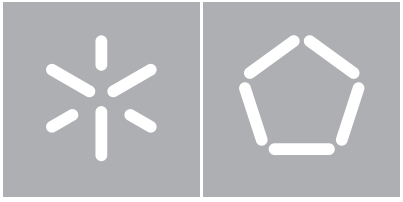
Universidade do Minho
Escola de Engenharia

Miguel da Silva Coutada

Database Preservation Toolkit

Outubro 2014



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Miguel da Silva Coutada

Database Preservation Toolkit

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
José Carlos Ramalho (DI - UM)

Outubro 2014

Anexo 3

DECLARAÇÃO

Nome
Miguel da Silva Coutada

Endereço electrónico: michaelcoutada@gmail.com Telefone: 919263361 / _____

Número do Bilhete de Identidade: 14012697

Título dissertação X/tese =
Ferramenta de Preservação de Bases de Dados

Orientador(es):
José Carlos Ramalho

Hélder Silva Ano de conclusão: 2014

Designação do Mestrado ou do Ramo de Conhecimento do Doutoramento:
Mestrado em Engenharia Informática

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

X 1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE/TRABALHO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, , MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

3. DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA TESE/TRABALHO

Universidade do Minho, 30/10/2014

Assinatura: Miguel da Silva Coutada

Agradecimentos

Gostaria de agradecer às seguintes pessoas pela ajuda, apoio e incentivo prestado no decorrer desta dissertação. Todas elas contribuíram de forma positiva para que a mesma fosse possível. As palavras poderão não o refletir da melhor forma, mas a todas essas pessoas, o meu "Muito obrigado" sincero.

À equipa da KEEP SOLUTIONS por me acolherem e por proporcionarem um ótimo ambiente de trabalho, tornando assim numa tarefa mais fácil a execução desta dissertação. Em particular, ao Professor Doutor José Carlos Ramalho pela sua orientação e conhecimento. De forma especial, ao Hélder Silva por todas as horas despendidas, pelo acompanhamento contínuo, incentivo e conhecimento que trouxe a esta dissertação e a mim.

Aos meus pais e irmã. Sem eles, todo este percurso, bem como esta dissertação (marco do final do mesmo), não seria possível.

À Cláudia.

Aos meus amigos e colegas por todos os momentos destes últimos anos.

Resumo

A preservação de sistemas de informação é um dos maiores desafios da preservação digital. Entre esses sistemas, encontram-se as bases de dados. Estas servem de sustento à maior parte dos sistemas de gestão de informação, apresentando, assim, um grande interesse no que diz respeito à sua preservação.

Se por um lado existe a necessidade de migrar as bases de dados para outras mais atuais, que vão aparecendo com o evoluir da tecnologia, por outro, existe também a necessidade de preservar a informação nelas contida durante um longo período de tempo, quer seja por questões legais, quer seja por questões de arquivo. Desta forma, essa informação deverá estar disponível independentemente do sistema de gestão da base de dados.

Nesta área, os produtos que existem para preservação de base de dados relacionais ainda são poucos - CHRONOS e SIARD são os principais. O primeiro é muitas das vezes inacessível devido ao preço que apresenta. O segundo apenas suporta funcionalidades básicas.

Assim, existe a oportunidade de explorar as principais qualidades e fraquezas das ferramentas existentes, de forma a melhorar a ferramenta de preservação de base de dados **db-preservation-toolkit**, componente extraída do projeto RODA.

Deste modo, esta ferramenta foi melhorada, quer pela adição de funcionalidades de forma a ser capaz de suportar um maior número de sistemas de gestão de bases de dados, quer pela adição do suporte ao formato SIARD, mas também pelo desenvolvimento de uma interface que possibilita a visualização e pesquisa de informação numa base de dados arquivada.

Abstract

The preservation of information systems is one of the biggest challenges of digital preservation. Among those systems we can find databases. Databases support the majority of the information management systems, showing themselves as a valuable resource to preserve.

If in one hand there is a need to migrate databases to newer ones that appear with technological evolution, on the other hand there is also the need to preserve the information they hold for a long time period, due to legal duties but also due to archival issues. That being said, that information must be available no matter the database management system where the information came from.

In this area, the existing products for relational database preservation are still scarce - CHRONOS and SIARD are the main ones. The first one is, in most of the cases, unreachable due to the associated costs. The second one only supports basic features.

Therefore there is the urge to explore the main features and limitations of the existing products in order to improve **db-preservation-toolkit**, an extracted component from the RODA project.

Therefore, this toolkit was improved by adding new features in order to support more database management systems, by adding support to the SIARD format, but also by the development of an interface that enables the possibility to visualize and search the information of an archived database.

Conteúdo

Lista de Figuras

Listagens

Lista de Tabelas

Glossário

character set

Descrição de character set.

command-line interface

Meio de interagir com um programa de computador onde o utilizador envia comandos para o programa na forma de linhas de texto (linhas de comando).

full-text

Full-text ou texto é simplesmente texto não estruturado.

multithreading

Multitreading é um modelo de execução e programação que permite que várias *threads* existam no contexto dum único processo.

open source

Quando um programa de computador é *open source* significa que o código do programa é de livre acesso ao público.

schema

Um *schema* é um esquema ou diagrama que descreve a estrutura de um tipo de dados específico.

script

Lista de comandos que são executados por um certo programa de computador ou por um motor de *scripting*.

servlet

Um *servlet* é uma classe da linguagem de programação Java usada para estender as capacidades de um servidor.

shell

Oferece uma interface ao utilizador para aceder aos serviços do sistema operativo.

snapshot

Refere-se a uma cópia feita para o disco num momento específico do tempo.

software

O mesmo que programa de computador. Termos relacionados como programas de *software*, aplicações, *scripts* pertencem todos na categoria de *software*.

web service

Forma de partilha da informação sem necessidade de copiar ou replicar ficheiros, uma vez que são usados protocolos de ligação e mensagens para invocação de serviços, um vocabulário e uma linguagem formal WSDL, que permitem às organizações descreverem, descobrirem e usarem serviços Web listados num diretório UDDI ou em diretórios de serviços Web.

metadata

Em geral "dados sobre os dados"; funcionalmente, "dados estruturados sobre dados".

REST

Representational state transfer é uma arquitetura sem estado para interagir com recursos via métodos HTTP.

Unicode

Um *stantard* internacional para sistemas de codificação de caracteres, correspondendo ao ISO 10646.

UTF

Unicode Transformation Format é um tipo de codificação Unicode de comprimento variável.

Acrónimos

AIP	Archival Information Package.
ANSI	American National Standards Institute.
API	Application Programming Interface.
ARELDA	ARchiving of ELctronic DAta.
BLOB	Binary Large OBject.
CLOB	Character Large OBject.
DBML	Database Markup Language.
DBMS	Database Management System.
DIP	Dissemination Information Package.
ISO	International Organization for Standards.
Java RMI	Java Remote Method Invocation.
JDBC	Java Database Connectivity.
LDAP	Lightweight Directory Access Protocol.
OAIS	Open Archival Information System.
ODBC	Open Database Connectivity.
PLANETS	Preservation and Long term Access via NETworked Services.
RODA	Repositório de Objetos Digitais Autênticos.
SFA	Arquivos Nacionais da Suíça.
SIARD	Software Independent Archiving of Relational Databases.
SIP	Submission Information Package.
SQL	Structured Query Language.

XML	eXtensible Markup Language.
XSD	XML Schema Definition.

Capítulo 1

Introdução

A obsolescência tecnológica da informação digital é maior do que noutros campos. Enquanto um livro com centenas de anos é praticamente igual aos de hoje em dia, o mesmo não acontece com a informação digital. Quando se fala deste assunto, fala-se de um conjunto variado e alargado de dispositivos e formatos, em que existem várias incompatibilidades de uns com os outros.

Muita da informação científica e cultural que será herdada pelas futuras gerações está armazenada digitalmente. Assim, a preservação digital, que se pode definir como o conjunto de métodos e técnicas para assegurar o acesso à informação digital a longo prazo [Fer06], é um tema da maior importância, no que diz respeito a assegurar que a informação será mantida para as gerações futuras. Neste contexto, e uma vez que servem de base ao armazenamento da maior parte dos sistemas de informação, surgem as bases de dados.

As bases de dados estão presentes desde sistemas de *data warehouse* a simples aplicações web. Os sistemas de informação da maior parte das organizações (governos, museus, galerias, bibliotecas, arquivos, etc) são suportados por bases de dados. Assim, e dada a grande importância de preservar a informação digital e esta estar, em muitos casos, armazenada em bases de dados, é essencial a preservação das mesmas.

Para o efeito de preservação de bases de dados, existem ferramentas como o CHRONOS e o Software Independent Archiving of Relational Databases (SIARD). Contudo, estas apresentam algumas lacunas. O CHRONOS, um produto comercial, devido ao seu custo não se apresenta na maior parte das vezes como uma ferramenta viável para preservação de bases de dados. O SIARD, por sua vez, desenvolvido pelos arquivos nacionais da Suíça, sendo referido como um formato aberto para preservação de bases de dados relacionais mas também como um produto de

software chamado SIARD Suite, apresenta apenas funcionalidades básicas.

A forma como estes produtos abordam a preservação de bases de dados consiste em guardar a informação da mesma num ficheiro de texto anotado (vários na prática, já que são separados para melhor organização e *performance*). Estes ficheiros, devido à sua composição textual (eXtensible Markup Language (XML) no caso do SIARD), terão a informação da base de dados, que será acessível a longo prazo, tornando-se assim independente de sistemas de gestão de bases de dados.

O formato destes ficheiros pode ser usado como uma forma de interoperabilidade, na medida em que permitem inserir a informação noutros sistemas de gestão de base de dados, como é demonstrado na figura 1.1.

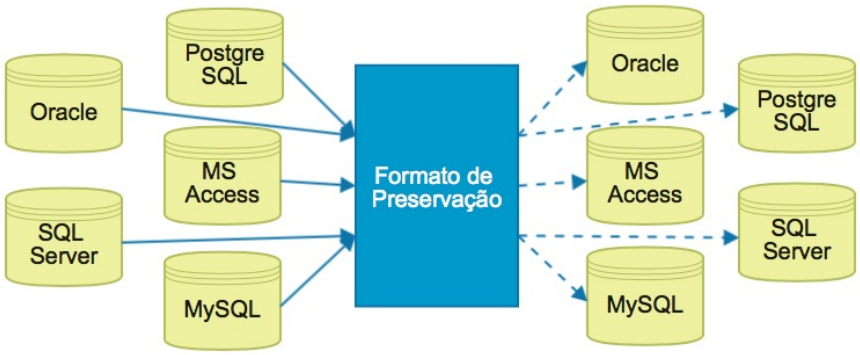


Figura 1.1: Formato de preservação como forma de interoperabilidade

Num outro contexto, não específico para preservação de bases de dados, mas sim para preservação digital em geral, encontra-se o Repositório de Objetos Digitais Autênticos (RODA). O RODA, nascido de um projeto português de investigação e desenvolvimento na área da preservação digital, promovido pela DGARQ com colaboração técnica da Universidade do Minho, é um repositório de preservação digital *open source* que fornece todas as funcionalidades descritas no modelo Open Archival Information System (OAIS).

Um dos seus componentes permite a ingestão e disseminação de base de dados em vários formatos, usando para isso um formato intermédio chamado Database Markup Language (DBML). Assim, à semelhança do SIARD, o RODA pode receber como Submission Information Package (SIP) uma base de dados, criando um pacote de arquivo - Archival Information Package (AIP) - que contém a base de dados em DBML e, pode, posteriormente, disseminar, sobre a forma de Dissemination Information Package (DIP), a base de dados para outro sistema de base de dados.

Desta forma, e dada a sua aplicabilidade para a preservação de bases de

dados, esta componente foi extraída do RODA, já que, por si só, é uma ferramenta útil e pode evoluir independentemente do projeto que lhe deu origem, tendo sido, assim, isolada e denominada como **db-preservation-toolkit**[1].

Existe, agora, um interesse em melhorar esta ferramenta, para que seja capaz de resolver alguns problemas dos produtos atualmente existentes, continuando por isso a ser uma ferramenta *open source* (superando o problema do CHRONOS, uma vez que este apresenta custos), e também adicionando funcionalidades como o suporte para mais sistemas de gestão de base de dados, pesquisa no ficheiro arquivado, entre outras, que colmatam alguns pontos não abordados nem pelo SIARD, nem pelo CHRONOS.

1.1 Motivação

Os sistemas de informação da maior parte das organizações são amplamente suportados por bases de dados. Assim, de forma a ser possível aceder à sua informação a longo-termo, a sua preservação é um assunto que merece atenção, havendo, por isso, a necessidade de usar ferramentas que facilitem esse processo.

Como [Lin13] refere, o ciclo de vida típico de uma base de dados pode ser descrito da seguinte forma:

- estado ativo, no qual a base de dados é acedida e modificada;
- estado de arquivo, no qual a base de dados não é alterada, mas continua ativa para ser acedida;
- estado de arquivo de longo termo, no qual a base de dados é mantida para retenção.

Deste modo, é possível verificar que parte significante da vida de uma base de dados é passada no estado de arquivo. Assim, torna-se visível o interesse da existência de uma ferramenta de preservação de base de dados.

O facto de haver, em muitos casos, uma necessidade de reter a informação da base de dados, assim como permitir a sua consulta devido a regulamentações legais, é outro aspeto que faz da preservação de base de dados um assunto importante, especialmente quando se tratam de instituições governamentais.

[1] `http://keeps.github.io/db-preservation-toolkit`

Também é importante referir que organizações como bibliotecas, museus, galerias e arquivos têm um grande interesse numa ferramenta que lhes permita o arquivo de base de dados.

Por outro lado, os sistemas de gestão de bases de dados em si, não fornecem mecanismos para o arquivo das mesmas, em parte por não haver um *standard* para o arquivo de base de dados. Apesar do formato SIARD ter sido adotado por algumas organizações como o formato de arquivo de base de dados, não pode ser considerado como um *standard*.

Como já referenciado, as principais ferramentas de preservação de base de dados apresentam certas lacunas. Assim sendo, providenciar funcionalidades que suprimam algumas dessas lacunas é um motivo forte pelo qual o melhoramento do **db-preservation-toolkit** se torna importante.

Deste modo, os seguintes fatores fazem com que exista um interesse relevante para que este projeto seja executado:

- ser gratuito, ao contrário do CHRONOS;
- possibilitar o acesso ao arquivo de um maior número de tipos de base de dados;
- permitir a compatibilidade entre formatos, como é o caso do formato SIARD;
- possibilitar facilmente a visualização e pesquisa de informação sobre uma base de dados arquivada.

1.2 Objetivos

O principal objetivo deste trabalho é melhorar a ferramenta de preservação de bases de dados, já existente, mas que ainda não está "madura" suficiente para o uso geral: **db-preservation-toolkit**. Desta forma, pode-se dizer que melhorar a ferramenta implica adicionar funcionalidades que superem as principais lacunas existentes nas aplicações do mesmo tipo, e torná-la, assim, numa ferramenta viável no que diz respeito à preservação de bases de dados.

Assim, adicionar suporte a um maior número de sistemas de gestão de base de dados ainda não suportados, suportar a integração com outros formatos, como é o caso do formato SIARD, e possuir uma interface que possibilite a visualização e pesquisa sobre a base de dados arquivada (ficheiro de texto), são objetivos inerentes à melhoria do **db-preservation-toolkit**.

1.3 Estrutura da dissertação

Nesta secção será descrita de forma resumida o que será abordado em cada um dos capítulos desta dissertação.

Assim, no capítulo 2 é abordado o estado da arte no que diz respeito a ferramentas de preservação de bases de dados existentes, bem como apresentados alguns conceitos que ajudam a melhor compreender a esta dissertação.

No capítulo 3 é explorada o estado em que se encontrava o **db-preservation-toolkit**, antes de se proceder a modificações desta ferramenta.

