Informação é constituído por dois tipos de informação: o Conteúdo de Informação – o objecto (Objecto de Informação incluindo a *Representation Information*) que originalmente se pretende preservar – e a Informação Descritiva de Preservação (PDI) do inglês *Preservation Description Information*. O Pacote de Informação irá encapsular estes dois tipos de informação e será visto e detectável em consequência de uma Informação Descritiva (aqui referente a todo o Pacote de Informação).

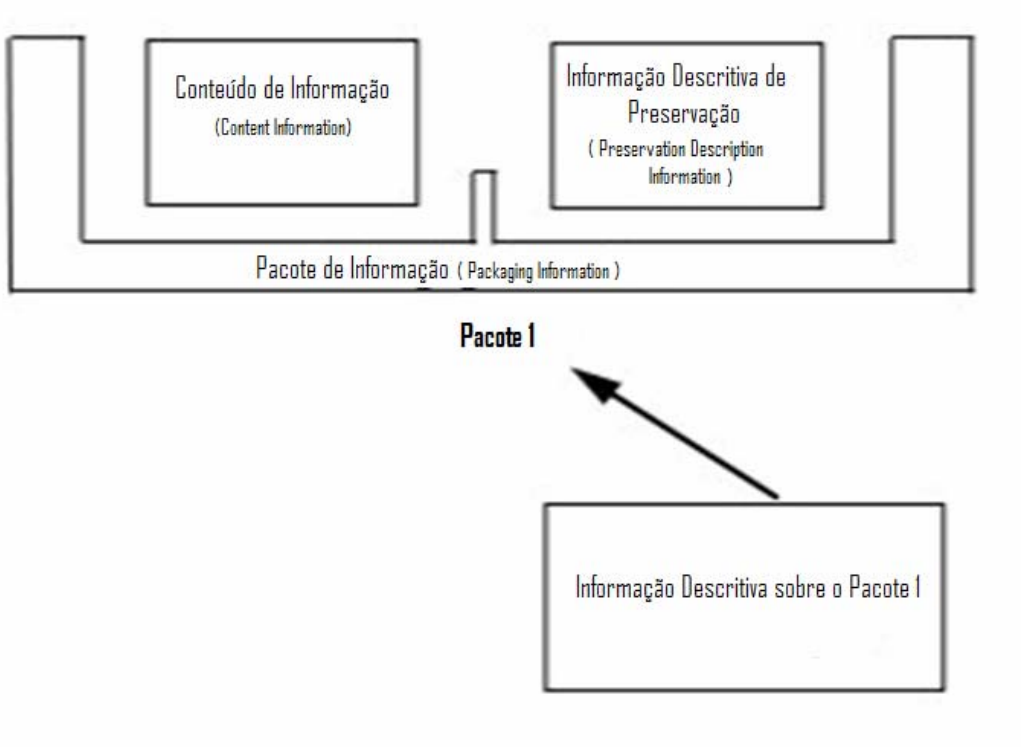

Figura 5 – Conceito e relações existentes num Pacote de Informação

A Informação Descritiva de Preservação tem que existir para identificar o Conteúdo de Informação e o ambiente em que esta foi criada. A Informação Descritiva de Preservação irá ser

dividida em quatro partes: Proveniência, Contexto, Referência e "Protecção" (*Fixity*). A Proveniência dar-nos-á a origem ou fonte do Conteúdo da Informação assim a sua história e percurso. O Contexto irá fornecer informação sobre como o Conteúdo da Informação se relaciona com possíveis informações exteriores ao Pacote de Informação; poder-nos-á dar também, por exemplo, o porquê da criação daquele Conteúdo de Informação. A Referência identificará de forma única Conteúdo da Informação. A Protecção funciona como um escudo protector para que o Conteúdo da Informação não seja alterado sem que se proceda à devida documentação da acção.

A informação associada ao Pacote de Informação (*Packaging Information*) terá que de alguma forma relacionar o Conteúdo da Informação com o PDI. Por sua vez a Informação Descritiva irá ajudar nas pesquisas e na descoberta de determinados Conteúdos de Informação.

Os Pacotes de Informação poderão assumir três diferentes formas ou três variações: Pacote de Informação para Submissão – SIP do inglês *Submission Information Package*; Pacotes de Informação para Arquivo – AIP do inglês *Archival Information Package*; e Pacotes de Informação para Disseminação – DIP do inglês *Dissemination Information Package*. Estas três variações vão de encontro às características específicas de cada momento no processo de preservação de informação. O Pacote de Informação submetido poderá não respeitar todos os requisitos para um arquivo correcto tendo em vista a preservação; assim como a um Consumidor pode ser entregue um pacote que não contenha toda a *Representation Information* ou toda a PDI.

Entre o Produtor e o Sistema serão negociadas de forma detalhada a composição do pacote que é submetido. Poderão ser necessários vários SIPs para formar um AIP, assim como a informação contida num SIP poderá constar em vários AIPs.

Um AIP será o pacote que efectivamente será preservado "dentro" do OAIS, isto é no interior do repositório. Este tipo de pacotes deverão respeitar os standards de um sistema OAIS; deverão, por exemplo, conter um conjunto de PDI associada ao Conteúdo de Informação. Um AIP pode conter conjuntos de outros AIPs.

A forma como se procede em resposta a um pedido de informação ao sistema é através de um DIP. Este pacote pode ser constituído por vários AIPs e pode ou não conter a Informação Descritiva de Preservação completa. Nesta fase da disseminação será então entregue ao Consumidor, satisfazendo os seus requisitos, um Pacote de Informação (DIP). Por exemplo, uma imagem pode ser disseminada no formato em que se encontra arquivada ou noutro formato mais apropriado às necessidades do consumidor. Se a ideia é disponibilizar a imagem num *browser* esta poderá possuir uma resolução mais baixa atendendo às necessidades.

4.6 Interacções

Iremos agora falar sobre as interacções entre os agentes ou entidades no sistema e sobre o fluxo da informação nas várias operações de alto nível. Como vimos o Produtor entrega ao OAIS um SIP; o OAIS haverá de transformar os SIPs em AIPs que são os pacotes efectivamente armazenados; finalmente um Consumidor irá efectuar pedidos ao sistema que irá responder sob a forma de DIPs (um repositório poderá ter vários disseminadores para cada tipo de objecto armazenado).

No que diz respeito à Administração do Sistema, esta é responsável por definir âmbitos, condições e a própria constituição do OAIS. A Administração irá abranger todas as actividades sobre o arquivo (OAIS). A Administração determinará também os grupos de Produtores e Consumidores abrangidos pelo sistema.

A nível administrativo, algumas actividades concretas da Administração poderão ser: financiamento do projecto, definição de linhas a seguir, gestão de recursos, gestão de conflitos, revisão de objectivos, etc. A Administração também se deverá preocupar em encaminhar para a o seguimento dos standards e utilização do OAIS, toda a esfera envolvente ao projecto.

A um nível mais técnico a Administração do sistema preocupa-se em monitorizar o estado de obsolescência dos objectos armazenados, efectuando auditorias ao sistema e despoletando acções de migração (estratégia de preservação adoptada) sempre que necessário.

Entre o Produtor e o OAIS terá que existir uma relação, na qual o principal requisito estabelece que o sistema (OAIS) deverá preservar a produção de dados do primeiro. Um Contrato de Submissão de Dados ou Compromisso é definido – *Submission Agreement*. No Contrato de Submissão uma ou mais Sessões de Submissão de Dados (*Data Submission Sessions*) são estabelecidas. A sessão poderá ocorrer de tempos a tempos e poderá conter um ou mais SIPs. Uma negociação entre o Produtor e o OAIS irá estabelecer um modelo que define o conteúdo das Sessões de Submissão de Dados: o Conteúdo de Informação, PDI, Informação de Pacote, Informação Descritiva, tudo isto será identificado através do modelo estabelecido. Através do Contrato de Submissão é definida também a periodicidade das Sessões.

Cada Pacote de Informação submetido, isto é cada SIP, deverá corresponder aos requisitos do sistema; no entanto como já foi dito, casos existirão em que serão necessários vários SIPs para constituir um AIP; também poderemos ter casos em que a informação contida em determinado SIP poderá constar em vários AIPs. Os protocolos de verificação, por parte do OAIS, de conformidade das Sessões de Submissão de dados também serão definidos pelo Contrato de Submissão de Dados.

O modelo de referência OAIS advoga que deve existir uma série de interacções entre o sistema e o Consumidor no sentido de satisfazer as necessidades de informação do segundo. Interacções essas, que poderão surgir sob a forma de questões lançadas sobre o sistema, buscas na literatura, pesquisas no catálogo, pedidos e consulta do estado dos pedidos. À semelhança da interacção que existe entre o Produtor e o Sistema, também será definido entre o Sistema e o Consumidor um Contrato de Pedido de Dados – *Order Agreement*. É este contrato ou acordo que nos indicará a periodicidade das Sessões de Disseminação de Dados. O Contrato de Pedido Dados é que irá identificar os AIPs de interesse e como estes serão transformados em Pacotes de

Disseminação de Dados (DIP), de forma a integrarem a Sessão de Disseminação de Dados. Dois tipos de pedidos estão à disposição do Consumidor: *Event Based Order* e *Adhoc Order*.

No caso de uma *Adhoc Order*, o Contrato de Pedido assenta sobre informação supostamente existente no sistema. Pode existir, por parte do Consumidor, a necessidade de Pesquisar no arquivo pela informação de interesse. Será então estabelecida uma Sessão de Pesquisa que poderá basear-se nas Informações Descritivas ou mesmo em informações que se encontrem dentro dos próprios AIPs. As pesquisas poderão ser interactivas. Uma vez encontrada a informação desejada, isto é, os AIPs de interesse, o Contrato de Pedido pode então ser definido. O Contrato irá identificar os AIPs de interesse e como posteriormente os DIPs serão adquiridos do sistema. Caso se verifiquem todas as premissas anteriores (a informação desejada encontra-se no sistema) iremos ter então uma *Adhoc Order*. Caso os AIPs de interesse não se encontrarem ainda no OAIS será implementada uma *Event Based Order*.

Os *Event Based Order* irão estabelecer também um Contrato de Pedido mas para informação que se prevê que seja recebida pelo sistema e que despoletará algum evento. O evento poderá ter alguma periodicidade ou ser apenas um evento único despoletado por alguma informação particular inserida no sistema. O Contrato de Pedido irá também ter que dizer ao sistema quando este deverá accionar uma nova Sessão de Disseminação de Dados baseada em determinado evento. O Contrato de Pedido comportará os critérios para a selecção da informação a ser incluída na Sessão de Disseminação de Dados. Um Contrato de Pedido poderá basear-se num formulário a ser preenchido na Internet que especifique os AIPs de Interesse.

4.7 Responsabilidades

Entre as responsabilidades que um OAIS deve assumir, encontram-se a negociação com os Produtores no sentido de aceitar os seus Pacotes de Informação. O sistema tem que obter um

controlo que garanta a preservação a longo-prazo desses Pacotes de Informação ingeridos. O sistema deverá ser capaz de determinar ou saber qual a comunidade ou comunidades de interesse que necessitam de entender a informação preservada. A informação preservada deverá ser entendida de forma independente para a comunidade de interesse. O sistema deverá também levar a cabo políticas e procedimentos, documentados, que assegurem que a informação será preservada em qualquer tipo de situação ou acontecimentos não planeados, ou seja, contra qualquer tipo de contingência. Em suma a informação preservada deverá estar disponível e inteligível à comunidade de interesse [6,23].

5

DBML

Sendo as bases de dados relacionais o nosso objecto do estudo para a preservação digital, foi premente a necessidade de encontrar um formato de preservação. Será este o formato que iremos utilizar para representar a base de dados e posterior armazenamento no repositório. Como anteriormente referimos a escolha recaiu sobre uma linguagem definida a partir do XML, o DBML [9]. É então neste capítulo que iremos proceder a uma análise cuidada do dialecto DBML.

Antes de falar concretamente sobre o DBML e sobre a sua estrutura – DTD/*XMLSchema* – vamos analisar o que é uma base de dados em termos genéricos. Voltamos nesta fase a fazer alguma alusão ao modelo relacional visto que as bases de dados que nos propomos preservar seguem este modelo. Posteriormente e tendo em conta a análise feita ao tópico bases de dados relacionais e suas propriedades significativas a preservar vamos olhar para o DBML.

5.1 Bases de dados

O que é uma bases de dados? Uma base de dados poderá ser definida como sendo um conjunto de informação que se encontra de alguma forma estruturado? No domínio da informática, a base de dados assenta num determinado programa ou software, normalmente chamado de Sistema Gestor de Bases de Dados SGBD, que trata do armazenamento e administração da informação. Uma base de dados implica essencialmente a existência de um conjunto de registos de informação; informação essa que levará de alguma forma à obtenção de conhecimento.

A forma ou estrutura de relacionamentos e relações entre os objectos no interior de uma base de dados pode variar dependendo antes de mais do tipo de modelo de BD usado. Como já mencionamos o nosso estudo centra-se no modelo relacional, amplamente divulgado e certamente o mais usado. Contudo existem outros modelos lógicos para bases de dados: o *Flat model*, o *Hierarchical model*, *Object database model* e até o *Post-relational database model*, entre outros [26].

Com o objectivo de encontrar ou definir quais as propriedades significativas do ponto de vista da preservação digital temos que começar por identificar todas as propriedades e/ou aspectos relevantes de uma base de dados. Como metodologia adoptada para proceder à identificação de *todas* as propriedades, iremos partir do que nos parece mais genérico caminhando para aquilo que será mais específico.

Sem dúvida que o modelo (relacional) usado no armazenamento e relacionamento da informação aparece logo à cabeça e acaba por definir uma série de pressupostos que se prendem com o que virá a seguir. Uma vez estabelecido o modelo em causa, pode ser importante, apesar deste tipo de modelos se encontrarem bem documentados na literatura, guardar, além da descrição do modelo, as características que o mesmo comporta assim como toda a informação contextual importante para que de futuro se possa entender o modelo (situação extrema em que

se perdeu também a própria informação acerca do modelo).

Através deste conceito de base de dados e da análise prévia ao modelo relacional (Cap. 2) é possível partir para uma fase onde especificamos todas as propriedades consideradas significativas para a preservação.

5.2 Propriedades significativas

Falando então dos aspectos significativos a preservar numa base de dados verificamos que, além das características do modelo (relacional) no qual a base de dados assenta, existem outros aspectos relevantes. As informações acerca do ambiente original da base de dados também devem ser preservadas.

- Sistema Operativo
- Sistema Gestor de Bases de Dados (SGBD)
- Data de criação da Base de Dados
- Identificação do criador

Portanto informações que indiquem qual o sistema operativo e o qual o SGBD que dão suporte à base de dados original são sem dúvida importantes, para caracterizarem o ambiente original da base de dados. A data de criação da base de dados e a identificação do seu criador são dados que iremos também preservar. Estas últimas informações em conjunto com o sistema operativo e o SGBD, que mencionamos antes, são os dados que identificamos como sendo meta-informação técnica que diz respeito à base de dados original.

Iremos também referenciar no documento DBML de preservação da bases de dados a data na qual aconteceu o evento de preservação. Também se poderá incluir mais alguma informação,

como por exemplo o agente que procedeu à preservação.

Em seguida usamos um pequeno excerto de um caso de estudo para demonstrar a organização e o conteúdo de um documento DBML.

```
<DB NAME="Inqueritos" SGBD="SQLServer" SO="Win2003"  DATAC="2007-05-11"
CRIADOR="rfreitas" DATAP="2008-04-15" …>
```

Bloco de Código 2 – Atributos para o Elemento Raiz (DB)

A informação existente na base de dados possui uma determinada estrutura baseada, como vimos anteriormente, em relações. Essa estrutura faz com que os dados façam sentido, isto é, a estrutura proporciona uma forma de interpretação dos dados para que se possa extrair informação válida – conhecimento. Temos então por um lado os dados armazenados na base de dados e por outro lado a estrutura da mesma. Através desta análise facilmente chegamos à conclusão que será necessário preservar não só os dados mas também a estrutura da base de dados. A seguir apresentamos a estrutura final para o documento DBML:

```
<DB NAME="Inqueritos" SGBD="SQLServer" SO="Win2003"  DATAC="2007-05-11"
CRIADOR="rfreitas" DATAP="2008-04-15" …>
        <STRUCTURE>
                ...
        </STRUCTURE>
        <DATA>
                ...
        </DATA>
</DB>
```

Bloco de Código 3 – Elementos Principais - Structure e Data

Começando pela estrutura, vimos que a base de dados é constituída por tabelas, assim sendo

preservamos cada uma das tabelas. Temos então que definir a estrutura para cada uma das tabelas identificando todos os seus atributos. Para cada um dos atributos identificamos o seu domínio/tipo de dados, o tamanho que o atributo pode assumir (tamanho do tipo de dados) e também se o atributo pode ou não ser nulo. Segue-se a estrutura em DBML para caracterizar os atributos do nosso caso de estudo.

```
<DB NAME=”Inqueritos” SGBD=”SQLServer” SO=”Win2003”  DATAC=”2007-05-11”
CRIADOR=”rfreitas” DATAP=”2008-04-15” …>
      <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"/>
                        <COLUMN NAME="DataCriacao" TYPE="smalldatetime" SIZE="24"
NULL="NO"/>
                        <COLUMN NAME="CodOwner" TYPE="varchar" SIZE="20" NULL="NO"/>
                        <COLUMN NAME="Fechado" TYPE="bit" SIZE="1" NULL="NO"/>
                        <COLUMN NAME="CodTipoDisponibilizacao" TYPE="varchar" SIZE="3"
NULL="NO"/>
                        <COLUMN NAME="CodTipoClassificacao" TYPE="int" SIZE="3" NULL="YES"/>
                        <COLUMN NAME="Descricao" TYPE="varchar" SIZE="300" NULL="YES"/>
                        <COLUMN NAME="InicioDisponibilizacao" TYPE="smalldatetime" SIZE="24"
NULL="YES"/>
                        <COLUMN NAME="FimDisponibilizacao" TYPE="smalldatetime" SIZE="24"
NULL="YES"/>
                        <COLUMN NAME="RespostaObrigatoria" TYPE="bit" SIZE="1" NULL="NO"/>
                        <COLUMN NAME="Instrucoes" TYPE="varchar" SIZE="400" NULL="YES"/>
                  </COLUMNS>
                  <KEYS>
                  …
                  </KEYS>
  </TABLE>
...
</STRUCTURE>
```

Bloco de Código 4 – Organização dos campos da Tabela

Estando as características para os atributos definidas, passamos agora para as chaves. Ainda para cada tabela, é necessário identificar quer as chaves primárias, quer as chaves estrangeiras.

Para as chaves primárias identificamos o atributo ou o conjunto de atributos que constituem a chave. Caso a chave primária possua apenas um atributo será definida como "*simple*"; no caso de chaves primárias constituídas por mais que um atributo estas serão definidas como "*composite*".

```
<KEYS>
        <PKEY TYPE="simple">
                <FIELD NAME="IDQuestionario"/>
        </PKEY>
        …
</KEYS>
```

Bloco de Código 5 – Estrutura das Chaves Primárias

Ficam a faltar as chaves estrangeiras. Nas chaves estrangeiras é fundamental identificar qual o atributo ou atributos que a constituem e também a tabela e os atributos onde esta é chave primária.

```
<KEYS>
        <FKEY NAME="CodOwner" IN="QuestionariosClassificacao" REF="CodOwner"/>
        <FKEY NAME="CodTipoClassificacao" IN="QuestionariosClassificacao"
REF="CodTipoClassificacao"/>
        <FKEY NAME="CodTipoQuestionario" IN="QuestionariosTipos" REF="CodTipoQuestionario"/>
        <FKEY NAME="CodTipoDisponibilizacao" IN="QuestionariosTiposDisponibilizacao"
REF="CodTipoDisponibilizacao"/>
</KEYS>
```

Bloco de Código 6 – Estrutura das Chaves Estrangeiras

É possível agora juntar todas estas definições e concluir a definição de toda a estrutura da base de dados. Temos em seguida uma instanciação, a título de exemplo, do elemento *STRUCTURE* onde se define toda parte da estrutura de base de dados utilizada no caso de estudo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DB NAME=”Inqueritos” SGBD=”SQLServer” SO=”Win2003”  DATAC=”2007-05-11”
CRIADOR=”rfreitas” DATAP=”2008-04-15” …>
        <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"/>
                                <COLUMN NAME="DataCriacao" TYPE="smalldatetime" SIZE="24"
NULL="NO"/>
                                <COLUMN NAME="CodOwner" TYPE="varchar" SIZE="20"
NULL="NO"/>
                                <COLUMN NAME="Fechado" TYPE="bit" SIZE="1" NULL="NO"/>
                                <COLUMN NAME="CodTipoDisponibilizacao" TYPE="varchar" SIZE="3"
NULL="NO"/>
                                <COLUMN NAME="CodTipoClassificacao" TYPE="int" SIZE="3"
NULL="YES"/>
                                <COLUMN NAME="Descricao" TYPE="varchar" SIZE="300"
NULL="YES"/>
                                <COLUMN NAME="InicioDisponibilizacao" TYPE="smalldatetime"
SIZE="24" NULL="YES"/>
                                <COLUMN NAME="FimDisponibilizacao" TYPE="smalldatetime"
SIZE="24" NULL="YES"/>
                                <COLUMN NAME="RespostaObrigatoria" TYPE="bit" SIZE="1"
NULL="NO"/>
                                <COLUMN NAME="Instrucoes" TYPE="varchar" SIZE="400"
NULL="YES"/>
                        </COLUMNS>