De seguida, no capítulo 4 são descritos tópicos relativos ao porquê da implementação do suporte ao formato SIARD no **db-preservation-toolkit**, bem como apresentadas algumas considerações sobre o formato SIARD. Neste capítulo, também o formato SIARD é descrito em detalhe, sendo, por fim, explicada a forma como foi implementado o seu suporte.

Já no capítulo 5, são discutidos os temas relativos ao serviço de visualização e pesquisa. Entre eles encontram-se o tópicos como o porquê, o modo de funcionamento, bem como algumas considerações sobre a implementação deste visualizador.

Por último, são tecidas algumas conclusões e apresentados alguns pontos que servem de referência para trabalho futuro.

Capítulo 2

Trabalho Relacionado

Primeiramente, serão apresentados, neste capítulo, alguns conceitos úteis à mais fácil perceção desta dissertação. Serão, em seguida, apresentados os trabalhos relacionados com a preservação de base de dados mais relevantes, de forma a entender as suas caraterísticas, assim como algumas das suas limitações, para que possam ser tomadas em conta, e serem depois usadas de forma a ser possível melhorar o **db-preservation-toolkit**. No final, são feitas algumas observações.

2.1 Conceitos chave

De seguida, são apresentados alguns conceitos úteis que melhor ajudam a entender esta dissertação.

2.1.1 Bases de Dados

Uma base de dados pode ser descrita como um "conjunto de dados organizados" [Wika], de tal forma que um computador consegue aceder facilmente à informação desejada. Os dados são uma coleção de partes distintas de informação, em particular de informação que foi "formatada" (organizada) de uma certa maneira para o uso em análises ou para fazer decisões.

Geralmente, uma base de dados pode ser vista como sendo uma coleção de "registos", em que cada um deles contém um ou vários "campos", isto é, partes de informação sobre uma entidade (pessoa, organização, produto, etc).

Foram desenvolvidos vários modelos de bases de dados, sendo alguns deles os modelos hierárquico, em rede, relacional e *flat*. Tipicamente, uma

base de dados possui um *schema* que é descrição do modelo, incluindo os tipos de entidades que estão presentes e as relações entre elas.

Deste modo, as bases de dados relacionais (o tipo mais comum de bases de dados) contêm tabelas, cada uma possuindo, geralmente, informações sobre uma entidade. Cada tabela contém vários registos, em que cada registo possui diversos campos. Assim, por exemplo, uma tabela de clientes armazena a informação de clientes, sendo que cada cliente é um registo, e cada registo contém vários campos (nome, data de nascimento, número de telefone, etc).

Um sistema de gestão de base de dados, também chamado Database Management System (DBMS), é uma aplicação especial que visa permitir interação com o utilizador e com a própria base de dados de forma a ser possível recolher e analisar informação.

Assim, um DBMS é um sistema feito para possibilitar a criação, a execução de *queries*, atualização e administração de bases de dados, sendo que uma base de dados não é, geralmente, portável entre diferentes DBMS's, contudo DBMS's diferentes podem interagir pelo uso de *standards* como o SQL, JDBC e ODBC.

2.1.2 SQL

O Structured Query Language (SQL) é uma linguagem de programação "desenhada para gerir a informação contida num sistema de gestão de base de dados relacionais" [Wikb]. Desta forma, O SQL permite a definição e manipulação de informação (inserção, atualização, eliminação, etc). O SQL tornou-se um *standard* do American National Standards Institute (ANSI) e do International Organization for Standards (ISO), em 1986 e em 1987, respetivamente. Contudo, o seu código não é completamente portável entre os diferentes sistemas de gestão de base de dados, devido, por exemplo, à adição de extensões, tornando o *standard* confuso.

Durante a sua existência, o SQL foi progredindo entre várias versões, importando destacar o SQL:92 e SQL:1999, uma vez que são as normas pelas quais se regem, respetivamente o CHRONOS e o SIARD.

2.1.3 JDBC e ODBC

O Java Database Connectivity (JDBC) é uma tecnologia da Oracle que funciona como uma API para o Java, definindo a forma como um cliente pode aceder a uma base de dados. Assim, o JDBC fornece métodos que

permitem executar *queries* e atualizar uma base de dados, sendo orientado a bases de dados relacionais.

O Open Database Connectivity (ODBC) tem a mesma função que o JDBC, visando assim, permitir o acesso a diferentes DBMS através das funções disponibilizadas. O ODBC foi desenvolvido pela Microsoft, existindo, hoje em dia, drivers disponíveis para a maior parte das plataformas e bases de dados. Contudo, pode existir a necessidade de instalar manualmente os drivers ODBC nas máquinas de cliente.

2.2 Projetos Relacionados

2.2.1 CHRONOS

O CHRONOS, produto comercial, desenvolvido pela CSP e emergente de uma pesquisa em conjunto com o departamento de ciências da computação da universidade de ciências aplicadas de Landshut, tem como objetivos principais a preservação, assim como o arquivo contínuo/parcial de bases de dados relacionais [Lin13].

Assim, no que diz respeito a cenários de preservação, o CHRONOS dá ênfase aos seguintes aspetos:

- **retiro de bases de dados**: transformação de bases de dados para um formato independente, percetibilidade do arquivo gerado, acesso à informação arquivada;
- **arquivo parcial/contínuo**: alterações de schema ao longo do tempo, retenção de informação.

Atualmente, o CHRONOS suporta o arquivo das seguintes bases de dados: Oracle, DB2, MS SQL e Informix [CSP].

No que diz respeito ao acesso à informação, o CHRONOS possui uma funcionalidade muito útil que permite executar *queries* compatíveis com SQL:92 sobre a informação arquivada. Apesar da informação estar armazenada fisicamente em ficheiros de texto, a *performance* da *query* pode ser comparável à de uma base de dados relacional [Lin13].

Em termos de elementos que são exportados de uma base de dados, o CHRONOS foca-se em exportar a informação e os principais tipos de dados. Assim, mostra-se na tabela 2.1 os tipos de suporte que o CHRONOS oferece em relação aos diferentes elementos aquando da exportação da base de dados para arquivo.

Exportados	**Não suportados**	**Não reimportados**
- tabelas - *views* - *indices* - *packages* - *procedures* - *functions* - *triggers* - *sequences* - *materialized views* - *scheduler* - *check constraints*	- *database links* - *jobs* - *queues* - gestão de permissões & utilizadores	- *triggers* - *procedures* - *views*

Tabela 2.1: Tipos de dados exportados/suportados pelo CHRONOS

Assim, na coluna **Exportados** encontram-se os elementos que o CHRONOS exporta, enquanto que na coluna **Não suportados** encontram-se os elementos que o CHRONOS não preserva/exporta. Na terceira coluna, **Não reimportados**, encontram-se os elementos que são exportados no processo de arquivo, mas que não podem ser reimportados de uma base de dados arquivada para um sistema de gestão de base de dados. Este aspeto pode ser interpretado como uma funcionalidade de segurança que visa evitar inconsistências de mapeamento, e também elimina da equação o que é específico de um sistema de gestão de base de dados: por exemplo, os *triggers* que não irão funcionar num sistema de gestão de base de dados diferente.

O CHRONOS permite interagir com o sistema de gestão de base de dados diretamente através de comandos da shell e scripts da base de dados.

Relativamente a mecanismos de retenção de informação, isto é, mecanismos que asseguram que certa informação é mantida e/ou apagada depois de um determinado período de tempo, o CHRONOS possui módulos para criar políticas de retenção de informação arquivada, assim como cumpre totalmente os requisitos de retenção legal de informação. Assim sendo, este produto dispõem de procedimentos que garantem que a retenção e eliminação de informação do ponto de vista legal é realizada.

Quanto aos controlos de acesso e gestão de utilizadores, isto é, as capacidades do produto oferecer funcionalidades de gestão de permissões e de utilizadores em cima do arquivo extraído da base de dados, o CHRONOS apresenta uma camada de gestão de acessos capaz de lidar com estes requisitos. Além disso, é possível a integração com sistemas de gestão central de utilizadores como o Lightweight Directory Access Protocol

(LDAP). O nível de granularidade oferecido para proteger informação sensível no arquivo pode ir até ao nível de colunas da base de dados.

O CHRONOS suporta o arquivo contínuo de informação. Deste modo, certa informação continua a ser usada e a estar presente no ambiente de produção, sendo sujeitas, ao longo do tempo, a alterações sintáticas e semânticas. Neste aspeto, modificações estruturais no *schema*, como é, por exemplo, o caso de adicionar mais colunas a uma tabela, são detetadas e tratadas automaticamente pelo CHRONOS. No caso de alterações semânticas, estas precisam de ser tratadas manualmente, já que não há forma de serem detetadas.

É possível interagir com o CHRONOS através das Application Programming Interface (API) disponibilizadas pelo mesmo: JDBC, Java Remote Method Invocation (Java RMI) e web services. Já a nível de interface gráfica, é possível satisfazer o processo de exportação de uma base de dados e a sua posterior reimportação num sistema de gestão de base de dados.

Em termos de escalabilidade e de *performance*, o CHRONOS faz uso de multithreading do Java, de forma a atingir melhores resultados.

No que diz respeito ao arquivo de base de dados gerado, este encontra-se divido em duas partes: a estrutura da informação e a informação em si. Assim, o CHRONOS armazena a estrutura da base de dados em XML, fornecendo o respetivo *schema* em XML Schema Definition (XSD). Por outro lado, a informação propriamente dita é guardada num ficheiro de texto delimitado.

2.2.2 SIARD

O SIARD, formato de representação de arquivos de bases de dados, mas também um software denominado de SIARD Suite, é propriedade dos Arquivos Nacionais da Suíça (SFA), e tem como objetivo, à semelhança do CHRONOS, a preservação de bases de dados relacionais [Lin13]. Contudo, foca-se apenas no cenário de "retiro de bases de dados". Assim, o cenário de "arquivo parcial/contínuo" não é contemplado pelo SIARD.

Como bases de dados possíveis de arquivar, o SIARD Suite suporta, neste momento, os sistemas de gestão Oracle, Microsoft SQL Server, MySQL e Microsoft Access [SFAc].

Contrariamente ao CHRONOS, não é possível efetuar pesquisas simples, nem complexas a partir de *queries* em cima da base de dados arquivada com o SIARD Suite. Assim sendo, é necessário reimportar o arquivo para um sistema de gestão de base de dados de forma a pesquisar o pre-

tendido.

No que toca a elementos exportados para o arquivo, o formato SIARD suporta apenas os elementos do SQL:1999 [Tho13]. Contudo, o formato SIARD concentra-se em preservar a informação principal e não código (*triggers*, *procedures*, etc). Assim, quando o SIARD Suite reimporta um arquivo para um sistema de gestão de base de dados, apenas as tabelas e a sua informação é restaurada [Lin13]. Deste modo, os outros elementos do SQL:1999 suportados são considerados, do ponto de vista do formato SIARD, como metadata, não sendo por isso restaurados já que poderiam causar problemas entre diferentes instâncias de bases de dados, e sendo guardados apenas para fins informativos.

Relativamente a questões de retenção de informação, o SIARD Suite não oferece nenhum tipo de apoio para retenção de informação ou sua eliminação.

O SIARD Suite não oferece nenhuma forma de controlo de acessos, isto é, de gestão de utilizadores e de permissões sobre a base de dados arquivada. Também não dispõe, na interface com o utilizador, de nenhuma forma de personalizar as vistas da informação tabular arquivada. Contudo, a informação que é recolhida e exportada da base de dados é determinada pela visibilidade e os direitos de acesso que o utilizador tem no sistema de gestão de base de dados. Assim, aquando da exportação, o SIARD Suite exporta todos os objetos "visíveis" na base de dados. Como já foi referido, o SIARD Suite não prevê o cenário de "arquivo parcial/contínuo". Contudo é possível, no caso do arquivo contínuo, usar o SIARD Suite para arquivar a versão mais recente da base de dados, mantendo assim um conjunto de snapshots com as várias versões da base de dados.

O SIARD Suite é um conjunto de ferramentas feitas em JAVA que, pelas interfaces JDBC, consegue atingir independência dos diferentes sistemas de gestão de base de dados. O SIARD Suite contém, assim, o **SIARD Edit**, uma ferramenta com interface visual, que permite importar e exportar bases de dados para arquivo e vice-versa, sendo possível visualizar metadata da base de dados e dos seus elementos, assim como visualizar a informação das tabelas. Contudo, não é possível efetuar pesquisas dos dados - informação contida nas tabelas. Além disso, o SIARD Suite oferece duas ferramentas command-line interface, o **SiardFromDb** e o **SiardToDb** que permitem, respetivamente, exportar da base de dados para arquivo e importar do arquivo para a base de dados. Contudo, é aconselhado o uso destas últimas para importação/exportação em vez do **SIARD Edit**, já que apresentam uma *performance* bastante melhor.

O SIARD Suite usa o ZIP 64 sem compressão para gerar o pacote que

contém a base de dados arquivada, encontrando-se, por isso, limitado pelo tamanho do ZIP 64 que é de aproximadamente de 16 Exabyte.

De forma semelhante ao CHRONOS, no formato SIARD também existe uma divisão entre a metadata e os dados. Desta forma, tanto a estrutura como a informação da base de dados são armazenadas em ficheiros XML distintos, acompanhados pelos respetivos schemas.

2.2.3 RODA e db-preservation-toolkit

O RODA, surgido de um projeto de investigação e desenvolvimento português na área da preservação digital, promovido pela DGARQ com colaboração técnica da Universidade do Minho, é um repositório de preservação digital open source.

Assim, o RODA tem como objetivo garantir o armazenamento de objetos digitais, o acesso contínuo aos mesmos, a gestão da sua metadata, e a preservação e autenticidade dos objetos digitais no contexto de arquivos digitais [Ram12b]. Desta forma, o RODA fornece todas as funcionalidades descritas no modelo OAIS.

Além de objetos digitais como imagens, áudio, vídeo, documentos de texto, o RODA permite também a preservação de bases de dados relacionais. Assim, é possível ingerir uma base de dados no RODA através de um SIP, contendo esse SIP, entre outros, a base de dados no formato DBML [RFFR07].

Para tal, foi criada uma componente que permite converter vários tipos de bases de dados para DBML e vice-versa, de forma a ser possível criar SIPs contendo a base de dados em DBML e também ser possível conseguir converter o DBML para SQL de forma a ser importado e visualizado através de um *hack* no *phpMyAdmin* [Ram12b].

Desta forma, estando esta componente criada, e tendo margem para crescer com vida própria, esta componente foi extraída do RODA, dando assim origem ao **db-preservation-toolkit**.

O **db-preservation-toolkit**, suporta, atualmente, MS SQL Server, PostgreSQL, MySQL, MS Access e Oracle (não testada). Estruturalmente, o DBML consiste num ficheiro contendo a metadata e informação da base de dados arquivada.

2.3 Discussão

Depois da análise dos projetos mais relevantes no âmbito da preservação de base de dados, assim como o estado inicial do **db-preservation-toolkit**, são expostos alguns aspetos interessantes que serão importantes no melhoramento e adição de novas funcionalidades ao **db-preservation-toolkit**.