                        <KEYS>
                                <PKEY TYPE="simple">
                                        <FIELD NAME="IDQuestionario"/>
                                </PKEY>
                                <FKEY NAME="CodOwner" IN="QuestionariosClassificacao"
REF="CodOwner"/>
                                <FKEY NAME="CodTipoClassificacao" IN="QuestionariosClassificacao"
REF="CodTipoClassificacao"/>
                                <FKEY NAME="CodTipoQuestionario" IN="QuestionariosTipos"
REF="CodTipoQuestionario"/>
                                <FKEY NAME="CodTipoDisponibilizacao"
IN="QuestionariosTiposDisponibilizacao" REF="CodTipoDisponibilizacao"/>
                        </KEYS>
                </TABLE>

                <TABLE NAME="QuestionariosTipos">
                        <COLUMNS>
                                <COLUMN NAME="CodTipoQuestionario" TYPE="varchar" SIZE="3"
NULL="NO"/>
                                <COLUMN NAME="DesTipoQuestionario" TYPE="varchar" SIZE="30"
```

```
NULL="NO"/>
                        <COLUMN NAME="BotaoGravar" TYPE="bit" SIZE="1" NULL="YES"/>
                        <COLUMN NAME="BotaoConcluir" TYPE="bit" SIZE="1" NULL="YES"/>
                    </COLUMNS>
                    <KEYS>
                        <PKEY TYPE="simple">
                            <FIELD NAME="CodTipoQuestionario"/>
                        </PKEY>
                    </KEYS>
                </TABLE>
                …
        </STRUCTURE>
        <DATA>
                ...
        </DATA>
</DB>
```

Bloco de Código 7 – Fragmento do documento DBML do caso de estudo escolhido – STRUCTURE

5.3 DTD/XMLSchema – STRUCTURE

Promovendo a clarificação da definição do elemento *STRUCTURE* passamos à apresentação quer do DTD quer do *XMLSchema* correspondente.

Em síntese, podemos dizer que definimos o elemento *STRUCTURE,* o qual diz respeito à estrutura da base de dados, como sendo formado por uma sequência de, um ou mais, elementos *TABLE*. Por sua vez o elemento *TABLE* tem como filhos uma sequência de dois elementos, o elemento *COLUMNS* e o elemento *KEYS*.

O elemento *COLUMNS* é constituído por uma sequência de um ou mais elementos *COLUMN,* já o elemento o elemento *KEYS* é constituído por uma sequência de um elemento *PKEYS* e zero ou mais elementos *FKEYS*

STRUCTURE: Elemento que acomoda toda a estrutura da base de dados. A sua constituição é

definida por uma sequência de elementos *TABLE*.

TABLE: Cada tabela existente na base de dados é mapeada para um elemento *TABLE*. Este elemento possui um atributo – *NAME* – que corresponde ao nome da tabela. O elemento *TABLE* possui dois filhos – *COLUMNS* e *KEYS*.

COLUMNS: Este elemento visa agregar uma sequência de elementos *COLUMN* que correspondem aos campos da tabela.

COLUMN: Para cada campo ou coluna da tabela iremos ter um elemento *COLUMN*. Este elemento possui quatro atributos: *NAME*, *TYPE*, *SIZE* e *NULL*.

- **NAME:** Atributo que indica o nome do campo.
- **TYPE:** Atributo no qual se designa o domínio ou tipo dos dados da coluna em questão.
- **SIZE:** Atributo que indica o tamanho possível para os dados na coluna em questão.
- **NULL:** Atributo que indica se o campo em questão pode ou não ser nulo (vazio).

KEYS: O elemento *KEYS* possui como filho um elemento *PKEY* e pode ou não também possuir como filhos elementos *FKEY*.

PKEY: Elemento no qual se define a chave primária da tabela. Este elemento possui um atributo *TYPE* e é constituído por uma sequência de um ou mais elementos *FIELD*.

- **TYPE:** Atributo que pode assumir dois valores: *COMPOSITE* ou *SIMPLE*. Caso a chave primária seja formada por mais de que um campo da tabela este atributo assume o valor *COMPOSITE*, caso contrário em que a chave primária é apenas formada por um campo, o atributo assume o valor *SIMPLE*.

FIELD: Para cada campo que faz parte da chave primária iremos ter um elemento *FIELD*. Este elemento possui um atributo *NAME*.

- **NAME:** Atributo que define o nome do campo ou de um dos campos que constituem a chave primária.

FKEY: Elemento no qual definimos a ou as chaves estrangeiras. Este elemento contem três atributos: *NAME*, *IN* e *REF*.

- **NAME:** Atributo que define o nome do campo ou de um dos campos que constituem a chave estrangeira.
- **IN:** Este atributo indica o nome da tabela onde esta chave estrangeira é chave primária, ou seja, a tabela a qual esta chave estrangeira se refere.
- **REF:** Atributo que indica o campo ao qual este corresponde na tabela (definida no atributo *IN*) onde ele constitui chave primária.

Apresentamos em seguida, primeiro, o DTD e logo após o *XMLSchema* correspondente à definição do elemento *STRUCTURE.*

```
<!ELEMENT STRUCTURE (TABLE+)>
<!ELEMENT TABLE (COLUMNS, KEYS)>
<!ATTLIST TABLE
        NAME CDATA #REQUIRED>
<!ELEMENT COLUMNS (COLUMN+)>
<!ELEMENT COLUMN 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>
<!ELEMENT FKEY EMPTY>
<!ATTLIST FKEY
        NAME CDATA #REQUIRED
        IN CDATA #REQUIRED
        REF CDATA #REQUIRED>
```

Bloco de Código 8 – DTD para o Elemento STRUCTURE

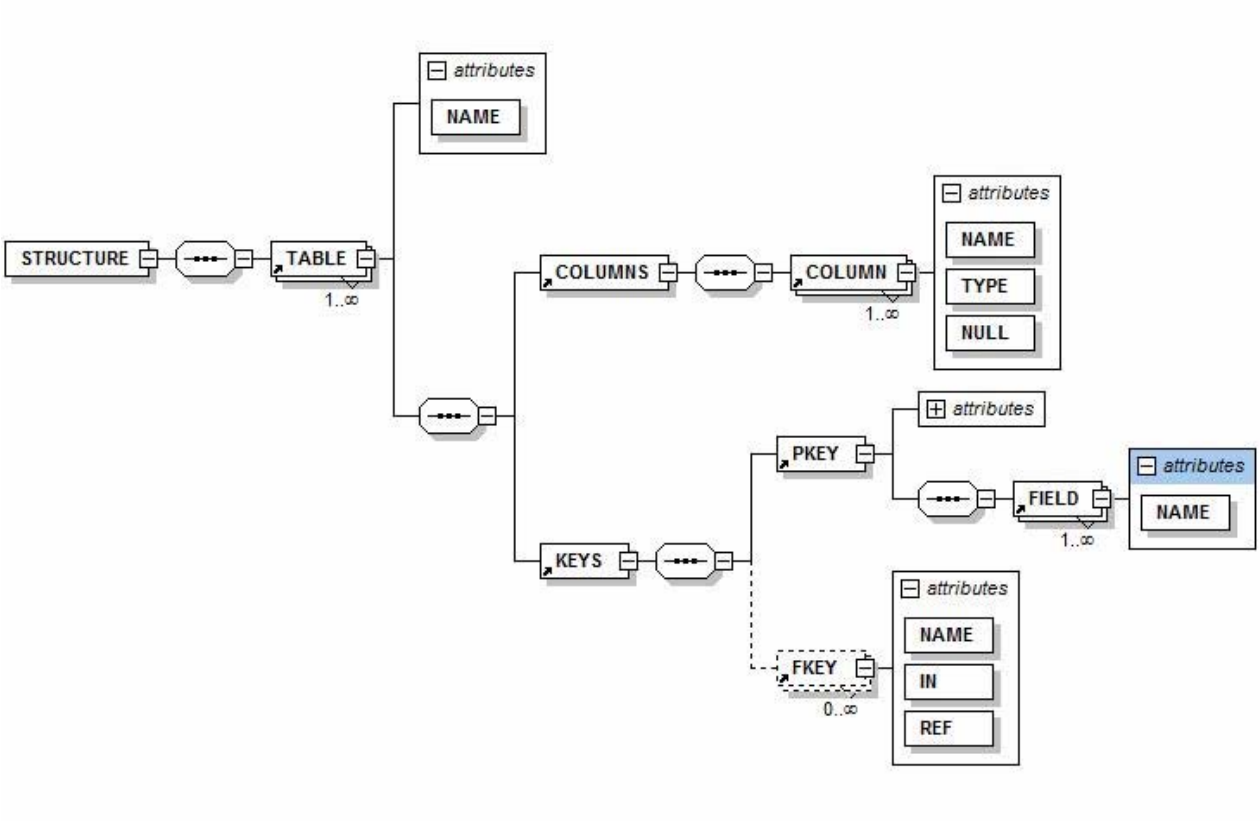

Figura 6 – XMLSchema para o Elemento STRUCTURE

5.4 Os dados

O documento XML deverá conter também os dados propriamente ditos. O ficheiro, que acomoda a estrutura da base de dados, irá acomodar a informação existente na base de dados. Para isso, já vimos que além do elemento *STRUCTURE*, cujo conteúdo reflecte a estrutura de tabelas e relacionamentos da base de dados, temos também o elemento *DATA*. Este elemento é então responsável por acomodar toda a informação, que podemos encontrar nas relações ou tabelas. Temos em seguida uma instância que mostra claramente como se procede ao

armazenamento no ficheiro DBML dos dados existentes na base de dados.