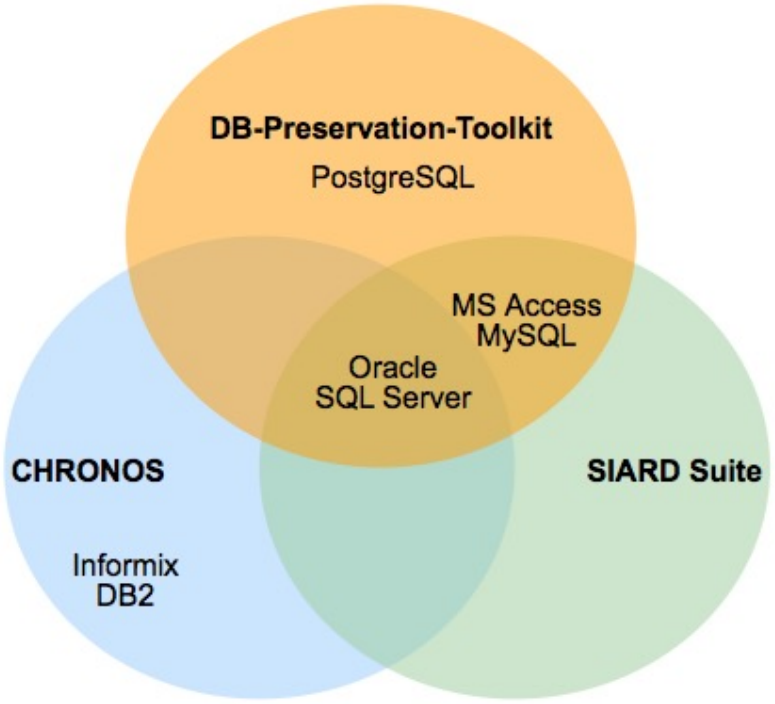


Figura 2.1: Sistemas de gestão de bases de dados suportados

No que toca a sistemas de gestão de base de dados suportados, podemos verificar que o **db-preservation-toolkit** (Oracle, MS SQL Server, MySQL, MSAccess e PostgreSQL) já suporta a maioria dos sistemas de gestão suportados pelo SIARD (Oracle, MS SQL Server, MySQL e MS Access) e CHRONOS (Oracle, DB2, MS SQL Server e Informix), sendo que em relação ao SIARD suporta os mesmos e ainda mais o PostgreSQL. Deste modo, podemos observar que o CHRONOS suporta o DB2 e Informix e o **db-preservation-toolkit** não. Assim, dado também o grau de popularidade[1] do DB2, este apresenta-se como um dos possíveis sistemas de gestão a adicionar suporte. A figura 2.1 ilustra os sistemas de gestão de base de dados suportados por cada produto.

[1] `http://db-engines.com/en/ranking`

Podemos constatar que o CHRONOS suporta o cenário de preservação contínua/parcial enquanto que o SIARD não demonstra preocupações nesse sentido. No âmbito da melhoria do **db-preservation-toolkit**, esta componente não parece ter uma relevância significativa porque em grande parte dos casos as bases de dados são arquivadas uma vez, no seu fim de vida. Contudo, não é uma funcionalidade que se possa descartar por completo.

Quanto à pesquisa de informação no arquivo da base de dados, é possível verificar que o CHRONOS possui mecanismos que asseguram uma pesquisa eficaz, enquanto que o SIARD não possui, de nenhuma forma a possibilidade de pesquisa sobre os dados arquivados (apenas permite a pesquisa de metadata). Para tal é necessário reimportar a base de dados arquivada para um sistema de gestão de base de dados, o que poderia ser evitável. Desta forma, a pesquisa de informação, sendo na forma de *queries* SQL (suportado pelo CHRONOS) ou numa forma mais simples, como pesquisa por texto, são componentes consideradas importantes, e sendo assim, merecem a devida atenção, pelo que será desenvolvido um visualizador que permite a pesquisa no arquivo de preservação de uma base de dados.

Relativamente aos elementos exportados, pode-se observar que tanto o CHRONOS como o SIARD exportam mais do que a informação tabular e estrutural, exportando, por isso, elementos como *triggers*, *views*, etc. Contudo, não quer dizer que os mesmos sejam reimportados do arquivo para um novo sistema de gestão de base de dados. Assim, é de notar que este tipo de elementos (os que são exportados) têm um papel importante no que diz respeito à forma como a base de dados é utilizada, podendo ser úteis a nível informativo. Na tabela 2.2 é possível ver a comparação de alguns aspetos das várias ferramentas, assim como algumas das funcionalidades a implementar no **db-preservation-toolkit**.

	CHRONOS	**SIARD**	**DB-Preservation-Toolkit**
Custos	pago	grátis	grátis
Suporte do formato SIARD	não	sim	a implementar
Suporte MySQL	sim	sim	sim
Suporte PostgreSQL	não	não	sim
Suporte SQL Server	sim	sim	sim
Suporte Oracle	sim	sim	sim (exportação não testada)
Suport MS Access	não	sim	sim
Suporte DB2	sim	não	a implementar
Suporte Informix	sim	não	não
Pesquisa de dados	sim	não	a implementar
Gestão de permissões e utilizadores	sim	não	não

Tabela 2.2: Algumas funcionalidades/caraterísticas das ferramentas de preservação

Capítulo 3

Descrição do db-preservation-toolkit v1.0

De forma a melhor compreender o processo de implementação do suporte ao formato SIARD (no capítulo 4), é primeiro necessário saber como funciona e está organizado o **db-preservation-toolkit**.

Assim, é neste capítulo descrita a forma como funcionava o **db-preservation-toolkit**, bem como descrito o DBML em mais detelhe e o modelo de dados intermédio do **db-preservation-toolkit**.

3.1 Funcionamento do db-preservation-toolkit v1.0

O **db-preservation-toolkit**, na sua génese é uma aplicação orientada ao evento, possuindo, deste modo, módulos de importação, assim como módulos de exportação.

Desta forma, os módulos de importação implementam a interface `DatabaseImportModule`. Esta interface contém um único método a ser implementado, denominado por `getDatabase`, como é possível observar pelo extrato de código 3.1.

```
public interface DatabaseImportModule {
    public void getDatabase(DatabaseHandler
        ↪ databaseHandler)
            throws ModuleException, UnknownTypeException,
            InvalidDataException;
}
```

Listagem 3.1: `Classe DatabaseImportModule`

Este método recebe um `DatabaseHandler`, interface que os módulos de exportação implementam. Esta interface define que os métodos seguintes devem ser implementados:

- `initDatabase()`
- `handleStructure(DatabaseStructure structure)`
- `handleDataOpenTable(String tableId)`
- `handleDataCloseTable(String tableId)`
- `handleDataRow(Row row)`
- `finishDatabase()`

Assim, os módulos de exportação implementam a interface `DatabaseHandler`, sendo esta usada pelo método `getDatabase` como parâmetro, de modo a que este método possa aceder aos métodos do `DatabaseHandler`, e assim, dada a informação importada, esta possa ser passada ao método respetivo do `DatabaseHandler` para ser tratada.

Em linhas gerais o comportamento do método `getDatabase` pode ser descrito pelo excerto de código 3.2.

```
public void getDatabase(DatabaseHandler handler) {
    handler.initDatabase();
    handler.handleStructure(getDatabaseStructure());
    for (Table table : getDatabaseStructure().getTables()
        ↪ ) {
        ResultSet tableRawData = getTableRawData(table.
            ↪ getId());
        handler.handleDataOpenTable(table.getId());
        while (tableRawData.next()) {
            handler.handleDataRow(convertRawToRow(
                ↪ tableRawData, table));
        }
        handler.handleDataCloseTable(table.getId());
    }
    handler.finishDatabase();
}
```

Listagem 3.2: Excerto do método `getDatabase`

O **db-preservation-toolkit** v1.0 suporta vários sistemas de gestão de bases de dados relacionais, refletindo-se isso nos vários módulos de importação e exportação que possui. Uma parte considerável desses módulos acedem aos sistemas de gestão de bases de dados relacionais via JDBC, de modo a suportar a importação/exportação de sistemas de gestão de

bases de dados como MySQL, PostgreSQL, SQLServer. Além disso, é possível encontrar módulos de importação/exportação que fazem uso de ODBC, módulos que permitem o suporte de MsAccess, e também módulos que permitem a exportação de bases de dados para "dumps" SQL (com diferentes *flavors* inclusive).

Contudo, em termos de preservação de bases de dados, no sentido em que se armazena a informação de uma base de dados independentemente do sistema de gestão de bases de dados e da sua versão, é necessária outra forma de preservação de bases de dados que não a "conversão" entre bases de dados, isto é, a transferência de informação de uma base de dados num sistema de bases de dados para outra base de dados noutro sistema de gestão de bases de dados. Para esse efeito, existe o DBML que permite arquivar a informação de uma base de dados em formato textual e de uma forma neutra, independente do sistema de gestão de bases de dados. Assim, existe também um módulo de importação/exportação para DBML, permitindo a preservação a longo-termo de uma base de dados. Em 3.2 será falado em mais detalhe o que é o DBML.

O **db-preservation-toolkit** como sugerido em cima, possuí módulos de importação independentes dos módulos de exportação. Deste modo, não existe, por exemplo, um módulo de conversão de SQLServer para MySQL, nem um módulo de SQLServer para PostgreSQL. Existe sim um módulo de importação de SQLServer, um módulo de exportação para MySQL, assim como um módulo de exportação para PostgreSQL, dado este exemplo.

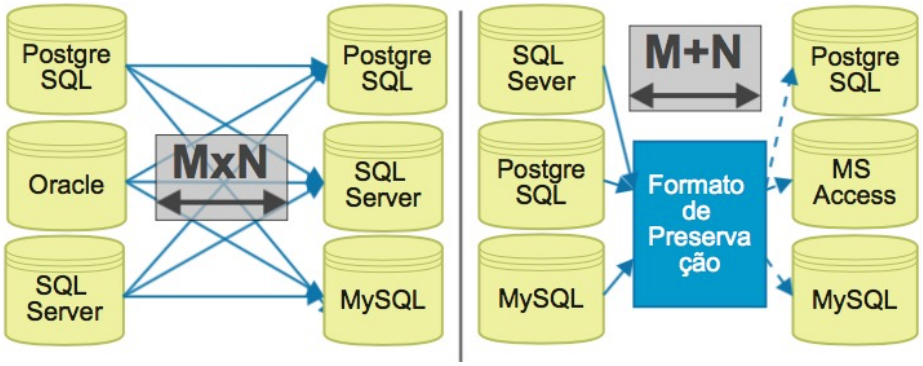


Figura 3.1: Número de módulos necessários

Desta forma, dados M *inputs* e N *outputs* de dados, não são necessários M*N módulos de "conversão", mas sim um total de M+N módulos [Ram12b]. Na imagem 3.1 pretende-se ilustrar tal situação. De forma a que tal seja possível, existe a necessidade de um modelo de dados que armazene a informação da base de dados importada para que possa poste-

riormente ser exportada. Isto é, é preciso um modelo de dados intermédio que armazene a informação a exportar.

Tal modelo de dados está presente no **db-preservation-toolkit**, tentando representar a informação importada de forma neutra, para que a mesma seja depois "moldável" para a exportação em diferentes destinos de dados (diferentes sistemas de gestão de bases de dados, "dumps" SQL, formatos de preservação de bases de dados, etc). Este modelo de dados intermédio ganha vida aquando a execução do **db-preservation-toolkit** e da tradução de uma fonte de dados para outro destino de dados (poderá ser de um sistema de gestão de base de dados para outro, mas também de um sistema de gestão de base de dados para DBML, ou para "dumps" SQL, entre outros). Isto é, este modelo de dados intermédio existe em memória central na execução do **db-preservation-toolkit**.

Analogamente, esta situação pode ser encontrada também na preservação de bases de dados a longo-termo propriamente dita, ou seja, quando uma base de dados é arquivada em formato textual (para DBML, por exemplo). O que se passa aquando da "conversão" de uma fonte de dados para outra, organizado por um modelo intermédio, que acontece em memória central, passa-se também de forma física e duradoura, quando se arquiva uma base de dados num formato de preservação de bases de dados, como é o caso do DBML. Essa base de dados arquivada em DBML, poderá posteriormente ser importada de forma a ser recuperada noutro sistema de gestão de base de dados diferente.

Deste modo, explicam-se as semelhanças entre o modelo de dados intermédio do **db-preservation-toolkit**, que será explicado em 3.3, e o DBML, único formato de preservação de dados suportado no **db-preservation-toolkit** v1.0.

3.2 DBML - Database Markup Language

O DBML é uma representação em XML de bases de dados relacionais com o objetivo da preservação das mesmas. Este formato de preservação está organizado de forma a representar a estrutura e a informação primária apenas num único ficheiro XML [JLRH02]. Como já referido, este formato é usado no projeto RODA.

É normal as bases de dados terem uma quantidade bastante acentuada de informação, fazendo com que a preservação de bases de dados usando DBML gerem ficheiros XML bastante grandes. Deste modo, armazenar informação num único ficheiro XML traz problemas no que diz respeito ao processamento e eficiência de pesquisa em ficheiros XML volumo-

sos. Assim, dado que um dos objetivos deste trabalho era resolver este problema através da segmentação da informação em vários ficheiros, tal é alacançado com a adiação do suporte ao formato SIARD.

O formato de preservação de bases de dados DBML, separa a estrutura da informação primária. Para tal, existem dois elementos XML principais: o elemento <structure> e o elemento <data>. Assim, o elemento <structure> contém a metadata para as tabelas, colunas e chaves de uma base de dados, fazendo uso de elementos e atributos XML para definir essa mesma metadata [RFFR07]. O exemplo 3.3 pretende clarificar a organização de um elemento <structure>.

```
<structure>
    <table id="departments" name="departments">
        <columns>
            <column id="departments.dept_no" name="
                ↪ dept_no" nillable="false" description="
                ↪ ">
                <type originalTypeName="CHAR">
                    <simpleTypeString length="4"
                        ↪ variableLegth="false"/>
                </type>
            </column>
            <column id="departments.dept_name" name="
                ↪ dept_name" nillable="false" description
                ↪ ="">
                <type originalTypeName="VARCHAR">
                    <simpleTypeString length="40"
                        ↪ variableLegth="true"/>
                </type>
            </column>
        </columns>
        <keys>
            <pkey type="SIMPLE">
                <field name="dept_no"/>
            </pkey>
        </keys>
    </table>
    ...
</structure>
```

Listagem 3.3: Excerto DBML do elemento <structure>

Por outro lado, o elemento <data> define a informação primária, isto é, a informação tabular, como é mostrado no excerto 3.4.

```
<data>
    <tableData id="departments">
        <row id="1">
```

```
            <cell id="departments.dept_no.1">
                <s>d009</s>
            </cell>
            <cell id="departments.dept_name.1">
                <s>Customer Service</s>
            </cell>
        </row>
        <row id="2">
            <cell id="departments.dept_no.2">
                <s>d005</s>
            </cell>
            <cell id="departments.dept_name.2">
                <s>Development</s>
            </cell>
        </row>
        ...
    </tableData>
    ...
</data>
```

Listagem 3.4: Excerto DBML do elemento <data>

É possível verificar no apêndice A um exemplo mais extenso de um ficheiro DBML. Por sua vez, no apêndice B a definição do *schema* do DBML.

3.3 Modelo de dados intermédio do db-preservation-toolkit v1.0

Do mesmo modo que o DBML, também o modelo de dados intermédio do **db-preservation-toolkit** v1.0 está organizado de forma a representar tanto a componente estrutural, como a componente que contém os dados primários de uma base de dados.

3.3.1 Estrutura

Assim, esta representação do modelo de dados ganha forma através de classes Java. Estas classes, depois na forma de objetos, são responsáveis por conter a informação necessária a uma conversão entre diferentes fontes de origem e destino de dados. Para representar tal modelo de dados usam-se as seguintes classes:

- `DatabaseStructure`
- `TableStructure`

- ColumnStructure
- PrimaryKey
- ForeignKey

Na classe DatabaseStructure é possível encontrar uma extensa lista de variáveis de instância. Estas apresentam-se também no DBML na forma de atributos do elemento <db>. Servem de exemplo variáveis como name (que representa o nome da base de dados) e productName (o nome do sistema de gestão de bases de dados). Além destas variáveis (não apenas as de exemplo), apresenta-se também a variável tables que representa uma lista de TableStructure, ou seja, representa as tabelas presentes numa base de dados.

Por sua vez, a classe TableStructure apresenta variáveis de instância como id, name, etc, como como é possível observar, de forma análoga no excerto de DBML 3.3, na forma de atributos do elemento <table>. Também se encontram presentes as variáveis columns e foreignKeys que representam listas de colunas e chaves estrangeiras, respetivamente, presentes numa dada tabela. Além destas variáveis, existe a variável primaryKey que representa a chave primária de uma tabela. Tanto as colunas como a chave primária de uma tabela, podem, analogamente, ser vistas no excerto de DBML 3.3: as colunas estão representadas sob o elemento columns, enquanto que a chave primária sob o elemento keys (elemento este que também contém as chaves estrangeiras). É ilustrado no excerto 3.5 as variáveis de instância da classe TableStructure.

```
public class TableStructure {
    private String id;
    private String name;
    private String description;
    private List<ColumnStructure> columns;
    private List<ForeignKey> foreignKeys;
    private PrimaryKey primaryKey;
    ...
    // construtores
}
```

Listagem 3.5: Variáveis de instância da classe TableStructure

Já a classe ColumnStructure, além das variáveis id, name e nillable (que diz se dada coluna pode ser nula ou não), apresenta a variável type, que guarda a informação do tipo de dados de uma coluna. Esta variável type é do tipo Type, uma classe com várias subclasses que pretendem acomodar, de forma neutra, os diferentes tipos de dados existentes nas colunas presentes em tabelas de diferentes sistemas de gestão de bases

de dados. No excerto 3.6 é possível observar as variáveis de instância da classe `ColumnStructure`.

```
public class ColumnStructure {
    private String id;
    private String name;
    private Type type;
    private Boolean nillable;
    private String description;
    ...
    // construtores
}
```

Listagem 3.6: Variáveis de instância da classe `ColumnStructure`

Para além das três classes já mencionadas, as classes `PrimaryKey` e `ForeignKey` representam, respetivamente, uma chave primária e uma chave estrangeira.

Como referido acima, cada coluna possui um tipo de dados, representado no modelo de dados intermédio pela classe `Type`. Esta classe como variáveis de instância apresenta apenas a variável `originalTypeName` (o nome do tipo de dados original) e `description` (descrição). Contudo, apresenta as seguintes subclasses:

- `SimpleTypeBinary`
- `SimpleTypeBoolean`
- `SimpleTypeDateTime`
- `SimpleTypeEnumeration`
- `SimpleTypeInterval`
- `SimpleTypeNumericApproximate`
- `SimpleTypeNumericExact`
- `SimpleTypeString`
- `ComposedTypeArray`
- `ComposedTypeStructure`

Estas classes pretendem reter a informação de cada um dos tipos de dados presentes nas diferentes colunas, de uma forma independente dos sistemas de gestão de bases de dados.

Assim, à exceção das classes `SimpleTypeEnumeration` e `SimpleTypeInterval` (não usadas), e `ComposedTypeArray` e `ComposedTypeStructure` (suporte a tipos de dados correspondentes

não implementado), todas as outras classes referidas em cima guardam a informação sobre o tipo de dados de uma coluna. Isto é, sejam, por exemplo, a precisão e a escala para um tipo de dados numérico exato (`SimpleTypeNumericExact`), ou o comprimento, e se esse mesmo comprimento é variável no caso de tipos de dados de sequências de carateres (`SimpleTypeString`).

Deste modo, o **db-preservation-toolkit** faz uso do método `getType` de forma a mapear um certo tipo de dados de uma coluna, com um certo tamanho e um certo número de dígitos decimais para uma destas classes (`SimpleTypeBinary`, `SimpleTypeBoolean`, etc). Este método encontra-se na classe `JDBCImportModule`, mas é também usado na importação de bases de dados que se acedam via ODBC, já que o módulo ODBC usa a ponte JDBC-ODBC. Isto é, a interação com sistemas de gestão de bases de dados que se liguem via ODBC é feita usando a API do JDBC. No caso do DBML, não é necessário o uso deste método, uma vez que o tipo de dados de certa coluna está presente no ficheiro DBML, na forma de elemento XML como mostra 3.3 (no DBML, também os tipos de dados são análogos aos do modelo de dados intermédio do **db-preservation-toolkit**).

Tal mapeamento, usando o método `getType`, ocorre de acordo com o tipo de dados de cada coluna, representado em Java por um inteiro, ou seja, o inteiro correspondente à constante presente em `java.sql.Types`. Cada uma destes valores representados por exemplo, por constantes do tipo `java.sql.Types.BIGINT`, `java.sql.Types.VARCHAR` são da responsabilidade das *drivers* JDBC, que fazem o mapeamento do sistema de gestão de bases de dados para um inteiro, na forma de `java.sql.Types`. No excerto 3.7 pretende-se demonstrar a forma como é feito o mapeamento do tipo de dados de cada coluna para os tipos de dados presentes no modelo de dados intermédio do **db-preservation-toolkit**.

```
protected Type getType(int dataType, String typeName, int
    ↪ columnSize,
        int decimalDigits, int numPrecRadix) throws
            ↪ UnknownTypeException {
    Type type;
    switch (dataType) {
    case Types.BIGINT:
        type = new SimpleTypeNumericExact(Integer.valueOf
            ↪ (columnSize),
                Integer.valueOf(decimalDigits));
        break;
    case Types.BINARY:
        type = new SimpleTypeBinary();
```

```
            break;
        case Types.BIT:
            type = new SimpleTypeBoolean();
            break;
        ...
        }
        ...
}
```

Listagem 3.7: Excerto do método `getType`

3.3.2 Dados primários

No que diz respeito aos dados primários no **db-preservation-toolkit**, o modelo de dados intermédio faz uso da classe `Row` para guardar a informação de cada linha da tabela a ser processada. Tal processamento é feito linha a linha, não sendo armazenada toda a informação de uma tabela, uma vez que a mesma poderá ser de dimensões bastante elevadas.

De forma a conter a informação de cada linha de uma tabela, a classe `Row` tem presente uma lista de células, que representam a informação de certa linha para cada uma das colunas da mesma tabela. Estas células podem ser células simples (`SimpleCell`) guardando a informação na forma de `String` Java, ou células binárias (`BinaryCell`).

Capítulo 4

Suporte ao formato SIARD

Um dos objetivos desta dissertação é que seja possível ao **db-preservation-toolkit** preservar bases de dados no formato SIARD. Assim, neste capítulo, é explicado o porquê (em 4.1) da sua implementação no **db-preservation-toolkit**, algumas considerações sobre o formato SIARD (em 4.2), e também é descrito o formato SIARD em mais detalhe (em 4.3).

Além disso, é explicada a forma como o suporte ao formato SIARD foi implementado (em 4.4) no **db-preservation-toolkit**, bem como algumas decisões que tiveram de ser tomadas de forma a acomodar este novo formato.

4.1 Porquê?

O SIARD (formato e aplicação SIARD Suite) foi criado como parte do projeto ARchiving of ELctronic DAta (ARELDA) para preservação digital dos SFA. O formato SIARD é uma descrição normativa do formato dos ficheiros usados para a preservação a longo prazo de bases de dados relacionais, sendo o formato de dados que sustenta o SIARD Suite [SFA11].

O formato SIARD é uma norma não proprietária, aberta e publicada, baseada em *standards* abertos, como por exemplo o Unicode, XML, SQL1999 e o ZIP. Ao usar *standards* aceites internacionalmente, pretende garantir a preservação e acesso ao longo termo de bases de dados relacionais.

O formato SIARD é utilizado pelos SFA, assim como várias entidades governamentais suíças. Em maio de 2008 foi aceite como formato oficial do projeto europeu Preservation and Long term Access via NETworked

Services (PLANETS)[1] para preservação de bases de dados relacionais.

Além disso, no início de 2013, o formato SIARD foi adotado como um *standard* eCH (eCH-0165: *Spécification de format SIARD*)[2] [SFAa]. Os *standards* eCH definem práticas para aplicações e para os seus resultados, como são, por exemplo, os casos de definições de formatos ou *standards* processuais. O objetivo destes *standards* é unificar e facilitar a colaboração eletrónica entre autoridades, assim como autoridades e organizações, instituições educacionais e de investigação, firmas e organizações privadas.

O uso do formato SIARD faz-se sentir em vários países do mundo, já que, para o efeito de preservação de bases de dados, o SIARD Suite, disponibilizado de forma gratuita, ajuda a impulsionar o uso deste formato. Contudo, o formato SIARD pode ser usado independentemente do SIARD Suite. Na imagem 4.1 é possível observar o número de licenças emitidas para o uso do SIARD Suite até à data de 01/02/2014.

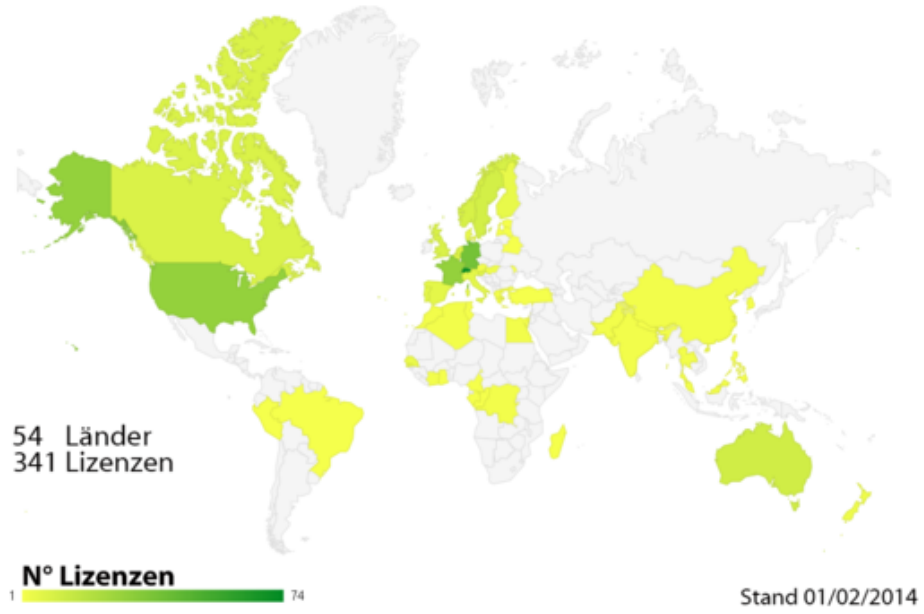


Figura 4.1: Mapa de licenças emitidas para o SIARD Suite (54 países, 341 licenças)

Assim, dada a sua ampla utilização para a preservação de bases de dados, faz sentido dar suporte ao formato SIARD no **db-preservation-toolkit**.

[1] `http://www.planets-project.eu`

[2] `http://www.ech.ch/vechweb/page?p=dossier&documentNumber=eCH-0165&documentVersion=1.0`

4.2 Considerações sobre o formato SIARD

4.2.1 Uso de standards

De forma a garantir a interoperabilidade do conteúdo das bases de dados a longo termo, o formato SIARD é baseado na sua essência nos standards ISO seguintes: XML e SQL:1999 [SFAb].

Todo o conteúdo das bases de dados é guardado num conjunto de ficheiros XML. A única exceção acontece para a informação do tipo de dados Binary Large OBject (BLOB) ou Character Large OBject (CLOB), em que tal informação é armazenada em ficheiros binários separados, mas referenciados nos ficheiros XML.

4.2.2 Bases de dados como documentos

Uma base de dados, para arquivo, é tratada pelo SIARD como sendo apenas um único documento. Assim, uma base de dados arquivada no formato SIARD é armazenada como um ficheiro único, sendo que este ficheiro é um arquivo ZIP que contém os ficheiros XML e binários acima mencionados, numa estrutura de pastas específica. Desta forma, bases de dados diferentes são arquivadas em ficheiros SIARD diferentes.

4.2.3 Caracteres

No geral, muita da informação digital é guardada num *character set* Unicode (*encoding*). Durante o processo de extração de uma base de dados que suporte outros *character sets*, o mapeamento para os carateres Unicode é efetuado. Assim, geralmente, o SIARD Suite transforma os tipos de dados de *strings* de carateres nacionais (NCHAR, NVARCHAR, NCLOB) em tipos de dados não nacionais (CHAR, VARCHAR, CLOB).

Esta convenção é bem suportada pelo XML, independentemente do ficheiro XML ser guardado no formato UTF-8 ou UTF-16.

Carateres com significado especial para o XML são substituídos por entidades referenciais nos arquivos SIARD. Além disso, carateres que não podem ser representados no Unicode, isto é, os carateres de controlo 0-8, 14-31, 127-159 são substituídos usando um \ como caráter de escape, ficando na forma de \u00<xx> no XML. Também o caráter \ e carateres de espaço são substituídos usando esta mesma regra. Na tabela 4.1 sumarizam-se as substituições de carateres que acontecem no formato SIARD [SFA11].

Carateres originais	Carateres no formato SIARD
0-8	\u0000 - \u0008
14-31	\u000E - \u001F
32	\u0020, para vários espaços
&	&amp;
<	<
\	\u005C
127-159	\u007F - \u009F

Tabela 4.1: Substituições de carateres no formato SIARD

4.3 Formato SIARD

De forma a melhor compreender a estrutura do formato SIARD, são de seguida descritos alguns conceitos sobre bases de dados relacionais.

4.3.1 Conceitos de bases de dados relacionais

Uma **base de dados** consiste, geralmente, num ou mais ***schemas***, assim como direitos de acesso para utilizadores e para *roles* em determinadas partes de uma base de dados. Assim, no SQL:1999 os **utilizadores** e os ***roles*** podem ter **privilégios** (autorizações).

Os ***schemas*** contêm **tabelas**, ***views*** e ***routines***.

As **tabelas** podem ser definidas como sendo um conjunto de campos (**colunas**) com um nome e um tipo de dados, e por um conjunto de registos que contêm os **dados primários**. Além disso, uma tabela pode conter uma **chave primária**, assim como **chaves estrangeiras**, de modo a garantir integridade referencial, **chaves candidatas** que servem para identificar unicamente um registo numa tabela e ***check constraints*** que servem para garantir consistência dos dados. Também podem ser adicionados ***triggers*** a uma tabela.

Pode-se dizer que as ***views*** são *queries* armazenadas na base de dados. O resultado de uma *query* é uma tabela que contém também registos, contudo não possui *check constraints*.

As ***SQL routines***, também conhecidas como *stored procedures*, neste contexto, são importantes para compreender as *queries* das *views*, já que as *SQL routines* podem ser invocadas nestas mesmas *queries* [GP99].

Assim, uma base de dados relacional é um conjunto de objetos estruturados de uma base de dados (*schemas*, *views*, etc), bem como o conteúdo

das tabelas (dados primários).

4.3.2 Estrutura de um arquivo SIARD

Uma base de dados relacional arquivada no formato SIARD é composta por duas partes: metadata, que descreve a estrutura da base de dados arquivada, e dados primários, que representam o conteúdo das tabelas. Além disso, a metadata fornece informação sobre onde encontrar certos dados primários no arquivo SIARD.

A metadata e dados primários da base de dados são armazenados conjuntamente num ficheiro ZIP não comprimido, sendo *.siard* a extensão do nome de ficheiro. Os dados são guardados na pasta *content* e a metadata na pasta *metadata* [Ber09].

A imagem 4.2 pretende clarificar a estrutura de um arquivo SIARD.

```
content/
    schema1/
        table1/
            table.xsd
            table.xml
            lob1/
                record1.txt (or record1.bin)
                record2.txt (or record2.bin)
            lob2/
                record1.txt (or record1.bin)
        table2/
            table.xsd
            table.xml
        ...
    schema2/
    ...
header/
    metadata.xsd
    metadata.xml
```

Figura 4.2: Estrutura exemplo de um arquivo SIARD

4.3.3 Metadata no arquivo SIARD

A metadata de um arquivo SIARD guarda a estrutura de uma base de dados arquivada. Toda a metadata é concentra num único ficheiro *meta-*

data.xml na pasta *header*.