```
<DATA>
	<Questionarios>
		<Questionarios-REG>
			<IDQuestionario>1</IDQuestionario>
			<Nome>Actividades Científico - Pedagógicas (Data limite de resposta 26-04-
2004)</Nome>
			<CodTipoQuestionario>INQ</CodTipoQuestionario>
			<DataCriacao>2004-02-04 15:00:00.000</DataCriacao>
			<CodOwner>00000001</CodOwner>
			<Fechado>0</Fechado>
			<CodTipoDisponibilizacao>DAT</CodTipoDisponibilizacao>
			<CodTipoClassificacao/>
			<Descricao>Com o intuito de conhecer a capacidade científico-pedagógica da
Universidade Lusíada, solicita-se a V. Excia o preencimento do seguinte questionário</Descricao>
			<InicioDisponibilizacao>2004-02-12 12:00:00.000</InicioDisponibilizacao>
			<FimDisponibilizacao>2006-04-26 23:59:00.000</FimDisponibilizacao>
			<RespostaObrigatoria>1</RespostaObrigatoria>
			<Instrucoes>&lt;br&gt;&#13;
&lt;b&gt;Instruções de Preenchimento:&lt;/b&gt;&lt;br&gt;&#13;
- Para gravar as respostas do inquérito carregue no botão "Gravar". Utilizando este botão poderá, mais
tarde alterar as respostas.&lt;br&gt;&#13;
- Para concluir o inquérito carregue no botão "Concluir". &lt;br&gt;&#13;
&lt;DIV cla</Instrucoes>
		</Questionarios-REG>
		<Questionarios-REG>
			<IDQuestionario>2</IDQuestionario>
			<Nome>Proposta de aquisição de Bibliografia (Disciplina 1)</Nome>
			<CodTipoQuestionario>INQ</CodTipoQuestionario>
			<DataCriacao>2005-11-07 11:30:00.000</DataCriacao>
			<CodOwner>00000001</CodOwner>
			<Fechado>0</Fechado>
			<CodTipoDisponibilizacao>DAT</CodTipoDisponibilizacao>
			<CodTipoClassificacao/>
			<Descricao>Ano lectivo 2005/06 (Disciplina 1)</Descricao>
			<InicioDisponibilizacao>2005-11-07 12:00:00.000</InicioDisponibilizacao>
			<FimDisponibilizacao>2005-11-30 23:59:00.000</FimDisponibilizacao>
			<RespostaObrigatoria>1</RespostaObrigatoria>
			<Instrucoes>&lt;br&gt;</Instrucoes>
		</Questionarios-REG>
	</Questionarios>
	<QuestionariosTipos>
		<QuestionariosTipos-REG>
			<CodTipoQuestionario>EXA</CodTipoQuestionario>
			<DesTipoQuestionario>Exame</DesTipoQuestionario>
			<BotaoGravar>1</BotaoGravar>
			<BotaoConcluir>0</BotaoConcluir>
		</QuestionariosTipos-REG>
		<QuestionariosTipos-REG>
```

```
                    <CodTipoQuestionario>INQ</CodTipoQuestionario>
                    <DesTipoQuestionario>Inquérito</DesTipoQuestionario>
                    <BotaoGravar>1</BotaoGravar>
                    <BotaoConcluir>1</BotaoConcluir>
                </QuestionariosTipos-REG>
                <QuestionariosTipos-REG>
                    <CodTipoQuestionario>MIN</CodTipoQuestionario>
                    <DesTipoQuestionario>Mini-Teste</DesTipoQuestionario>
                    <BotaoGravar>1</BotaoGravar>
                    <BotaoConcluir>1</BotaoConcluir>
                </QuestionariosTipos-REG>
        </QuestionariosTipos>
        ...
</DATA>
```

Bloco de Código 9 – Fragmento do elemento DATA

Falando em termos de DTD e *XMLSchema*, facilmente se percebe que não se trata de alguma estrutura difícil de representar. O problema fulcral coloca-se na questão dos elementos possuírem nomes que variam em função do nome das tabelas e também dos atributos. Atendendo a isto e se pretendermos validar o nosso ficheiro antes de o ingerir no OAIS teremos que elaborar ou um DTD ou um *XMLSchema* específico para cada base de dados. Apresentamos em seguida um DTD e um *XMLSchema* para o elemento *DATA* do nosso caso de estudo.

```
<!ELEMENT DATA (Questionarios, QuestionariosTipos)>
<!ELEMENT Questionarios (Questionarios-REG+)>
<!ELEMENT Questionarios-REG (IDQuestionario, Nome, CodTipoQuestionario, DataCriacao,
CodOwner, Fechado, CodTipoDisponibilizacao, CodTipoClassificacao, Descricao, InicioDisponibilizacao,
FimDisponibilizacao, RespostaObrigatoria, Instrucoes)>
<!ELEMENT QuestionariosTipos (QuestionariosTipos-REG+)>
<!ELEMENT QuestionariosTipos-REG (CodTipoQuestionario, DesTipoQuestionario, BotaoGravar,
BotaoConcluir)>

<!ELEMENT BotaoConcluir (#PCDATA)>
<!ELEMENT BotaoGravar (#PCDATA)>
<!ELEMENT CodOwner (#PCDATA)>
<!ELEMENT CodTipoClassificacao EMPTY>
<!ELEMENT CodTipoDisponibilizacao (#PCDATA)>
<!ELEMENT CodTipoQuestionario (#PCDATA)>
<!ELEMENT DataCriacao (#PCDATA)>
<!ELEMENT DesTipoQuestionario (#PCDATA)>
<!ELEMENT Descricao (#PCDATA)>
```

```
<!ELEMENT Fechado (#PCDATA)>
<!ELEMENT FimDisponibilizacao (#PCDATA)>
<!ELEMENT IDQuestionario (#PCDATA)>
<!ELEMENT InicioDisponibilizacao (#PCDATA)>
<!ELEMENT Instrucoes (#PCDATA)>
<!ELEMENT Nome (#PCDATA)>
<!ELEMENT RespostaObrigatoria (#PCDATA)>
```

Bloco de Código 10 – DTD para o Elemento DATA

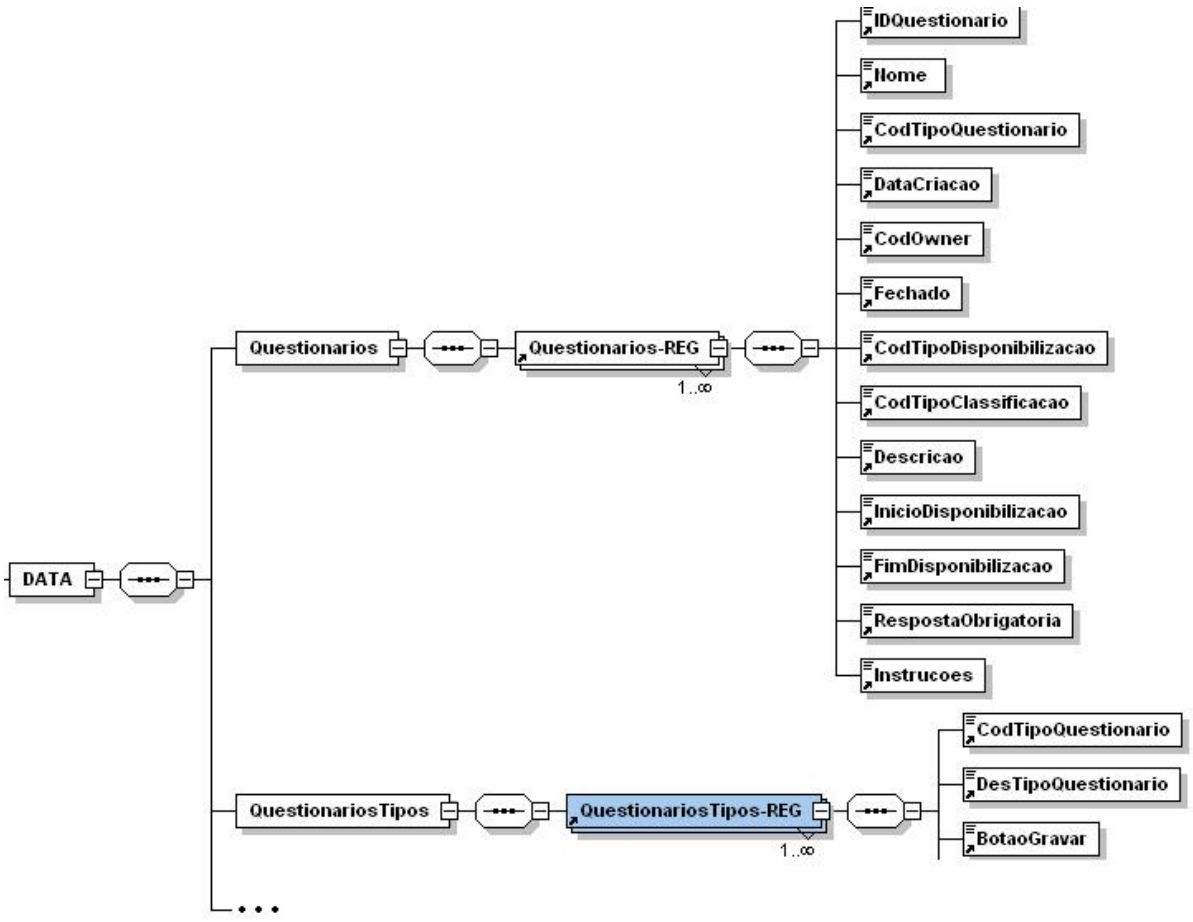

Figura 7 – XMLSchema para o Elemento DATA

Podemos agora dizer que a estrutura para o ficheiro DBML está totalmente definida. É de

realçar que a estrutura se prende em muito às características ou propriedades consideradas por nós importantes e fundamentais para a preservação de uma base de dados. Questões como *Store Procedures* não foram incluídas para preservação. No nosso entender, por exemplo, um *Store Procedure* prende-se mais com a parte aplicacional ou semântica da base de dados. É também sabido que um *Store Procedure* pode perfeitamente ser recriado; a sua perda não pode ser considerada crítica uma vez que os dados e a estrutura de dados estão preservados.

6

Arquitectura do Sistema

Neste capítulo procuraremos descrever e analisar a arquitectura do sistema proposto e concretizado, tendo como finalidade a Preservação Digital de Bases de Dados Relacionais. Em termos conceptuais, o nosso sistema baseia-se no modelo de referência OAIS, previamente analisado no capítulo 4. Numa primeira fase, a preocupação será clarificar algumas questões ainda soltas. O modelo de referência OAIS, como já foi visto, não impõe rigidez no que diz respeito à implementação, mas antes define uma série de recomendações. Assim sendo, houve antes de mais a necessidade de pensar e conceptualizar antes de concretizar. Foi nesta fase que uma série de questões foram levadas em consideração. Algumas destas são as tecnologias envolvidas no desenvolvimento e as linguagens de programação usadas, como será feita a ingestão dos SIPs, a administração dos AIPs, a disseminação dos DIPs e como serão implementadas as políticas de preservação. A plataforma e paradigma pretendidos também são decisivos para a implementação final.

6.1 O Repositório

O que foi idealizado consiste na elaboração de uma aplicação *Web*, a qual terá a função de, a partir de uma base de dados concreta, assistir o utilizador na criação de um SIP, que depois poderá ser ingerido pelo repositório. Numa outra aplicação *Web*, teremos uma interface de pesquisa com a qual será possível inquirir o sistema sobre determinada base de dados, que se encontra armazenada e que será devolvida sob a forma de um ou mais DIPs. No repositório, além de encontrarmos a base de dados preservada em DBML, esta deverá também estar guardada num SGBD. Ao usarmos também um SGDB, que no nosso caso será o MySQL, oferecemos aos consumidores uma forma prática para navegar e aceder aos DIPs – SQL. Através de *querys* SQL, as consultas tornam-se indubitavelmente mais simples do que se usarmos *querys* directas no documento DBML – *XPath*.

A opção pelo uso de uma plataforma baseada na *Web* foi considerada acertada para a elaboração e implementação do projecto tendo em conta um conjunto de factores. Para a tomada desta decisão pesou a experiência já adquirida no que diz respeito ao uso da linguagem de programação usada no desenvolvimento – PHP. Um factor importante e decisivo para a escolha deste tipo de plataforma foi a exponencial utilização de serviços de Internet em conjunto com a facilidade de acesso ao sistema a partir de qualquer localização geográfica. O desenvolvimento, programação e implementação de sistemas *Web* que interagem com diversas bases de dados foi um factor que também nos levou a optar por este paradigma. A decisão foi então tomada e deu-se início à análise mais concreta no que se refere a tecnologias envolvidas – sistema operativo, ligações às bases de dados, linguagens de programação, etc. Existindo a possibilidade de utilização de tecnologias *Open Source*, a opção foi caminhar nesse sentido.

Começando pelo sistema operativo, a escolha caiu sobre o Linux. A distribuição usada não é importante referir nesta fase, uma vez que também não condiciona de forma crítica o desenvolvimento e implementação do nosso sistema para a preservação digital. Estando nós a

lidar com a problemática da preservação digital, devemos ter a preocupação de escolher as tecnologias, para o desenvolvimento do sistema, que ofereçam algumas garantias nesse sentido. A plataforma Linux é uma das mais usadas, principalmente a nível de servidores, e cada vez mais por organizações e instituições em todo mundo. A correr sobre o sistema operativo, iremos ter o servidor HTTP *Apache*, também este um dos mais usados em todo mundo.

Para a ligação e acesso às bases de dados utilizamos a conexão via ODBC; do inglês *Open Data Base Connectivity*. Como a sigla sugere falamos de um padrão aberto para aceder a bases de dados. Digamos que esta tecnologia oferece uma API para interagir com as mais variadas bases de dados, desde que estas possuam suporte ODBC – praticamente todas. Para o nosso protótipo foi usado o *unixODBC Project*, cujo objectivo é promover e disponibilizar um standard para acesso a plataformas que possuam suporte ODBC. Grande parte das distribuições Linux já inclui o suporte unixODBC.

É necessária agora uma linguagem de programação para o desenvolvimento. Foi escolhido o PHP e foi, como já referimos, a experiência de programação com esta linguagem um dos factores que também pesou nas decisões relativamente à implementação. PHP do inglês *Hypertext Preprocessor* é uma linguagem de programação para o paradigma *World Wide Web*. Actualmente o PHP já inclui a programação Orientada a Objectos.

Além destas tecnologias foi usado o MySQL como SGBD para dar o tal suporte às bases de dados que se encontram preservadas no repositório. Contudo este conceito será abordado já de seguida ao falarmos concretamente sobre os processos de ingestão e disseminação.

6.2 A Arquitectura

Como havíamos referido anteriormente, o repositório tem por base uma plataforma *Web*. O

modelo de referência OAIS é também um requisito para o nosso repositório. A plataforma possibilita então a criação e ingestão de SIPs, tranforma-os em AIPs e facilita a disseminação dos DIPs. Existe também uma interface na qual é possível administrar o repositório e também implementar as necessárias políticas de preservação. A figura ilustra genericamente o repositório e as interacções dos diferentes componentes e agentes. Na figura, também podemos observar o percurso dos dados a serem preservados – base de dados.

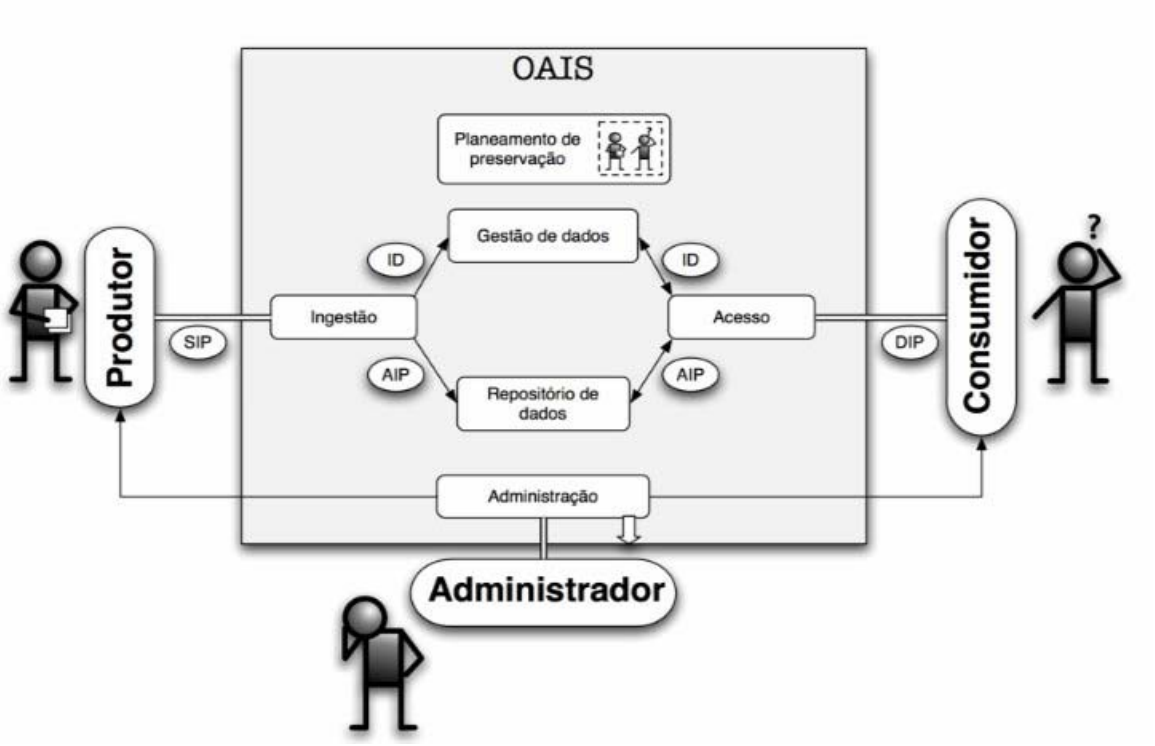

Figura 8 – Arquitectura do Repositório baseada no Modelo de Referência OAIS. Retirado de "XML e Preservação Digital" [24]

Voltando a referenciar o modelo OAIS, a constituição do sistema passa pela existência de três actores, assim como três mega-processos e também três tipos de pacotes de informação. Os actores são o produtor, o consumidor e o administrador. No que diz respeito aos processos mais relevantes para o repositório, temos o processo de ingestão, o processo de administração (inclui-

-se neste processo a implementação de políticas de preservação) e, por fim, o processo de disseminação. Quando nos referimos aos tipos de pacotes de informação, estamos a falar obviamente dos SIPs, AIPs e DIPs. Todos estes componentes estão directamente relacionados, ora vejamos: o produtor é responsável pela criação dos SIPs a serem ingeridos no repositório; já o administrador lida com os AIPs no processo de administração; o consumidor recebe os DIPs através do processo de disseminação dos dados. Até aqui nada de novo, uma vez que seguimos as recomendações do modelo de referência OAIS, consoante já foi explorado anteriormente.

Passamos agora a concretizar todos estes componentes, processos e interacções, tendo como pano de fundo o nosso repositório para preservação digital de bases de dados relacionais.

6.3 Processo de Ingestão

Olhemos para o pacote de informação SIP. A constituição e estrutura dum SIP têm que ser negociadas entre o produtor e o arquivo ou repositório. A definição ou concretização dos SIPs é então definida através de um contrato de submissão de dados estabelecido entre o Produtor e o arquivo. Além da definição da estrutura do SIP, o contrato serve ainda para especificar as várias etapas do processo de ingestão. Após a referida negociação, definimos o contrato de submissão dos dados.