De seguida, a título de exemplo, será descrito na tabela 4.2 o conteúdo da metadata ao nível do elemnto <schema>, guardada pelo formato SIARD. Um *Sim* na coluna *Opt.* indica que aquele item é opcional [SFA11].

Identificadores	**Opcional**	**Descrição**
name	não	nome do *schema*
folder	não	nome da pasta do *schema*
description	sim	descrição do *schema*
tables	não	lista de tabelas no *schema*
views	sim	lista de views no *schema*
routines	sim	lista de *routines* no schema

Tabela 4.2: Descrição da metadata do nível `schema`

É possível analisar no apêndice C um exemplo de um ficheiro *metadata.xml*, enquanto que no apêndice D é possível visualizar a definição do *schema* XML de um ficheiro *metadata.xml*.

4.3.4 Dados primários no arquivo SIARD

Os dados primários, ou seja, a informação tabular, pode ser encontrada na pasta *content* na raiz do documento do arquivo SIARD. O formato SIARD define que se a pasta *content* estiver vazia, então tratar-se-á de um arquivo SIARD vazio, que contém apenas as definições da metadata que descrevem a estrutura da base de dados.

Os dados primários de cada tabela são armazenados no arquivo SIARD na pasta *content*, na sub-pasta do *schema* à qual a tabela pertence. O SIARD Suite gera os nomes `schema1`, `schema2`, etc automaticamente para as pastas de *schemas*, e `table1`, `table2`, etc para pastas de tabelas [SFA11].

A informação tabular (dados primários) é armazenada num ficheiro XML chamado *table<N>.xml*, em que <N> é o número da tabela. Para cada tabela, é gerado um ficheiro XSD denominado por *table<N>.xsd*, que define o formato XML de armazenamento (xs:string, xs:decimal) da informação tabular.

Deste modo, uma tabela é armazenada como sendo uma sequência de linhas, que contêm uma sequência de colunas. O nome da *tag* da tabela é `table`, da linha é `row` e das colunas é `c1`, `c2`, etc.

Quando uma tabela contém informação dos tipos de dados CLOB ou

BLOB e o tamanho de dada linha nessa coluna é maior do que 4000 caracteres e/ou 2000 bytes, são criados ficheiros únicos para cada linha nestas condições, de forma a acomodar a informação nela contida, sendo, em vez dos dados, criada uma referência externa para o ficheiro de texto/binário.

O SIARD Suite gera automaticamente as pastas com nomes `lob1`, `lob2` para cada coluna correspondente que necessite de criar ficheiros dado as condições acima referidas. Por sua vez, os ficheiros que representam a informação de uma linha numa dada coluna são chamados de `record1.txt`, `record2.txt`, etc ou `record1.bin`, `record2.bin`, etc, dependendo do tipo de dados da coluna.

No apêndice E é possível visualizar um exemplo de um ficheiro que armazena os dados primários de uma tabela.

4.4 Implementação do suporte ao formato SIARD

Como é possível verificar, existem diferenças entre o formato DBML (3.2) e SIARD (4.3), em termos de estrutura e de informação que capturam. É de referir, como já descrito, que o funcionamento e o modelo de dados intermédio do **db-preservation-toolkit** estavam intimamente ligados ao formato DBML.

Dado que o formato SIARD acomoda diferentes propriedades de um sistema de gestão de bases de dados quando comparado com o DBML, foi necessário modificar o modelo de dados intermédio de forma a suportar estas propriedades descritas pelo formato SIARD. Tal alteração do modelo de dados intermédio é descrito em 4.4.1.

Uma diferença notável entre os dois formatos de preservação é o facto do formato DBML capturar a informação de tabelas, não considerando os diferentes *schemas* de um sistemas de gestão de base de dados, enquanto que o formato SIARD tem em consideração os possíveis diferentes *schemas*.

Assim, também o funcionamento do **db-preservation-toolkit** teve de ser adaptado de modo a comportar as necessidades do formato SIARD. Além disso, foi necessário um trabalho exaustivo de forma a corretamente mapear os diferentes tipos de dados dos vários sistemas de gestão de bases de dados. Também foram adicionadas algumas funcionalidades extra.

4.4.1 Nova estrutura interna db-preservation-toolkit

Dadas as especificações do formato SIARD, o modelo de dados intermédio foi alterado de forma a acomodar este formato. Assim, foram criadas as seguintes classes:

- `SchemaStructure`
- `UserStructure`
- `RoleStructure`
- `PrivilegeStructure`
- `ViewStructure`
- `RoutineStructure`
- `CandidateKey`
- `CheckConstraint`
- `Parameter`
- `Reference`
- `Trigger`

Do mesmo modo, classes como `DatabaseStructure`, `TableStructure`, `ColumnStructure`, `PrimaryKey` e `ForeignKey` já existentes quando o formato suportado era o DBML, tiveram de ser modificadas de forma a suportar as alterações introduzidas pela descrição do formato SIARD.

Estas novas classes, bem como a alteração das já existentes passaram a permitir que as propriedades definidas pelo formato SIARD fossem capturadas.

Assim, entre estas modificações, é importante destacar que a estrutura de uma base de dados (definida por `DatabaseStructure`), passou a suportar uma lista de *schemas* (`SchemaStructure`) em vez de uma lista de tabelas (`TableStructure`). A lista de tabelas passou, por sua vez, a estar presente na definição da classe `SchemaStructure`. Assim, um *schema*, entre outras propriedades, contém uma lista de tabelas.

Além de uma lista de *schemas*, a classe `DatabaseStructure` passou a incorporar listas de *users*, *roles* e *privileges*, informações não contempladas no DBML. A definição de `SchemaStructure` introduzida com o formato SIARD, além de uma lista de tabelas, apresenta também uma lista de *views* (`ViewStructure`) e *routines* (`RoutineStructure`).

A definição de estrutura de tabela foi também ela modificada, apresentando uma lista de *triggers* (`Trigger`). Adicionalmente, apresenta também uma lista de *candidate keys* (`CandidateKey`) e *check constraints* (`CheckConstraint`).

Assim, o modelo de dados intermédio, de acordo com o formato SIARD, passou a suportar a captura de propriedades comportamentais de uma base de dados.

4.4.2 Novo funcionamento do db-preservation-toolkit

De modo a suportar o funcionamento de múltiplos *schemas*, alguns ajustes tiveram de ser efetuados. Exemplo disso é a alteração do método `getDatabase`. Tal método passou a comportar-se como se mostra no excerto 4.1.

```
public void getDatabase(DatabaseHandler handler) {
    ...
    for (SchemaStructure schema: getDatabaseStructure().
        ↪ getSchemas()) {
        for (TableStructure table : schema.getTables()) {
            ...
        }
    }
    ...
}
```

Listagem 4.1: Excerto do novo método `getDatabase`

Além disso, e dada a necessidade de capturar mais informação relativa às bases de dados, foram adicionados métodos que permitem a recolha de dados como *schemas*, *users*, *roles* e *privileges* (a nível da estrutura de uma base de dados), bem como de tabelas, *views* e *routines* (a nível da estrutura de um *schema*). Ao nível da estrutura de uma tabela foi também necessário adicionar/alterar métodos responsáveis pela captura de informação relativa a chaves primárias, chaves estrangeiras, *triggers*, etc.

Os métodos responsáveis por obter a informação acima referida necessitam de aceder às base de dados. Tal é feito, via JDBC, e dentro do possível esses métodos tentam recolher a informação necessária fazendo uso de métodos específicos do JDBC. São exemplos desses métodos específicos, métodos como `getTables` ou `getColumns`, que permitem de forma direta recolher um conjunto, neste caso, de tabelas ou colunas que obedeçam a certos parâmetros. Contudo, nem toda a informação que necessita de ser recolhida pode ser obtida de forma tão direta.

Assim, é preciso fazer uso do método `executeQuery`, passando-lhe como parâmetro uma *query* SQL específica, de forma a obter os dados necessários. Por exemplo, o método do **db-preservation-toolkit** que recolhe a informação sobre os *triggers* ou *check constraints* necessita de fazer uso do `executeQuery`. Assim, para todos os sistemas de gestão de bases de dados que usam JDBC, foi preciso escrever *queries* que permitissem a recolha dos dados necessários.

4.4.3 Módulos de importação/exportação SIARD

Além das modificações necessárias a nível do modelo de dados intermédio do **db-preservation-toolkit**, e consequentemente do seu modo de funcionamento, foi necessária a implementação de módulos que permitissem a importação e exportação do pacotes SIARD. Deste modo, foram criados dois módulos: `SIARDImportModule` e `SIARDExportModule`, correspondendo aos módulos de importação e exportação, respetivamente.

Dado que um pacote SIARD é o mesmo que um arquivo ZIP, o módulo de importação SIARD faz uso do `ZipFile` (presente em `org.apache.commons.compress.archivers.zip`) e acede aos vários ficheiros contidos nesse arquivo, de forma a ler a sua informação. Assim, dado que um pacote SIARD é constituído por um ficheiro *metadata.xml*, contendo a metadata de uma base de dados arquivada, foi necessário utilizar um *parser* de forma a percorrer e ler a informação desse ficheiro XML. O mesmo procedimento é aplicado aos ficheiros *table<N>.xml* do pacote SIARD, que contêm os dados primários de uma base de dados.

Assim, de modo a fazer o *parse* dos vários ficheiros XML foi utilizado um `SAXParser`. De forma a conseguir fazer o *parse* de um ficheiro XML, o `SAXParser` necessita que seja passado um *handler* que faça o *extend* à classe `DefaultHandler`. Esta classe possui alguns métodos que devem ser substituídos de forma a capturar a informação necessária em ficheiros XML distintos.

Deste modo, foram criados os *handlers* `SIARDHeaderSAXHandler` e `SIARDContentSAXHandler`, responsáveis, respetivamente, por lidar com o ficheiro *metadata.xml* e *table<N>.xml* (vários ficheiros, mas com a mesma definição de *schema* XML).

Uma vez que o `SIARDImportModule` se trata de um módulo de importação, este implementa a interface `DatabaseImportModule`, sendo o método `getDatabase` responsável por articular o *parse* da metadata, e posteriormente o *parse* dos vários ficheiros contendo a informação tabular. Os *parsers* tanto da metadata, bem como dos ficheiros que

contêm a informação tabular são os responsáveis por executar os métodos `handleDatabaseStructure` (no caso do *parser* da metadata) e `handleDataOpenTable`, `handleDataCloseTable` e `handleDataRow` (no caso do *parser* dos dados primários).

No que diz respeito ao módulo de exportação SIARD, faz-se uso do `ZipArchiveOutputStream` (presente em `org.apache.commons.compress.archivers.zip`) de modo a criar e escrever para um arquivo ZIP. Dado que o formato SIARD usa o ZIP 64, foi necessário assegurar que o mesmo é usado na criação do arquivo SIARD de exportação. Tal pode ser observado no excerto 4.2.

```
...
zipOut = new ZipArchiveOutputStream(siardPackage);
zipOut.setUseZip64(Zip64Mode.Always);
...
```

Listagem 4.2: Uso do modo ZIP 64

O modo utilizado para criar os vários ficheiros XML contendo tanto a informação tabular, bem como a metadata de uma base de dados a ser arquivada, passa por escrever *bytes* para os diferentes *archive entries* de um arquivo ZIP. Cada uma destas entradas representa um ficheiro dentro do arquivo ZIP.

O `SIARDExportModule` implementa a interface `DatabaseHandler`. Assim sendo, através dos seguintes excertos, será demonstrada a forma como se implementam os métodos dessa interface, de forma a escrever os dados primários nas diversas *entries* de um arquivo ZIP, assim como a metadata de uma base de dados.

Quando se dá início ao processo de preservação de uma base de dados com o **db-preservation-toolkit**, a certa altura do código do módulo de importação, serão chamados os métodos do *handler* de exportação. Assim, como descrito no excerto 4.3, é criada uma entrada do arquivo ZIP, e posteriormente exportada a informação necessária.

```
public void handleDataOpenTable(String tableId) throws
    ↪ ModuleException {
    ...
    ArchiveEntry archiveEntry = new ZipArchiveEntry(
        ↪ entryPath);
    ...
    zipOut.putArchiveEntry(archiveEntry);
    isWritingContent = true;
    exportDataOpenTable(tableFolder);
    ...
```

```
}
```

Listagem 4.3: Excerto do método `handleDataOpenTable`

Essa exportação, ou seja, escrita de informação para a entrada do arquivo ZIP, é feita através do método `print` (visível no excerto 4.4).

```
private void print(String s) throws IOException {
    byte[] bytes = s.getBytes();
    if (isWritingContent) {
        digest.update(bytes);
    }
    zipOut.write(bytes);
}
```

Listagem 4.4: Excerto do método `print`

De forma a exportar uma linha de uma tabela é usado o método `handleDataRow`. No caso do `SIARDExportModule` este método está implementado da forma retratada pelo excerto 4.5. De forma semelhante ao método `handleDataOpenTable`, também este método faz uso do método `print`, de modo a escrever a informação na entrada do arquivo ZIP.

```
public void handleDataRow(Row row) throws
    ↪ InvalidDataExceptio, ModuleException {
    ...
    exportRowData(row);
    ...
}
```

Listagem 4.5: Excerto do método `handleDataRow`

Por fim, é apresentado no excerto 4.6 o método responsável por concluir o tratamento da informação de exportação de uma tabela.

```
public void handleDataCloseTable(String tableId) throws
    ↪ ModuleException {
    ...
    exportDataCloseTable();
    zipOut.closeArchiveEntry();
    ...
    isWritingContent = false;
    ...
}
```

Listagem 4.6: Excerto do método `handleDataCloseTable`

Como é possível verificar, no método `print`, a linha `digest.update(bytes)` é responsável pelo *update* do *message digest*. Isto é, a cada escrita de informação primária no arquivo SIARD, é construído o *message digest* (usando o algoritmo MD5) que tem como propósito garantir a integridade dos dados primários. A atualização contínua do *message digest* ao longo da escrita dos dados foi a forma encontra de conseguir construir o *message digest*, necessário de acordo com o formato SIARD.

O ficheiro *metadata.xml* é criado no arquivo SIARD de forma semelhante ao que acontece com os vários ficheiros XML que contêm a informação tabular.

Contudo, e porque o *message digest* da informação primária só pode ser obtido depois do processamento dessa mesma informação, o ficheiro *metadata.xml* só pode ser criado no final da escrita dos ficheiros de informação primária. Por esta razão, e dado que o método `handleDatabaseStructure` é usado cronologicamente, pelos módulos de importação, primeiro que os métodos que processam a informação primária, foi necessário escrever o código correspondente à exportação da metadata no método `finishDatabase`, que é o último método a ser executado aquando a preservação de uma base de dados.

4.4.4 Mapeamento para SQL:1999

Com a adição do suporte ao formato SIARD, os tipos de dados deixaram de ser definidos de acordo com o formato DBML. Assim, e passando os tipos de dados a estar de acordo com o SQL:1999, como especifica o formato SIARD, várias alterações tiveram de ser feitas. Tais modificações tiveram de acontecer nos seguintes níveis:

- **ao nível de importação vs. exportação**: tanto a nível de importação como ao nível de exportação, tiveram de ser modificadas as formas de como ocorria o mapeamento a nível dos tipos de dados, bem como o seu conteúdo.
- **ao nível de comunicação via JDBC vs. arquivo SIARD**: também a forma de mapear os tipos de dados e seu conteúdo é diferente se este mapeamento ocorrer ao nível de JDBC, ou se a fonte/destino for um arquivo SIARD. O primeiro necessitou de vários ajustes, enquanto que o último precisou que fosse criado um mecanismo de mapeamento.

Assim, de modo a dar uma visão geral do que teve de ser modificado e criado, é apresentada na tabela 4.3 os principais componentes que sofre-

ram alterações e em que categoria se enquadram.