O SIP é constituído por uma meta-informação descritiva, meta-informação técnica e pela base de dados propriamente dita. Para a meta-informação descritiva usamos a norma, já descrita anteriormente, *Dublin Core* – um documento em XML com informação descritiva acerca do objecto digital que se pretende preservar. A meta informação técnica, assim como a base de dados serão incorporadas num único ficheiro XML – DBML. A estrutura do documento DBML já foi especificada no capítulo 5. É importante ressalvar que a informação binária contida na base de dados segue no SIP em ficheiros separados.

Com o SIP pronto para ser entregue segue-se o processo de ingestão (Fig. 9). As etapas do processo de ingestão são elas próprias definidas também através do contrato de submissão de dados.

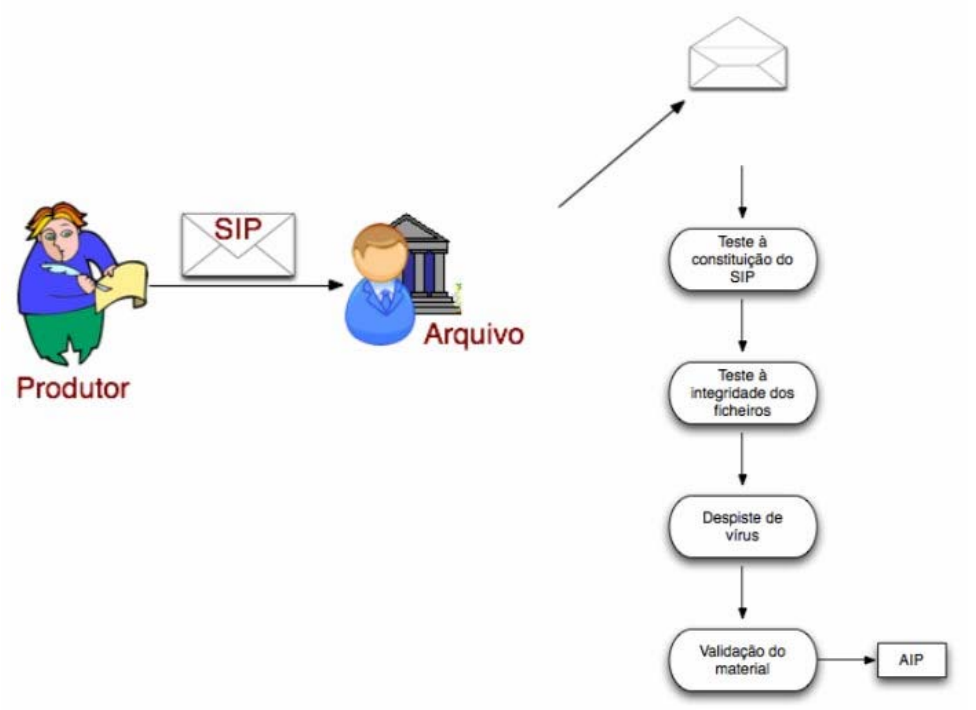

Figura 9 – Processo de Ingestão (SIP). Retirado de "Sistemas de suporte à Governação Electrónica" [5]

A primeira etapa é verificar a constituição do SIP, isto é, verificar se o SIP contém o ficheiro XML com a meta-informação descritiva, o ficheiro DBML e eventualmente ficheiros binários correspondentes à informação binária que possa existir na base de dados. Na etapa seguinte vamos proceder à validação, através de *XMLSchemas*, quer do ficheiro XML com a meta-informação descritiva quer do ficheiro DBML (meta-informação técnica e base de dados). Nesta etapa também se procede à verificação da correspondência entre os campos binários e os ficheiros a eles associados. Existe também uma etapa onde podemos fazer uma verificação de existência de vírus. Depois de validarmos todo o material preparamo-nos para deixar de ter um

SIP e passar a ter um AIP.

Em termos técnicos a criação do SIP passa, numa primeira fase, pela definição da base de dados que se pretende preservar. Isto é feito através do preenchimento de um formulário numa página *Web* (Fig. 10), na qual se define o SGBD em causa; assim como o caminho para a ligação à base de dados (pode ser local ou remota); ter-se-á que especificar também qual o utilizador e *password* com as permissões necessárias para usar na ligação. Estas informações são passadas através do PHP ao unixODBC e a ligação é efectuada.

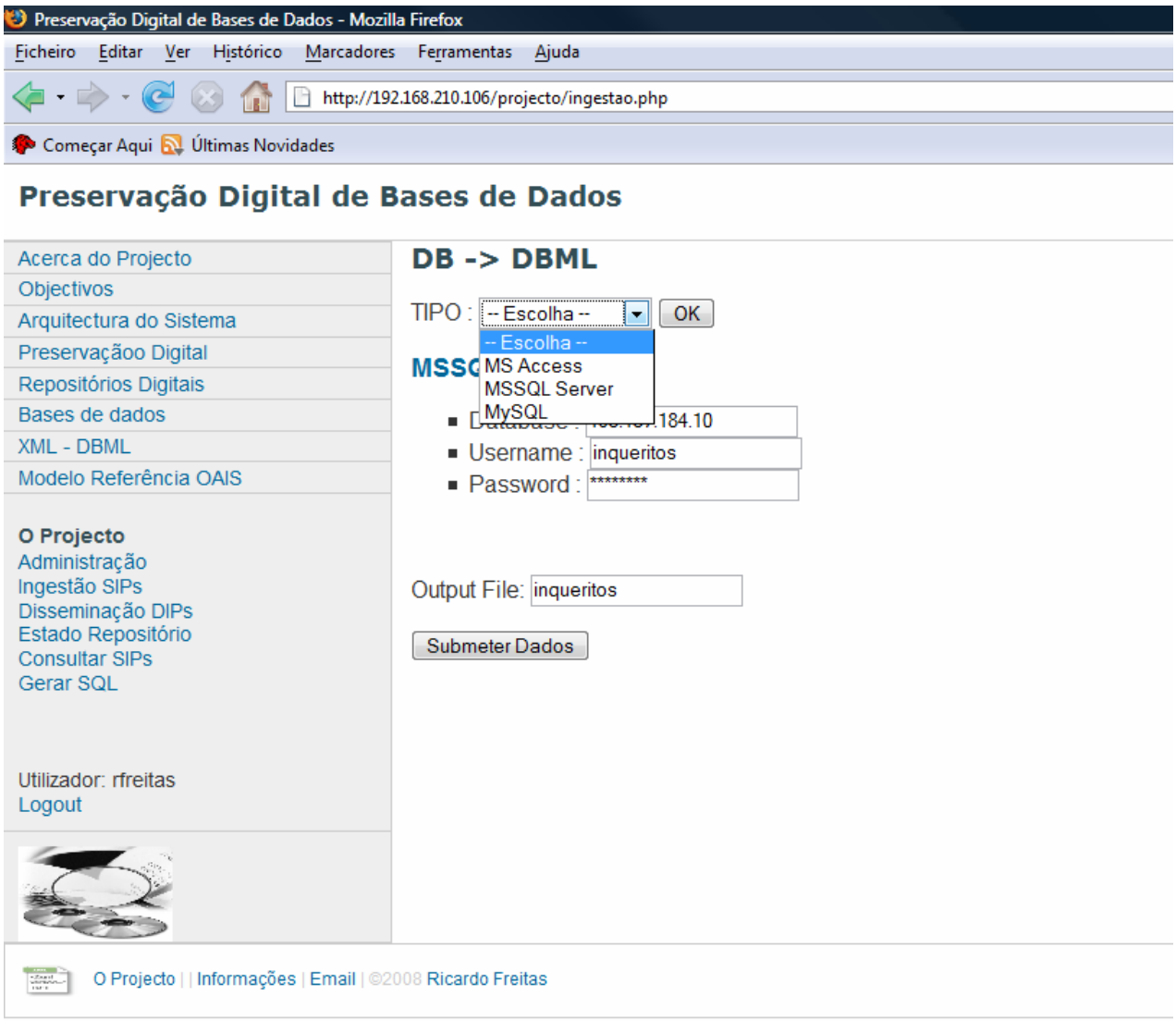

Figura 10 – Interface Web para definição da Base de Dados a preservar

Foi criada uma classe *db* que serve especificamente para efectuar esta ligação e carregar toda a informação necessária da base de dados para *array's* temporários do PHP. Como é sabido, existem diferentes SGBD. Contudo esta classe foi criada para que a ligação seja transparente. Uma vez que no formulário é definido o SGBD e desde que o unixODBC possua o *driver* para esse SGBD, temos a possibilidade de construir o DSN (*Data Source Name*) e então efectuar a ligação.

Concretamente, a classe *db* possui um construtor onde também se define o DSN. Em relação aos métodos da classe, temos um que estabelece ligação à base de dados – *connect*; temos um outro que carrega para um *array* o nome das tabelas existentes na base de dados – *tables*; outro método irá preencher um *array* com a informação acerca dos atributos das várias tabelas – *get_structure_columns*; o método responsável por inserir num outro *array* as chaves primárias e estrangeiras é o método *get_structure_keys;* finalmente teremos um método que irá povoar um *array* com todos os dados armazenados na base de dados – *get_data*. Todos estes métodos usam a extensão ODBC, que o PHP possui para ligação às bases de dados.

Encontramo-nos agora numa situação em que toda a informação relevante para a preservação se encontra armazenada em variáveis membros da classe *db* (Fig. 11).

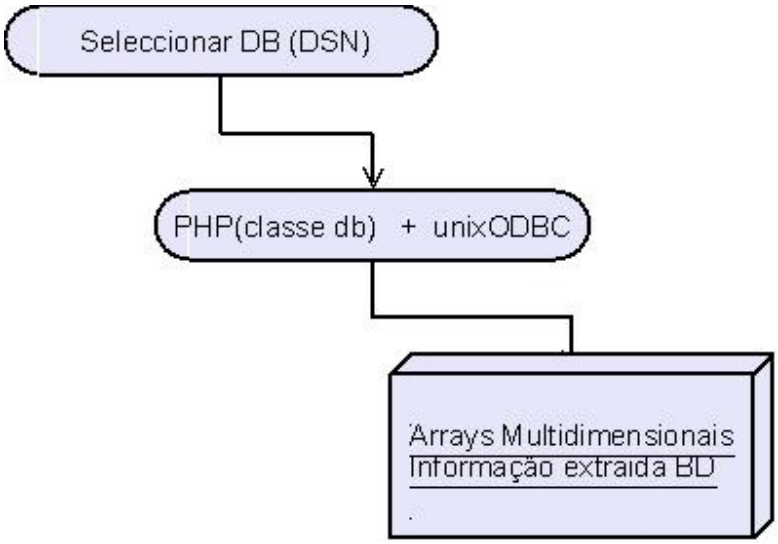

Figura 11 – Conexão e Extracção da informação contida na Base de Dados

O carregamento para variáveis do PHP dos dados relativos à estrutura e dos dados relativos à informação guardada na base de dados é efectuada para possibilitar, ao PHP, a manipulação de

toda esta informação no sentido de criar o ficheiro DBML.

A próxima etapa, ainda nesta fase de criação do SIP, é a de gerar o DBML e, de seguida, criar o ficheiro em disco. Criamos então outra classe para este efeito – *dbml*. Esta classe é constituída por dois métodos: um para gerar o elemento *STRUCTURE – gera_dbml_structure –* e outro para gerar o elemento *DATA – gera_dbml_data*. Nestes dois métodos, usamos a extensão do PHP que permite manipular documentos XML, *simpleXML*. Aquando da construção da classe *dbml*, passamos como parâmetros ao seu construtor os *arrays* que comportam toda a informação extraída da base de dados pela classe precedente.

Depois da execução dos métodos que geram o XML, mediante a extensão *simpleXML*, construir-se-á uma *string* correspondente ao que se pretende para o ficheiro DBML. Usando o PHP, estamos em condições para criar em disco o referido ficheiro. Além deste ficheiro, que corresponde à base de dados e à meta-informação técnica, um outro ficheiro é também criado seguindo a norma *Dublin Core* para armazenar a meta-informação descritiva.

A questão que se prende com a informação binária armazenada na base de dados é também tratada nesta fase, através do PHP. É primeiramente referenciada aquando do carregamento da informação existente na base de dados para *arrays* do PHP, para agora serem criados ficheiros independentes para cada dado binário da base de dados. O nome do ficheiro irá identificar a tabela a que pertence, assim como o atributo e tuplo ao qual se refere.

Uma *script* em PHP irá verificar a constituição do SIP e, se o valor retornado for verdadeiro, significa que o nosso pacote de informação SIP está pronto. Segue-se a fase da validação. Voltamos a usar o PHP para nos ajudar nesta etapa. Introduzimos agora a extensão DOM do PHP, uma vez que será através do uso do método *schemaValidate* da classe *DOMDocument* desta extensão, que iremos proceder à validação dos dois ficheiros XML que fazem parte do SIP. Através dos respectivos *XMLSchemas*, procede-se à validação do XML de ambos os ficheiros (Fig. 12).

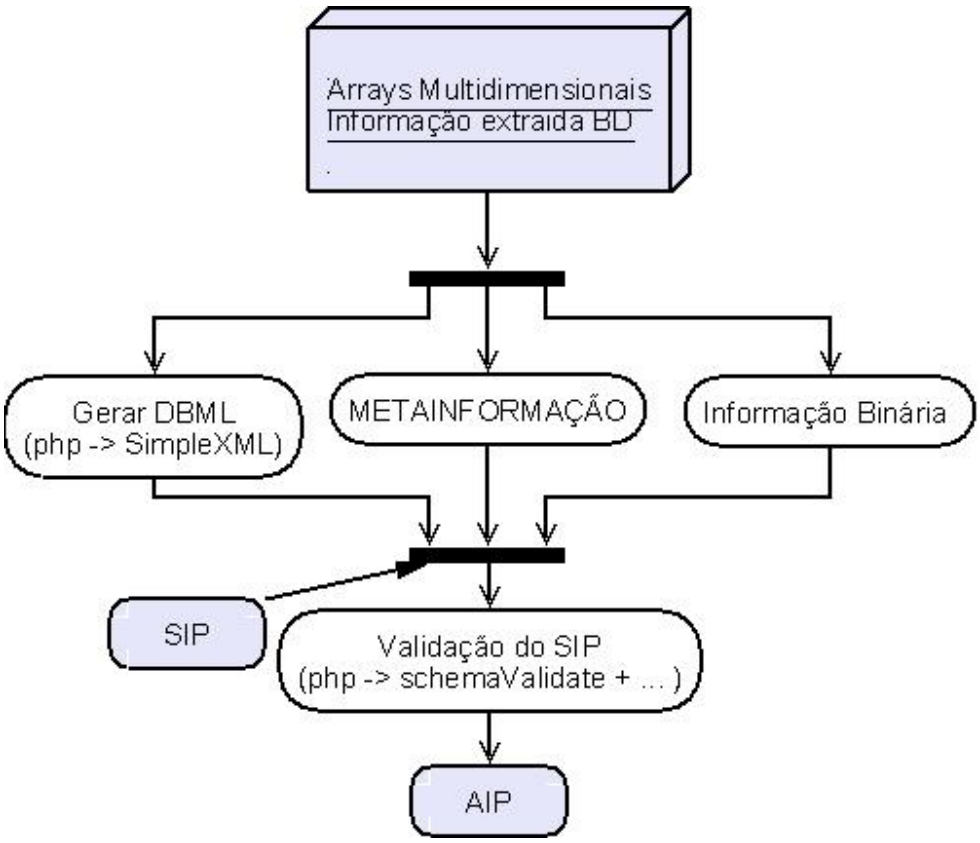

Figura 12 – Construção do SIP + Validação

Para a validação da parte binária foi construído um método específico para verificar e testar a correcta correspondência entre os ficheiros binários e as referências na base de dados (DBML). Em seguida, procede-se a uma despistagem de vírus (ainda não implementado) e posto isto, consideramos que todo o material está validado e pronto para se tornar num AIP.

6.4 Processo de Administração

Passamos à fase na qual surge a preocupação com a administração dos Pacotes de Informação, agora AIPs. Estes pacotes encontram-se efectivamente armazenados no repositório e certamente haverá a necessidade, com o passar dos tempos, de os manipular de alguma forma. Algumas das actividades e responsabilidades neste processo passam pela garantia de disponibilidade da informação guardada para a respectiva comunidade de interesse. Políticas de preservação devem, então, ser implementadas no sentido de garantir esse acesso continuado e a longo-prazo pelos utilizadores com interesse na informação arquivada.

Para o armazenamento dos AIPs usamos o sistema de ficheiros do sistema operativo. Foi criado um directório com o nome de *Repositório*, no qual serão criados os vários directórios correspondentes aos AIPs. Portanto, no interior do directório *Repositório* existirá um directório para cada AIP armazenado. O directório correspondente ao AIP tem o nome do tipo: *AIP_**, em que o * deverá corresponder ao nome da base de dados relacional arquivada.

Através da interface *Web* temos a possibilidade de obter uma listagem com informação descritiva dos vários AIPs armazenados no sistema. Existirá a possibilidade de aceder a cada um dos AIPs com o intuito de intervir no sentido de garantir a sua preservação e consequente disponibilidade para a comunidade de interesse.

De salientar que é nesta fase que se dá início ao processo de Disseminação da informação que iremos falar de seguida. Existirá nesta fase a responsabilidade de gerar, a partir de um documento DBML, o SQL necessário para reconstruir a base de dados no SGBD MySQL. Uma classe em PHP foi construída para este efeito. Usamos uma vez mais a extensão DOM do PHP para, neste caso, poder aceder ao documento DBML e extrair a informação (*XPath*) necessária.

6.4.1 Operações Disponíveis no Sistema

As várias interfaces *Web* que vimos mencionando só se encontram acessíveis mediante uma

autenticação no sistema. Existe uma base de dados (MySQL) que dá suporte a esta validação de acessos. Compete também ao componente administração gerir estes acessos, assim como os diferentes privilégios no que concerne à manipulação da informação arquivada.

Em termos de utilizadores do sistema estes podem ser divididos em dois tipos, ou seja, dois perfis de utilizadores: administradores e utilizadores.

Um utilizador tem ao seu dispor a seguinte lista de operações:

- Criação de SIPs
- Ingestão de SIPs no repositório (AIPs)
- Consulta do estado do repositório
- Produção de SQL a partir de AIPs
- Disseminação de AIPs

O administrador tem ao seu dispor todas as operações disponíveis aos utilizadores com o acréscimo das operações inerentes ao componente administração:

- Gestão dos utilizadores do sistema
- Acesso directo ao sistema de ficheiros (directório *Repositório*)
- Manipulação dos *drivers* (unixODBC) para ligação às bases de dados
- Monitorização do estado de obsolescência da informação no repositório
- Acções de migração da informação armazenada sempre que necessário (políticas de preservação)

6.5 Processo de Disseminação

Segundo o modelo de referência OAIS, a informação armazenada no repositório deverá estar acessível mediante a disponibilização de Pacotes de Informação. Estes pacotes, denominados de

DIPs, podem assumir diversas formas desde que consigam oferecer aos consumidores a informação por eles desejada. Contudo deverá existir, à semelhança do que acontece entre o Produtor e o Arquivo, uma negociação prévia, entre o Arquivo e o Consumidor, para definir como se irá dar o processo de Disseminação e qual a forma que os DIPs deverão assumir.

No repositório, a base de dados encontra-se preservada essencialmente no formato DBML. Decisão já justificada com a neutralidade do XML, assim como com a sua legibilidade quer por máquinas quer por seres humanos. Todavia no que concerne à disseminação da informação, optou-se pela reconstrução da base de dados no MySQL através da geração de código SQL. Escolhemos o MySQL pelas razões já mencionadas, mas urge salientar que uma vez gerado o código SQL correspondente a toda a base de dados a reconstrução poderá ser implementada num qualquer outro SGBD. De ressalvar as ligeiras diferenças que podem surgir na sintaxe SQL de um SGBD para outro (poder-se-ia colmatar este problema com a parametrização destas questões na classe que gera o SQL a partir do DBML,).

Para a interacção por parte dos consumidores com a base de dados preservada, encontramos então esta forma de agilizar as dificuldades inerentes às pesquisas directas em documentos DBML. Desta forma, criamos uma versão da base de dados arquivada também num SGBD – MySQL – facilitando o processo de Disseminação.

A classe criada denomina-se de *sql* e, quando instanciada, define um objecto (*DOMDocument)* a partir do DBML do repositório. Os métodos da classe possibilitam a conversão desse objecto DBML para SQL. Todas as instruções SQL necessárias para uma reconstrução da base de dados num qualquer motor de base de dados (SGBD) são nesta fase definidas. Através do SQL gerado podemos então reconstruir a base de dados no MySQL.

Com a Base de Dados original reconstruída no MySQL, vamos usar a ferramenta *phpMyAdmin* no processo de Disseminação dos pacotes de informação. A negociação entre o Arquivo e o Consumidor é neste processo de disseminação relativamente simples. Digamos que passará pela execução de *querys* (*phpMyAdmin*) à base de dados, onde o critério de filtragem da cláusula

WHERE funcionará como a tentativa de descobrir informação de interesse na base de dados. Num momento anterior, foi seleccionada a base de dados a consultar através do uso da informação descritiva (*Dublin Core*) do AIP (Fig. 13).

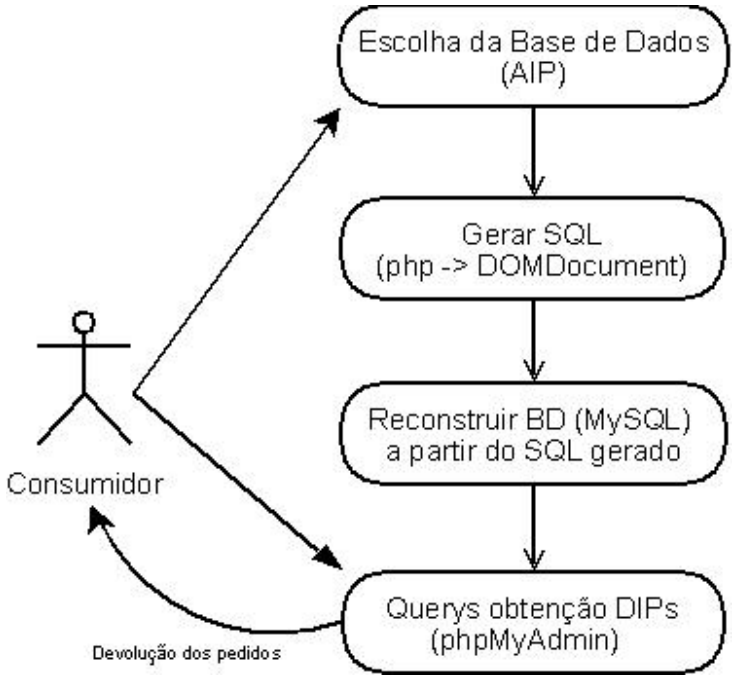

Figura 13 – Interacções do Consumidor com o sistema (Disseminação)

A ferramenta *phpMyAdmin* é ela própria escrita em PHP e tem como objectivo fazer a administração de bases de dados em MySQL. Achou-se por bem não criar uma ferramenta específica para esse efeito, uma vez que através desta ferramenta é possível atingir todo tipo de interacção com a versão da base de dados armazenada no MySQL. Talvez se possa pensar em construir também uma interface *Web* especializada para o processo de Disseminação em trabalhos futuros.

7

Conclusão

Chegados a esta fase, compete-nos estabelecer algumas sínteses e tirar algumas conclusões acerca do estudo e trabalhos desenvolvidos.

Numa das vertentes temos a revisão da literatura e estudo efectuado no âmbito da preservação digital e mais concretamente preservação digital de bases de dados relacionais. Numa outra vertente, posicionamos a componente prática na qual foi implementado um protótipo de um repositório para base de dados relacionais.

Vamos em primeiro lugar olhar para a implementação prática, focando os testes efectuados e os resultados obtidos.

Terminaremos tentando firmar alguns consensos e ideias relativamente à temática da preservação digital em geral, e relativamente à preservação digital de bases de dados relacionais em particular.

7.1 Protótipo – Testes e Resultados

A implementação prática revelou-se bastante significativa uma vez que foi possível obter resultados interessantes facilmente. Após a montagem inicial de todas as ferramentas necessárias ao desempenho das tarefas pretendidas, rapidamente se começou a desenvolver o protótipo. Alguns ajustes foram efectuados nesta fase objectivando um melhor desempenho do sistema.

Foram efectuados testes com diferentes bases de dados e diferentes SGBD: MS Access[1], MS SQL Server[2] e MySQL. Destes três, o MS Access (formato 2000) oferece mais dificuldades no que concerne à extracção da estrutura da base de dados. Certamente, isto deve-se à natureza mais fechada do SGBD. Digamos que para atingir os objectivos pretendidos com o MS Access foi necessário contornar alguns obstáculos, nomeadamente a utilização de uma ferramenta adicional – MDB Tools[3].

Contudo, urge salientar que o desenvolvimento deste trabalho visou de certa forma testar a viabilidade de se preservar digitalmente bases de dados relacionais deste modo. Tal foi efectivamente possível, isto é, foi atingido o objectivo de converter bases de dados relacionais (diferentes SGBD) para DBML.

Uma vez extraída a informação (dados + estrutura) da base de dados, o processo de criação dos SIPs é conseguido ao nível da programação em PHP. Depois de realizados alguns testes, algumas alterações foram efectuadas ao nível dos métodos (classe *dbml*) com o objectivo de melhorar o desempenho na geração do DBML. O tempo e recursos utilizados pelo sistema foi um dos aspectos a melhorar, uma vez que nos casos em que as bases de dados atingem determinadas dimensões, o processamento dos dados tornava-se muito demorado.

1 http://www.microsoft.com/

2 http://www.microsoft.com/

3 http://mdbtools.sourceforge.net/

Muito provavelmente, mais melhorias a este nível serão possíveis. Outra fase que envolve também bastante processamento é a fase de geração do SQL a partir do DBML. Em trabalhos futuros poder-se-á apostar em outras melhorias no que concerne ao processamento, isto é, em relação à análise e geração de código (DBML, SQL).

7.2 Conclusões e Consensos obtidos

Através do estudo efectuado, foi possível estabelecer alguns consensos relativamente à temática da preservação digital. De referir que existem diferentes estratégias e abordagens ao problema, e que nenhuma delas sozinha consegue satisfazer todas as necessidades no que à preservação digital diz respeito. É de alguma forma consensual que, para diferentes tipos de artefactos digitais, diferentes estratégias devem ser adoptadas. Casos existirão onde a preservação só se torna viável mediante uma conjugação de estratégias de preservação digital.

Desde o início que este trabalho revelou-se bastante interessante devido ao facto de tratarmos de um tema actual e com implicações futuras. Sabemos que o número de objectos digitais não parará de aumentar nos próximos anos e por isso esta é uma problemática séria e que justifica veementemente os trabalhos e estudos realizados.