	Importação	**Exportação**
JDBC	`getType`	`createTypeSQL`
SIARD	`createType`	`exportType`

Tabela 4.3: Divisão de métodos criados/alterados de forma a suportar o mapeamento para SQL:1999

`getType`

Como já foi referido no capítulo 3, o método `getType` é o responsável por definir o tipo de dados de cada coluna, nos módulos de importação que usam JDBC. Para isso, tal informação é armazenada num objeto da classe `Type`. Esta classe, em comparação ao **db-preservation-toolkit** v1.0 possui uma variável adicional chamada `sql99TypeName`, que guarda o tipo de dados de acordo com o SQL:1999.

Desta forma, quando o tipo de dados JDBC (`java.sql.Types`) não deixa dúvidas de como ser mapeado, a sua representação no modelo de dados intermédio é criada e o tipo de dados SQL:1999 é definido correspondentemente. Tal é possível verificar pelo excerto 4.7. Neste exemplo, independentemente da *driver* JDBC que defina uma coluna da base de dados como sendo do tipo `Types.BLOB`, este será representando internamente como sendo do tipo `SimpleTypeBinary` e sendo o seu tipo SQL:1999 *BINARY LARGE OBJECT*.

```
protected Type getType(int dataType, String typeName, int
    ↪  columnSize, int decimalDigits, int numPrecRadix)
    ↪ throws UnknownTypeException {
     Type type;
     ...
     case Types.BLOB:
          type = new SimpleTypeBinary(Integer.valueOf(
             ↪ columnSize));
          type.setSql99TypeName("BINARY␣LARGE␣OBJECT");
          break;
     case Types.BOOLEAN:
          type = new SimpleTypeBoolean();
          type.setSql99TypeName("BOOLEAN");
          break;
     ...
}
```

Listagem 4.7: Excerto do método `getType`

Contudo, dadas as muitas diferenças entre os vários sistemas de gestão de bases de dados; a forma como as suas *drivers* definem os tipos de dados de uma coluna; e o facto de colunas de tipos de dados distintos num sistema de gestão de base de dados serem definidas com o mesmo tipo de dados JDBC, faz com que seja necessário que cada módulo de importação seja capaz de escolher o tipo de dados do modelo intermédio do **db-preservation-toolkit**e o tipo de dados SQL:1999 mais adequado.

Assim, quando o tipo de dados não é claro sobre a forma como deve ser mapeado (ao contrário do que se mostra no excerto 4.7), cada módulo de importação fica responsável por substituir o método que define como é feito o mapeamento para um tipo de dados do modelo intermédio.

O excerto 4.8 mostra que o tipo de dados de uma coluna, em que o seu tipo de dados JDBC é `Types.TIMESTAMP`, é definido à custa do método `getTimestampType`. Cada módulo de importação deve fazer *override* a este método se a sua definição não for a adequada. Exemplo disso pode ser encontrado no excerto 4.9, que mostra como o método `getTimestampType` é redefinido de forma a ajustar-se ao modo como opera a *driver* JDBC do PostgreSQL.

```
protected Type getType(int dataType, String typeName, int
    ↪  columnSize, int decimalDigits, int numPrecRadix)
    ↪ throws UnknownTypeException {
     ...
     case Types.TIMESTAMP:
         type = getTimestampType(typeName, columnSize,
             ↪ decimalDigits,
                  numPrecRadix);
         break;
     ...
}
```

Listagem 4.8: Excerto do método `getType`

```
protected Type getTimestampType(String typeName, int
    ↪ columnSize,
         int decimalDigits, int numPrecRadix) {
     Type type;
     if (typeName.equalsIgnoreCase("TIMESTAMPTZ")) {
         type = new SimpleTypeDateTime(Boolean.TRUE,
             ↪ Boolean.TRUE);
     } else {
         type = new SimpleTypeDateTime(Boolean.TRUE,
             ↪ Boolean.FALSE);
     }
     type.setSql99TypeName("TIMESTAMP");
```

```
    return type;
}
```

Listagem 4.9: Excerto do método `getTimestampType` da classe `PostgreSQLJDBCImportModule`

`createTypeSQL`

De modo a exportar a informação de uma base de dados que seja acedida via JDBC, são criadas *queries* SQL específicas para cada sistema de gestão de base de dados, de forma a que seja definida a estrutura da base de dados, de acordo com o modelo de dados intermédio. Tal é feito com recurso a `SQLHelper`'s, classes que possuem métodos de criação de *queries* específicas para diferentes sistemas de gestão de bases de dados. É exemplo disso, uma *query* como "*CREATE TABLE <tableName> (<columnName> <dataType>, ...)*".

As diferenças mais importantes entre as *queries* de sistemas de gestão de bases de dados encontram-se nos diferentes tipos de dados que cada sistema de gestão de bases de dados usa. Assim, cada sistema de gestão de bases de dados suportado pelo **db-preservation-toolkit**, mais especificamente o `SQLHelper` correspondente, possuí um método `createTypeSQL` essencial à forma como o tipo de de dados de cada coluna, representado no modelo de dados intermédio do **db-preservation-toolkit**, é mapeado para o tipo de dados específico de um sistema de base de dados.

O excerto 4.10 pretende representar como é feito o mapeamento do tipo de dados `SimpleTypeNumericApproximate` para um tipo de dados do PostgreSQL.

```
protected String createTypeSQL(Type type, boolean isPkey,
    ↪  boolean isFkey) throws UnknownTypeException {
     ...
     } else if (type instanceof
         ↪ SimpleTypeNumericApproximate) {
          SimpleTypeNumericApproximate numericApproximate =
                  (SimpleTypeNumericApproximate) type;
          if (type.getSql99TypeName().equalsIgnoreCase("
             ↪ REAL")) {
              ret = "real";
          } else if (StringUtils.startsWithIgnoreCase(
                  type.getSql99TypeName(), "DOUBLE")) {
              ret = "double␣precision";
          } else {
              ret = "float(" + numericApproximate.
```

```
                ↪ getPrecision() + ")";
        }
    }
    ...
}
```

Listagem 4.10: Excerto do método `createTypeSQL` da classe `PostgreSQLHelper`

Este processo teve de ser feito para todos os sistemas de gestão de bases de dados, havendo a necessidade de definir o mapeamento mais adequado de um tipo de dados do modelo de dados intermédio, para o tipo de dados de um sistema de gestão de bases de dados específico.

`createType` e `exportType`

O módulo de importação SIARD, uma vez que possui apenas uma origem de tipo de dados, e esses mesmos tipos de dados já são definidos de acordo com o SQL:1999, faz com que o processo de mapeamento para o modelo de dados intermédio seja mais direto.

Assim, através do método `createType`, usado na altura em que ocorre o *parse* do ficheiro *metadata.xml* de um arquivo SIARD, é feito o mapeamento para o modelo de dados intermédio. A variável `sql99TypeName`, pertencente à classe `Type`, será igual à informação contida em *metadata.xml*, uma vez que os tipos de dados do arquivo SIARD se encontram em SQL:1999. A única exceção acontece quando o tipo de dados de uma coluna de um arquivo SIARD é do tipo *BIT*. Neste caso, essa coluna será definida internamente como sendo do tipo *BOOLEAN*.

Por sua vez, aquando da exportação de um sistema de gestão de base de dados para um arquivo SIARD, é necessário que o os tipos de dados do modelo de dados intermédio sejam mapeados para SQL:1999. Apesar do modelo de dados intermédio conter informação sobre a que tipo de dados SQL:1999 certa coluna pertence, existem alguns fatores que fazem com que o mapeamento para SQL:1999 não seja direto [GP99]. Isto é, apesar de cada coluna representada internamente ter um tipo de dados já definido em SQL:1999, pode acontecer de não ser esse o tipo de dados SQL:1999 escolhido para representar dada coluna num arquivo SIARD. Assim, no excerto 4.11 mostra-se um exemplo do que foi referido em cima.

```
protected String exportSimpleTypeBinary(Type type) {
    ...
    if (sql99TypeName.equalsIgnoreCase("BIT")) {
```

```
        if (type.getOriginalTypeName().equalsIgnoreCase("
            ↪ TINYBLOB")) {
            ret = "BIT␣VARYING(2040)";
        } else if (type.getOriginalTypeName().
            ↪ equalsIgnoreCase("BIT")) {
            ret = "BIT(" + length + ")";
        } else {
            ret = "BIT(" + length * 8 + ")";
        }
    } else if ...
    ...
    return ret;
}
```

Listagem 4.11: Excerto do método `exportSimpleTypeBinary` da classe `SIARDExportHelperMySQL`, responsável por lidar com dados do tipo `SimpleTypeBinary`

Neste caso, apesar de uma coluna ser do tipo SQL:1999 *BIT*, dado que a sistema de gestão de base de dados importado foi o MySQL, faz com o que o tipo de dados SQL:1999 possa ser *BIT(<size>)* ou *BIT VARYING(2048)*. Assim, dadas diferentes origens da bases de dados a serem preservadas, e o mapeamento do tipo de dados (já na forma de SQL:1999) do modelo de dados intermédio não ser direto para SQL:1999, foi necessário criar *helpers* distintos para sistemas de gestão bases de dados distintos.

4.4.5 DB2 e sistemas de gestão de bases de dados suportados

Com a adição do suporte ao formato SIARD, todos os módulos de importação/exportação previamente suportados deixaram de operar corretamente. Assim, o facto de se tratar de um módulo já existente no **dbpreservation-toolkit** v1.0, ou o facto de se tratar de um novo módulo, foi, na prática, pouco relevante. Isto é, os módulos já existentes, nas componentes que tratam do mapeamento dos tipos de dados, tiverem de ser revistos e modificados, processo que aconteceu também aquando da adição de um novo módulo de suporte a outro sistema de gestão de bases de dados via JDBC.

Assim, uma vez feitos os desenvolvimentos que permitem a importação/exportação de SIARD, e feitos os vários ajustes a nível do módulo `JDBCImportModule`, o trabalho necessário para adicionar mais módulos de suporte a sistemas de gestão de bases de dados via JDBC, prende-se, de modo geral, em definir corretamente os métodos que tratam do mape-

amento dos tipos de dados.

Dado, desta forma, o facto de se perder o suporte a todos os outros módulos de importação/exportação, foi necessário re-adicionar, o suporte aos módulos já existentes. Assim, os módulos de importação e exportação MySQL, PostgreSQL, SQLServer e Oracle (apenas importação) continuam a ser suportados com as modificações impostas pelo suporte ao formato SIARD. Do mesmo modo, o suporte ao sistema de gestão de bases de dados DB2 foi adicionado.

Capítulo 5

Serviço de visualização e pesquisa

No âmbito desta dissertação foi desenvolvido um serviço que possibilita a visualização e pesquisa de informação de uma base de dados preservada no formato SIARD.

Assim, neste capítulo, são descritos certos tópicos no que diz respeito ao que levou a que fosse feito este serviço (em 5.1), considerações sobre o seu funcionamento (em 5.2), o módulo responsável pelo visualizador (em 5.3), bem como considerações sobre a sua implementação (em 5.4).

5.1 Porquê

Como já foi mencionado, um dos objetivos desta dissertação passa por adicionar suporte ao formato SIARD. Uma necessidade que surge frequentemente é facto de ser preciso compreender a estrutura, visualizar e pesquisar a informação de uma base de dados arquivada no formato SIARD.

Para tal, como já foi referido, o SIARD Suite permite visualizar a informação e estrutura de uma base de dados. Contudo, não é possível pesquisar informação de uma base de dados. Uma forma de contornar esta incapacidade passa por reimportar a base de dados arquivada num sistema de gestão de base de dados, e então fazer as pesquisas necessárias. Porém, esta é uma opção que poderá ser considera trabalhosa e não muito simples.

Desta forma, e com vista a melhorar as poucas soluções existentes, se pode justificar o desenvolvimento deste serviço de visualização e pesquisa.

5.2 Considerações sobre o funcionamento

Antes de demonstrar como foi desenvolvido este serviço, é importante referir o caminho tomado no que diz respeito à avaliação de tecnologias disponíveis para levar a cabo esta operação. Também é necessário referir que um tópico importante no desenvolvimento deste serviço passou pela facilidade de instalação/configuração de dependências necessárias para executar este serviço.

Assim, de forma a visualizar e pesquisar informação de uma base de dados, foi decidido que tal seria feito através de uma aplicação web. Deste modo, a futura integração deste serviço com outros já existentes seria mais fácil. Para tal, o modo de alcançar este objetivo passou por adicionar um novo módulo de exportação ao **db-preservation-toolkit**, permitindo a criação de um pacote com as componentes necessárias para executar o visualizador.

Idealmente, este pacote, seria constituído por ficheiros .html e respetivos recursos (.js, .css), sendo a sua execução feita exclusivamente no lado do cliente. Esta seria uma solução simples que implicaria apenas uma exportação (de forma a criar o pacote) e um *click* (de forma a abrir a página web). Para que esta solução fosse possível, era necessário que a informação a ser visualizada e pesquisada fosse processada pelo lado cliente, através de Javascript.

Para tal, de modo a implementar este serviço, foi necessário investigar tecnologias que permitissem a pesquisa em grandes quantidades de informação textual. É possível alcançar este requisito através de pesquisas *full-text*. Em pesquisas *full-text* o motor de pesquisa examina todas as palavras presentes em vários documentos, de forma a tentar associar essas palavras com a pesquisa efetuada pelo utilizador.

Além deste requisito, seria importante que o motor de pesquisa *full-text* pudesse ser executada unicamente do lado do cliente. Assim, depois de alguma investigação feita sobre tecnologias que poderiam ser úteis, foram conseguidos alguns resultados:

- **Lunr.js**: é um motor de pesquisa de texto simples para aplicações *client-side*. Foi desenhado para ser pequeno, mas com todas as funcionalidades necessárias de forma disponibilizar uma boa experiência de pesquisa sem a necessidade de serviços de pesquisa externos, *server-side*.

  É uma tecnologia *open source*, que na altura de escrita desta dissertação se encontra na versão 0.5.4, não tendo atingido ainda uma *release* estável [Nig].

- **Lucene.js**: descreve-se como sendo um *port* para Javascript do Apache Lucene Core (um motor de pesquisa). Lucene.js pode ser usada como um módulo Javascript *server-side* executada em Node.js, mas também como uma biblioteca Javascript que pode ser executada num *browser client-side*.

  À semelhança do Lunr.js, também é uma tecnologia *open source*. Contudo, não existem desenvolvimentos recentes neste serviço, sendo que o mesmo não possui nenhuma *release*, levando a questionar acerca da maturidade desta tecnologia [Dai].

- **Norch**: Este motor de pesquisa define-se como sendo um motor de pesquisa experimental construído em Node.js. Apesar desta tecnologia aparentar ser mais estável e madura que as tecnologias previamente mencionadas, bem como apresentar atividade recente no que diz respeito a desenvolvimentos, necessita de um servidor para ser executada. Também ela é *open source* [McD].

- **Apache Solr**: É descrito como uma plataforma popular de pesquisa empresarial *open source* do projeto Apache Lucene. Entre as suas principais funcionalidades encontra-se a pesquisa *full-text*, *faceted search*, entre outras.

  O Solr define-se como sendo altamente confiável, escalável e tolerante a falhas, e é responsável pela pesquisa e funcionalidades de navegação de muitos dos maiores sites a nível mundial [Foub].

Pode-se constatar que tanto o Lunr.js como o Lucene.js apresentam a capacidade de correr exclusivamente do lado do cliente. Contudo, como já referido, não apresentam a maturidade necessária para servirem de suporte ao serviço de pesquisa a ser desenvolvido, que terá, possivelmente, de lidar com quantidades enormes de informação. Por outro lado, o Norch, além de se definir ele próprio como sendo um motor de pesquisa experimental, necessita de ser executada no lado do servidor.