As organizações tomaram consciência deste problema ao ponto de se ter concebido um modelo de referência (OAIS) para fornecer ajudas na resolução desta problemática. O modelo de referência OAIS foi efectivamente objecto do nosso estudo e por nós adoptado na concepção do protótipo. Este modelo, que aborda directamente esta temática, é considerado um *standard* ISO e estabelece uma série de recomendações e define um conjunto de vocabulário para responder ao desafio da preservação digital. O modelo de referência OAIS pode revelar-se de grande utilidade para as organizações se estas o adoptarem como seu modelo conceptual para construção de arquivos digitais e se estas seguirem as recomendações que se encontram

descritas no *Blue Book* [6].

No que concerne aos objectos digitais, estes pela sua natureza necessitam de uma plataforma informática para que os seres humanos os possam entender. Esta dependência de meios informáticos faz com que os objectos digitais possam deixar de ser inteligíveis ao ser humano caso esses meios falhem [1].

Vimos que existe uma cadeia de níveis de interpretação (físico, lógico, conceptual) e que, se essa cadeia de algum modo se romper, podemos não mais conseguir chegar ao objecto digital percebendo o que lá está, ou seja, não será possível extrair informação e conhecimento do objecto [4]. Para tentar garantir que a informação contida nos objectos digitais permaneça acessível e inteligível foram definidas uma série de estratégias. Algumas destas estratégias são, como vimos, a "preservação tecnológica", "emulação", "refrescamento", "migração", "normalização" e "encapsulamento" [1]. Mediante o estudo efectuado foi possível perceber que o nível no qual se pretende preservar o objecto digital irá determinar, em parte, a estratégia a utilizar. Contudo, a ideia principal no que diz respeito às estratégias para a preservação digital, é que muito provavelmente e na maioria das situações será necessário conjugar algumas dessas estratégias para que seja possível atingir uma preservação digital efectiva.

7.3 Preservação digital de Base de Dados Relacionais – Conclusões

Na tentativa de concepção de um protótipo, deparamo-nos com a necessidade de envolver diferentes estratégias. Para o nosso caso, que são as bases de dados relacionais, as estratégias de preservação digital adoptadas foram: "refrescamento", "migração" e "normalização". Como referimos, o refrescamento apenas consiste numa técnica para manter a informação em suportes físicos actuais. O cerne da questão prende-se com a migração e a normalização. Será conveniente que a informação se encontre normalizada porque assim, e em principio, bastará

conhecer um formato (o que é usado para normalizar). A migração terá que ser usada quer na conversão das bases de dados para o formato normalizado, quer em posteriores intervenções no sentido de manter a informação preservada.

Ao analisarmos esta ideia verificamos que o formato normalizado para o qual a base de dados será migrada assume especial relevância. A escolha recaiu sobre o DBML que como vimos é definido a partir do XML, o qual oferece garantias devido à sua neutralidade (independência de plataformas quer de hardware quer de software), devido à sua aceitação pela comunidade científica e devido também à sua legibilidade. O DBML graças à sua definição e às suas características tem a capacidade de acomodar bases de dados relacionais tanto ao nível dos dados como ao nível da estrutura. Das várias linguagens de anotação para bases de dados, não encontramos nenhuma com estas características [25]. Por norma, só os dados são objecto de anotação em linguagens XML. O DBML não se preocupa apenas com os dados mas também com a estrutura das bases de dados.

7.4 Perspectivas futuras

A preocupação com a perda de informação digital é deveras relevante, e tal facto é facilmente corroborado pelas políticas de cópias de segurança (*backups*) que as organizações actualmente implementam e pelo dinheiro investido nessas mesmas políticas. O problema é que num futuro a longo-prazo nada nos garante que, apesar da informação estar registada em suportes físicos, esta possa ser acedida no sentido da sua compreensão. Sem a envolvência tecnológica na qual a informação foi produzida, podemos correr o risco de perder a informação para sempre.

Por tudo isto, a preservação digital é fundamental para garantir um futuro acesso à informação legada. Ainda não existe uma solução capaz de dar resposta de forma definitiva a este problema, e não sabemos se isso alguma vez virá a ocorrer. Contudo o facto de esta temática ter-se vindo a

tornar objecto de estudo científico só poderá no futuro contribuir para a resolução do problema. Compete-nos divulgar e continuar a tratar o problema no sentido de consciencializar a Humanidade para esta problemática associada à passagem de conhecimento para gerações futuras.

8

Bibliografia e Referências

[1] 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.

[2] C. Webb, "Guidelines for the Preservation of Digital Heritage," United Nations Educational Scientific and Cultural Organization - Information Society Division, 2003.

[3] 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.

[4] Miguel Ferreira, “Introdução à preservação digital - Conceitos, estratégias e actuais consensos,” Escola de Engenharia da Universidade do Minho, Guimarães, Portugal, 2006.

[5] J. Ramalho e M. Ferreira, "Sistemas de suporte à Governação Electrónica," presented at Seminário Governação Digital, Amarante, Portugal, 2008

[6] Consultative Committee for Space Data Systems. “Reference Model for an Open Archival Information System (OAIS) – Blue Book,” National Aeronautics and Space Administration, Washington, 2002.

[7] M. d. L. Saramago, "Metadados para preservação digital e aplicação do modelo OAIS," presented at VIII Congresso da BAD, Estoril, Portugal, 2004.

[8] XML, “Extensible Markup Language”, in W3C - The World Wide Web Consortium [Online]. Available: http://www.w3.org/XML/

[9] M. Jacinto, G. Librelotto, J. Ramalho, P. Henriques, "Bidirectional Conversion between Documents and Relational Data Bases," Departamento de Informática, Universidade do Minho, Braga, Portugal, 2002

[10] Edgar Codd, "A Relational Model of Data for Large Shared Data Banks," in Communications of the ACM, 1970.

[11] Donald D. Chamberlin, Raymond F. Boyce, “SEQUEL: A Structured English Query Language,” IBM, 1970

[12] 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, R. Yost, “A History and Evaluation of System R,” IBM Research Laboratory, San Jose, California, 1981

[13] J. Ramalho, P. Henriques, “XML & XSL - Da Teoria à Prática,” FCA - Editora Informática, 2002

[14] John Cowan, “extensible Markup Language A Basic Overview,” a presentation on basic XML, 1998

[15] Wikipedia contributors, “History of programming languages,” in Wikipedia, The Free Encyclopedia, 2008. [Online]. Available: http://en.wikipedia.org/wiki/History_of_programming_languages

[16] M. Rans, “A History of Object-Oriented Programming Languages and their Impact on Program Design and Software Development,” [Online]. Available: http://jeffsutherland.com/papers/Rans/OOlanguages.pdf

[17] Gail M. Hodge, “Best Practices for Digital Archiving,” D-Lib Magazine, 2000

[18] H. Besser, "Digital Preservation of Moving Image Material?," The Journal of the Association of Moving Image Archivists, 2001.

[19] J. Ramalho, M. Ferreira “Relational Database Preservation through XML modelling, ” Extreme Markup Languages 2007, Montréal, Québec, 2007

[20] Claire Eager, “The State of Preservation Metadata Practices in North Carolina Repositories,” Chapel Hill, North Carolina, 2003

[21] D. Waters, "Good Archives Make Good Scholars: Reflections on Recent Steps Toward the Archiving of Digital Information,"

[22] B. F. Lavoie, "The Open Archival Information System Reference Model: Introductory Guide," Digital Preservation Coalition, Dublin, USA, Technology Watch Report Watch Series Report, 2004.

[23] Michael Day, “The OAIS Reference Model,” Digital Curation Centre UKOLN, University of Bath, 2006

[24] J. Ramalho, M. Ferreira, R. Castro, L. Faria, F. Barbedo, L. Corujo, "XML e Preservação Digital," *Dep. Informática, Universidade do Minho* e *Instituto dos Arquivos Nacionais, Torre do Tombo*

[25] Ronald Bourret, "XML and Databases, " Copyright 1999-2005 by Ronald Bourret. Last updated September, 2005

[26] Wikipedia contributors, "Database models," in Wikipedia, The Free Encyclopedia, 2008. [Online]. Available: http://en.wikipedia.org/wiki/Database_models