Deste modo, dada a inabilidade e imaturidade destas tecnologias, foi necessário procurar tecnologias maduras, mesmo que estas necessitassem de instalações/configurações por parte do utilizador final. Assim sendo, já que as três primeiras tecnologias apresentadas não se apresentavam como viáveis para suportar este serviço fez com que não fossem utilizadas. Pelo contrário, o Solr apresentava boas capacidades, funcionalidades e a maturidade pretendida para este serviço, acabando por ser a escolha evidente (ignorando o facto de ser uma tecnologia *server-side*).

5.2.1 Apache Lucene

Conceitos base

O Apache Lucene é uma biblioteca de um motor de pesquisa *full-text* de alta *performance*, escrita inteiramente em Java. É uma tecnologia que pode ser usada em quase todas as aplicações que necessitem de pesquisa *full-text*. O Lucene não se trata de uma aplicação completa, mas sim de uma biblioteca e API que pode ser facilmente usada para adicionar capacidade de pesquisa a aplicações. É com recurso ao Lucene, que o Solr funciona.

De modo a operar, o Lucene adiciona conteúdo a um *index full-text*. Assim, é possível executar *queries* a este *index*, retornando os resultados classificados por relevância ou ordenados por um campo do documento [MHG10].

Pesquisa e indexação

O Lucene consegue atingir rápidas respostas de pesquisa porque, em vez de pesquisar diretamente no texto, este pesquisa num *index*. Isto é o equivalente a obter as páginas de um livro relacionadas com uma dada palavra por via do índice na parte de trás do livro, em oposição a pesquisar as palavras em todas as páginas do livro.

Este tipo de índice é chamado um índice invertido porque inverte a estrutura de dados centrada na página (página -> palavras) para uma estrutura de dados centrada na palavra a pesquisar (palavra -> páginas) [Tana].

Documentos

No Lucene, um **documento** é a unidade de pesquisa e indexação. Um *index* consiste num ou mais documentos.

A indexação passa por adicionar documentos a um `IndexWriter` e sua pesquisa passa por obter esses documentos de um *index* via um `IndexSearcher` [Tana].

Um documento Lucene não é necessariamente um documento no sentido literal da palavra. Por exemplo, num *index* Lucene de uma tabela de utilizadores de uma base de dados, cada utilizador seria representado no *index* Lucene como um documento Lucene [BMI12].

Campos

Um documento consistem num ou mais campos. Um campo é simplesmente um par chave-valor. Por exemplo, um campo que represente um título teria como chave o nome "título", e o seu valor seria o conteúdo do título.

Assim, a indexação no Lucene, envolve criar documentos contendo um ou mais campos, e adicionar esses documentos ao `IndexWriter` [Foua].

Pesquisa

A pesquisa requer que um *index* tenha já sido construído, envolvendo criar uma `Query` e passar essa `Query` a um `IndexSearch`, que retorna a lista de resultados [Tana].

Queries

O Lucene apresente a sua própria sintaxe para efetuar pesquisas. Esta é a chamada `Lucene Query Syntax` [Foua].

5.2.2 Apache Solr

Conceitos base

O Solr foi desenvolvido em Java e corre como um servidor de pesquisa *full-text* dentro de um *servlet*, como por exemplo o Jetty. O Solr usa a biblioteca Java do Lucene como base para a pesquisa e indexação *full-text*, possuindo APIs HTTP/XML e JSON do tipo REST que fazem com que seja fácil ser usado em praticamente todas as linguagens de programação.

Além disso, o Solr, através da sua boa configuração externa, permite que este seja ajustado a qualquer tipo de aplicação sem a necessidade de código Java, possuindo uma extensa arquitetura de *plugins* quando são precisas modificações mais avançadas.

O Solr usa os mesmo conceitos do Lucene, já descrito em cima. Assim as componentes ligadas à indexação, como por exemplo, os documentos de um *index*, os campos, etc, podem ser descritas da mesma forma que foram descritas para o Lucene. Assim, é o Lucene que se encontra como mecanismo que serve de indexação e pesquisa que o Solr usa [Foub].

Representação de informação

Apesar de já referido acima, convém reforçar que um *index* consiste num ou mais documentos, e um documento consiste num ou mais campos. Traduzindo esta organização para terminologia de base de dados relacionais, um documento corresponde uma linha de uma tabela, e um campo a uma coluna da tabela.

Antes de adicionar documentos ao Solr, é preciso especificar o *schema*, representado pelo ficheiro *schema.xml*. O *schema* declara que tipo de campos existem, que campos devem ser usados como sendo único/chave primária, que campos são obrigatórios e como indexar e pesquisar certo campo.

Cada campo pertence a um tipo. Além disso, o Solr expande a variedade de tipos de campos disponíveis no Lucene. Alguns exemplos de tipos de campos básicos disponíveis no Solr são o caso de `float`, `long`, `double`, `date`, `text`, etc. O Solr também permite que sejam definidos novos tipos de campos [Tanb].

Por sua vez, a forma de, no *schema*, definir um campo parece-se com a seguinte:

```
<field name="id" type="text" indexed="true" stored="true"
    ↪  multiValued="true"/>
```

Listagem 5.1: Descrição do elemento `field` de um *schema* Solr

Cada atributo pode ser descrito da seguinte forma:

- **name**: nome do campo
- **type**: tipo do campo
- **indexed**: se este campo deve ser adicionado ao *index* invertido
- **stored**: se o valor original deste campo deve ser guardado
- **multiValued**: se este campo pode ter valores múltiplos num documento (o mesmo documento apresenta este campo com vários valores diferentes)

5.3 Módulo de exportação do visualizador

De forma a criar o pacote que contém o visualizador foi necessário desenvolver um módulo de exportação no **db-preservation-toolkit**. Este

módulo funciona de forma semelhante aos outros módulos de exportação. Isto é, dada certa informação, importada através de um módulo de importação, esta é processada de acordo com o definido no módulo de exportação.

Neste caso, o objetivo deste módulo de exportação é criar o pacote contendo os recursos necessários para a execução do visualizador. Assim, podem-se dividir, em três, os conjuntos de recursos necessários para dar vida a este visualizador:

- `EmbeddedSolrServer` (contendo os *indexes* Lucene)
- aplicação web (.html, .css, .js)
- *scripts* de inicialização

Como já foi referido mais acima, a opção de desenvolver um visualizador que funcione puramente com recurso ao *client-side* não é viável, devido à pouca maturidade das tecnologias existentes.

Pode-se verificar pelos três conjuntos de recursos mencionados na lista imediatamente acima, que o pacote do visualizador, além de conter a aplicação web, se faz acompanhar de um servidor Solr, que permite aceder à informação contida nos *indexes* Lucene.

O terceiro item, ou seja, os *scripts* de inicialização minimizam as configurações necessárias por parte do utilizador, tentando trazer ao serviço de visualização o efeito originalmente desejado: "um *click* para executar o visualizador". Deste modo, é possível, na maior parte dos casos, alcançar o mesmo resultado que se esperaria de uma aplicação exclusivamente *client-side*.

Uma das componentes mais importantes deste visualizador é o servidor Solr. Quando se fala deste assunto, está-se implicitamente a falar do *index* Lucene que contêm a informação de uma base de dados arquivada. É sobre este *index* que o servidor Solr trabalha de forma a disponibilizar a informação necessária ao visualizador.

Deste modo, é importante explicar como está organizado um *index* Lucene no **db-preservation-toolkit**.

5.3.1 Estrutura do *index* Lucene

Como já foi referido, um documento é a unidade de indexação e de pesquisa de um *index* Lucene. De forma a ser possível indexar e pesquisar a informação de uma base de dados relacional é necessário transcrever tal informação para se adapte à forma de **documento** Lucene.

Neste caso, essa transcrição/mapeamento de informação de uma base de dados relacional para um documento, ocorre ao nível da linha de uma tabela de uma base de dados. Isto é, cada linha de uma tabela terá a si associado um documento. Desta forma, a cada *field* (chave) de um documento será associada a informação de uma coluna na linha a que se refere o documento. Por outras palavras, cada documento terá vários pares chave-valor, contendo na chave a referência à coluna a que pertence, e no valor a informação da célula naquela coluna, em determinada linha [GP14].

Além disso, existem outros pares chave-valor muito importantes em cada documento. São estes os pares que contêm a metadata de uma tabela. Cada documento (contendo a informação de uma linha) contém a informação necessária para se identificar globalmente perante outros documentos.

Assim, por exemplo, a informação sobre o *schema* e a tabela a que essa linha pertence é vital e está presente em cada documento. Outras informações correspondentes à metadata de um documento Lucene são o nome e tipo de dados das colunas da tabela a que essa linha pertence. Como é possível constatar, haverá metadata repetida nos vários documentos, contudo tal é necessário dada a hierarquia *flat* dos *indexes* Lucene.

De forma a melhor compreender a organização de um documento, é possível visualizar um exemplo no excerto 5.2.

```
...
{
  "dbpres_meta_schema": "public",
  "dbpres_meta_table": "myTable",
  "dbpres_meta_tableId": "public.myTable",
  "dbpres_meta_id": "public.myTable.228275",
  "dbpres_meta_rowN": 228275,
  "dbpres_meta_col_1": "id",
  "dbpres_meta_colType_1": "serial",
  "dbpres_data_1": "228245",
  "dbpres_meta_col_2": "name",
  "dbpres_meta_colType_2": "varchar",
  "dbpres_data_2": "Jesse Jennings",
  "dbpres_meta_col_3": "company",
  "dbpres_meta_colType_3": "varchar",
  "dbpres_data_3": "Phasellus Libero Company",
  "_version_": 1479598395177828400
}
...
```

Listagem 5.2: Exemplo de documento Lucene

Como é possível observar, os nomes das chaves de informação relativa à metadata encontram-se prefixados com **dbpres_meta**, enquanto que os nomes dos campos de informação primária encontram-se prefixados com **dbpres_data**. É de referir que cada conjunto de chaves "dbpres_data_col", "dbpres_data_col_type" e "dbpres_data" contêm o sufixo referente ao número da coluna que representam.

5.3.2 EmbeddedSolrServer

A forma de criar o *index* Lucene e de o fornecer ao servidor Solr é feita através do módulo `EmbeddedSolrServer`, da biblioteca do Solr. Para tal são necessários de antemão os ficheiros responsáveis para executar o servidor Solr. Num diretório específico, denominado por *solr home*, onde ficará disponível o *index* Lucene, terão de estar presentes certas configurações para que seja possível ter acesso à informação desejada [Foub].

Um desses ficheiros de configuração trata-se do ficheiro *schema.xml*, como já referido acima. O *schema* de um certo *index* deve ser construído de acordo com a estrutura do documento que se quer representar.

O módulo de exportação do pacote contendo o visualizador, por sua vez, faz a ligação com o *solr home* e cria o *index*, contendo os documentos que possuem tanto metadata como informação primária.

No excerto 5.3 é possível verificar alguns elementos mais importantes do *schema.xml*.

```
...
<uniqueKey>dbpres_meta_id</uniqueKey>
<defaultSearchField>text</defaultSearchField>
<field name="text" type="string" indexed="true" stored="
    ↪ true" multiValued="true" />

<field name="dbpres_meta_id" type="string" indexed="true"
    ↪  stored="true" required="true" />
<field name="dbpres_meta_schema" type="string" indexed="
    ↪ true" stored="true" />
<dynamicField name="dbpres_meta_colType_*" type="string"
    ↪ indexed="true" stored="true"/>
<dynamicField name="dbpres_meta_col_*" type="string"
    ↪ indexed="true" stored="true" />
<dynamicField name="dbpres_data_*" type="string" indexed=
    ↪ "true" stored="true" />
<!-- ... (definicao de campos de meta informacao) -->
<copyField source="dbpres_data_*" dest="text" />
...
```

Listagem 5.3: Excerto do *schema.xml*

É possível verificar no excerto alguns elementos presentes no *schema.xml* como é o caso do elemento `uniqueKey` e alguns elementos correspondentes à metadata. Destacam-se os campos dinâmicos "dbpres_meta_colType_*", "dbpres_meta_col_*" e "dbpres_data_*" que na prática, num documento se traduzirão em chaves do tipo "dbpres_data_1", por exemplo.

O elemento `defaultSearchField` identifica em que campo deve o Solr pesquisar, quando numa *query* não é definido explicitamente um campo a pesquisar. Os elemento `copyField` torna possível copiar a informação de um campo de origem para outro campo de destino. Assim, ao copiar a informação do campo `dbpres_data_*` (todos os campos de informação) para o campo `text` (campo definido como `defaultSearchField`) faz com que seja possível pesquisar por um termo sem definir à partida em que campo deve esse termo ser pesquisado [Nay14].

5.4 Implementação da aplicação web

De forma a implementar o serviço de visualização e pesquisa foi usada a *framework* de Javascript Angular.js [1].

De modo a desenvolver o visualizador, mostrar informação, assim como pesquisa-la, é necessário obter a informação correspondente à construção de cada umas destas partes. Para tal, é necessário fazer *queries* específicas ao servidor Solr, de acordo com sintaxe própria de *queries* Solr [Kuc13].

Assim, na aplicação web, foi desenvolvido um serviço denominado por `Solr` que é responsável por fazer pedidos ao servidor Solr [Lav14]. Para tal, faz-se uso do serviço `$resource` do Angular.js, que tem como função criar um objeto que permite interagir com uma fonte de dados *server-side* REST.

Como já foi referido, existem dois tipos de dados a recolher do servidor: informação primária e metadata. O primeiro tipo de dados permite que a informação tabular, ou seja, a informação contida numa tabela seja exibida. Por usa vez, o segundo tipo de dados permite mostrar informações sobre a base de dados, como por exemplo, o nome dos *schemas*, tabelas, colunas, os tipos de dados das colunas, entre outros. É também com este tipo de informação que é possível criar menus de navegação.

[1] `http://angularjs.org`

Desta forma, através do serviço `Solr` desenvolvido, são acedidos os dados responsáveis para criação de cada uma das componentes da interface da aplicação web [Bra14].

Como é possível verificar na figura 5.1, a interface do visualizador **db-preservation-toolkit** apresenta um menu na esquerda e uma área de visualização de dados à direita. Além disso, nessa mesma área de visualização está presente uma caixa de texto que permite a realização pesquisas.

5.4.1 Informação no menu de navegação

De modo a construir o menu, foram desenvolvidos outros serviços na aplicação, que usando o serviço `Solr`, disponibilizam a metadata necessária a ser exibida. No excerto 5.4 são apresentados os parâmetros de uma *query* Solr necessários para obter a metadata dos *schemas* de uma base de dados arquivada (presente no *index* Lucene).

```
"params":{
  "q":"*:*",
  "facet.field":"dbpres_meta_schema",
  "start":"0",
  "rows":"0",
  "facet":"true"
 }
```

Listagem 5.4: Parâmetros enviados no pedido de metadata de *schemas*

Esta informação é conseguida através do uso da funcionalidade de *faceting* do Solr. Esta funcionalidade, dada uma certa chave presente em documentos Lucene, passada como `facet.field`, permite a contagem do número de documentos em que cada valor dessa chave está presente [Foub]. Isto é, e dando como exemplo o excerto referido acima, para um pedido em que o `facet.field` é "dbpres_meta_schema" (e em que "dbpres_meta_schema" é uma chave que está presente nos vários documentos do *index* Lucene do visualizador **db-preservation-toolkit**), serão resultados deste pedido, o número de documentos pertencentes a cada *schema*. O resultado seria por exemplo `[schema1, 100, schema2, 121]` em que `schema1` e `schema2` são valores da chave "dbpres_meta_schema".

Aproveitando esta funcionalidade, é possível saber o nome dos vários *schemas* de uma base de dados arquivada. Do mesmo modo, é possível saber, entre outra informações, o nome das tabelas de um dado *schema*. Para tal, um pedido semelhante ao apresentado em 5.4 teria de ser exe-

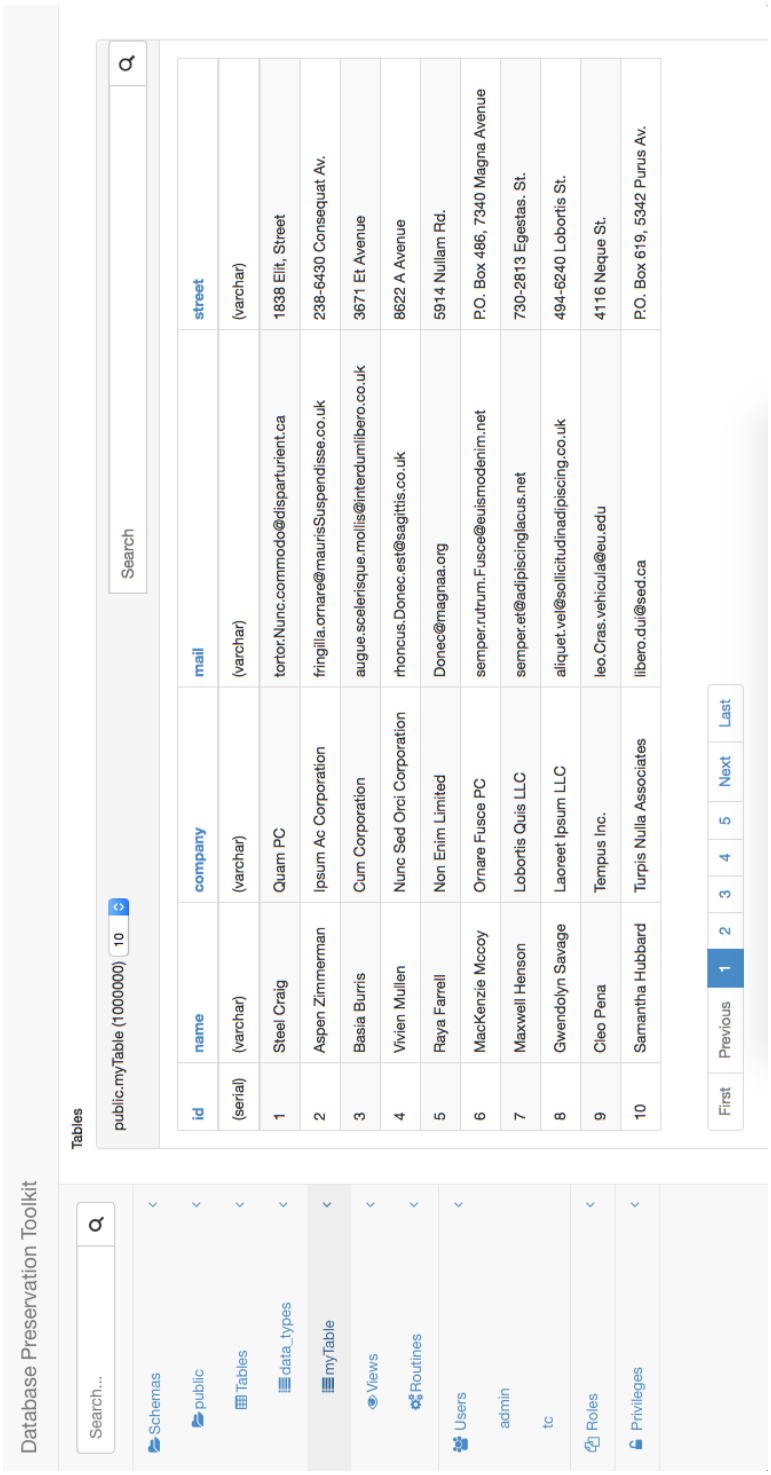


Figura 5.1: Screenshot da interface do visualizador **db-preservation-toolkit**

cutado, trocando o valor do `facet.field` para "dbpres_meta_table".

Além disso o parâmetro `q` teria de passar de `*:*` para `dbpres_meta_schema:<nome_do_schema>`. Isto significa que em vez de tomar em consideração todos os documentos (definido por `*:*`), apenas os documentos que cumpram a condição, nesta caso, do seu *schema* ser igual ao `<nome_do_schema>` dado, serão avaliados [Kuc13].

5.4.2 Informação na área de visualização de dados

No que diz respeito à informação tabular, foi criado um serviço de dados partilhado que permite mostrar as linhas de uma tabela dependendo do estado em que a aplicação se encontrar. Assim, por defeito, quando, através do menu, se seleciona o nome de uma tabela, serão apresentadas as dez primeiras linhas dessa tabela [Gre13].

Contudo, ao longo do uso da aplicação, por exemplo, ao mudar de página, ao ordenar por coluna, ao fazer uma pesquisa ou ao mudar o número de linhas a ver por página, o estado da aplicação muda, sendo apresentadas as linhas da tabela que se adequam a tais condições. Isto acontece devido ao serviço de dados partilhado que disponibiliza a informação sobre o estado da aplicação aos vários controladores da mesma.

Os valores numéricos presentes no fim das chaves "dbpres_data_*", "dbpres_meta_col_*" e "dbpres_meta_colType_*", servem, entre outras coisas (como por exemplo, saber a ordem original das colunas), para manter a ligação entre da informação e a coluna a que pertence essa informação. Assim, a informação de "dbpres_data_1" pertence à coluna "dbpres_meta_col_1", sendo o tipo de dados dessa coluna o valor de "dbpres_meta_colType_1".

No que diz respeito à pesquisa de pesquisa de dados, esta é feita em *real-time*. Isto é, quando o utilizador digita um termo a pesquisar, são imediatamente retornadas as linhas da tabela que apresentam esse termo. Assim, no excerto 5.5 é possível ver um exemplo de parâmetros de uma *query* Solr de pesquisa simples, isto é, o termo é pesquisado em todas as colunas de uma tabela.

```
"params":{
  "q":"dbpres_meta_tableId:public.myTable AND *avenue*",
  "fl":"dbpres_data_*",
  "start": "10"
  "rows":"10",
  "sort": "dbpres_data_4 ASC"
}
```

Listagem 5.5: Parâmetros enviados numa pesquisa de informação

Analisando os parâmetros apresentados no excerto de cima temos o seguinte:

- `q`: define que os documentos (linhas de uma tabela) a devolver devem ter presentes na chave "dbpres_meta_tableId" o valor "public.myTable". Esta chave permite identificar de forma única uma tabela.

  Além disso, outra condição que os documentos devem cumprir é terem presentes num dos campos "dbpres_data_*" o valor *avenue*, ou seja, esse termo tem de estar presente numa coluna da linha de uma certa coluna naquela linha da tabela. Quando se pesquisa um termo, sem especificar o campo em que o mesmo deve ser pesquisado, este é pesquisado nos campos "dbpres_data_*", já que a sua informação está disponível devido à ação do elemento <`copyField`> presente no *schema.xml*, já referido anteriormente.

  É importante referir que os asteriscos presentes antes e depois do termo a pesquisar, permitem que o termo seja encontrado sem que seja feito *match* exato do termo no documento [Foub].

- `fl`: faz retornar apenas os campos "dbpres_data_*".

- `start` e `rows`: em conjunto, estes parâmetros refletem que devem ser retornadas o segundo conjunto de 10 linhas da tabela (atendendo às condições impostas pelos outros parâmetros: `q` e `sort`).

  São este os parâmetros usados quando se quer mostrar a segunda página em que são exibidas 10 linhas.

- `sort`: retorna os documentos ordenados ascendentemente de acordo com o a coluna 4 da tabela.

Do mesmo modo que os parâmetros apresentados em 5.5 permitem a pesquisa de um termo em todas as colunas, é possível especificar os termos que devem ser encontrados em cada coluna, tornando assim possível uma pesquisa mais particular.

Assim, no parâmetro `q`, teria-se, por exemplo o apresentado em 5.6, sendo possível adicionar mais condições relativas a outras colunas [Foub].

```
"params":{
```

```
  "q":"dbpres_meta_tableId:public.myTable AND
     ↪ dbpres_data_2:*William* AND dbpres_data_3:*edu*",
  ...
}
```

Listagem 5.6: Parâmetros enviados numa pesquisa de informação de termos em colunas específicas

Capítulo 6

Conclusões e Trabalho Futuro

Neste capítulo final são apresentadas algumas conclusões. Além disso, são abordados alguns tópicos que podem servir de trabalho futuro.

6.1 Conclusões

O principal objetivo desta dissertação era melhorar a ferramenta de preservação de bases de dados, **db-preservation-toolkit**.

Assim, de modo a conseguir alcançar esse grande objetivo, foram delineadas certas metas a atingir. Entre elas encontravam-se a adição do suporte do formato SIARD, bem como a adição de novos módulos que suportem outros sistemas de gestão de bases de dados. Outra dessas metas passava pelo facto de criar um visualizador quer permitisse a exploração e pesquisa da informação de uma base de dados arquivada.

Desta forma, primeiramente, e dado ser onde grande parte das alterações necessárias a fazer na aplicação se encontravam, foram feitos os desenvolvimentos precisos de forma a adicionar o suporte ao formato SIARD.

A adição do suporte ao formato SIARD, envolveu, contudo, a necessidade da alteração de componentes importantes do **db-preservation-toolkit**, como é o caso do seu modelo de dados intermédio, e consequentemente levou à necessidade refazer os módulos de importação/exportação já existentes.

Concluídas estas modificações significativas, adicionado novamente o suporte aos sistemas de gestão de bases de dados previamente suportados e adicionado o suporte ao formato SIARD, foi altura de adicionar o suporte ao sistemas de gestão de bases de dados DB2.

Já numa fase posterior, foi desenvolvido o visualizador, havendo para

isso a necessidade de investigar as tecnologias que melhor se enquadrariam para levar a cabo este trabalho, como são o caso do Solr e do Angular.js.

Concluindo, dado o grande objetivo que passava por melhorar a ferramenta de preservação de bases de dados, **db-preservation-toolkit**, através das metas estabelecidas, pode-se afirmar que tal objetivo foi cumprido. Assim, o **db-preservation-toolkit**, que se apresenta como uma ferramenta *open source*, não tendo, por isso, a si associados quaisquer custos, é uma opção a quem não está disposto/não pode pagar para usufruir do CHRONOS.

Além disso, através do seu serviço de visualização e pesquisa, oferece a possibilidade de explorar e pesquisar uma base de dados preservada no formato SIARD (ou qualquer tipo ficheiros/base de dados que sejam suportados pelos módulos de importação). Deste modo, apresenta vantagens em relação ao SIARD Suite, que não permite que a informação tabular de uma base de dados seja pesquisada.

Por fim, a variedade de sistemas de gestão de base de dados que o **db-preservation-toolkit** suporta também se apresenta como uma vantagem relevante. Este fatores permitem afirmar que o **db-preservation-toolkit**, é agora uma aplicação mais madura, com capacidades para ser considera uma opção no que diz respeito à preservação de bases de dados.

6.2 Trabalho Futuro

Além das funcionalidades adicionadas ao **db-preservation-toolkit**, há ainda espaço para a implementação de mais melhorias:

- Paralelizar a aplicação, de modo a tirar o máximo de partido dos recursos de uma máquina.
- Adicionar suporte a outros sistemas de gestão de bases de dados e/ou diferentes versões desses mesmos sistemas de gestão de bases de dados.
- Adicionar, no visualizador, a possibilidade de criar vistas específicas e que as mesmas sejam guardas e exportadas em diferentes formatos, como CSV, por exemplo.
- Identificar claramente que informação é perdida no processo de preservação de uma base de dados.

- Implementar de mecanismo que permita o uso de *queries* SQL para obter informação de uma base de dados a ser mostrado no visualizador.
- Adicionar mecanismo de gestão de bases de dados arquivadas. Isto é, permitir que sejam tirados *snapshots* ao estado de uma base da dados e que sejam visualizadas diferenças entre as várias alturas em que a mesma base de dados foi arquivada.

Bibliografia

[Ald13] Carlos Filipe Pereira Aldeias. Open archival information systems for database preservation, 2013.

[Ber09] Amir Bernstein. Archiving relational databases with siard suite, 2009.

[BKM07] Stefan Brandl and Dr. Peter Keller-Marxer. Long-term archiving of relational databases with chronos. 2007.

[BMI12] Andrzej Bialecki, Robert Muri, and Grant Ingersoll. Apache Lucene 4. In *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 17–24, 2012.

[Bra14] Rodrigo Branas. *AngularJS Essentials*. Packt Publishing, 2014.

[CSP] CSP. Chronos webpage. `http://www.csp-sw.de/en/inhalt.php?kategorie=c271_Solutions_CHRONOS`. Accessed: 2013-12.

[Dai] Denny C. Dai. Lucene.js github page. https://github.com/dennycd/lucene.js. Accessed: 2014-08.

[DAN] DANS. Mixed webpage. `https://sites.google.com/a/datanetworkservice.nl/mixed/`. Accessed: 2013-12.

[ERRD11] Mette van Essen, Maurice de Rooij, Bill Roberts, and Maurice van den Dobbelsteen. Database preservation case study: Review. 2011.

[FCF+07] Luís Faria, Rui Castro, Miguel Ferreira, José Carlos Ramalho, and Francisco Barbedo. Roda - repository of authentic digital objects. 2007.

[Fer06] Miguel Ferreira. *Introdução à preservação digital : conceitos, estratégias e actuais consensos*. Universidade do Minho. Escola de Engenharia, 2006.

[Fit13] Neal Fitzgerald. Using data archiving tools to preserve archival records in business systems – a case study. 2013.

[Foua] Apache Software Foundation. Lucene webpage. http://lucene.apache.org. Accessed: 2014-07.

[Foub] Apache Software Foundation. Solr webpage. https://lucene.apache.org/solr/. Accessed: 2014-07.

[GP99] Peter Gulutzan and Trudy Pelzer. *SQL-99 Complete, Really*. CMP Books, 1999.

[GP14] Trey Grainger and Timothy Potter. *Solr in Action*. Manning Publications Co., 2014.

[Gre13] Brad Green. *AngularJS*. O'Reilly Media, 2013.

[JLRH02] Marta H. Jacinto, Giovani R. Librelotto, José C. Ramalho, and Pedro R. Henriques. Bidirectional conversion between xml documents and relational data bases. 2002.

[Kuc13] Rafal Kuc. *Apache Solr 4 Cookbook*. Packt Publishing, 2013.

[Lav14] Jim Lavin. *AngularJS Services*. Packt Publishing, 2014.

[Lin13] Andrew Lindley. Database preservation evaluation report - siard vs. chronos:preserving complex structures as databases through a record centric approach. 2013.

[McD] Fergus McDowall. Norch github page. https://github.com/fergiemcdowall/norch. Accessed: 2014-08.

[MHG10] Michael McCandless, Erik Hatcher, and Otis Gospodnetić. *Lucene in Action, Second Edition*. Manning Publications Co., 2010.

[Nay14] Mathieu Nayrolles. *Mastering Apache Solr: A practical guide to get to grips with Apache Solr*. Inkstall Solutions LLC, 2014.

[Nig] Oliver Nightingale. Lunr webpage. http://lunrjs.com. Accessed: 2014-08.

[papdsdi11] Associação para a promoção da sociedade da informação. Glossário da sociedade da informação. Technical report, Associação para a promoção da sociedade da informação, 2011.

[Rah] Arif Ur Rahman. Transformation Rules for Model Migration in Relational Database Preservation. pages 1–12.

[Ram12a] José Carlos Ramalho. Database migration : Cli, 02 2012.

[Ram12b] José Carlos Ramalho. Roda : a service-oriented digital repository : database archiving, 02 2012.

[Rau12] Joachim Rausch. Database archiving with siard - experiences of the federal archives, 2012.

[RF10] José Carlos Ramalho and Ricardo André Pereira Freitas. Significant properties in the preservation of relational databases. 09 2010.

[RFFR07] José Carlos Ramalho, Miguel Ferreira, Luís Faria, and Castro Rui. Relational database preservation through xml modelling relational database preservation through xml modelling. 2007.

[SFAa] Swiss Federal Archives SFA. Sfa - archiving of databases: Siard suite. `http://www.bar.admin.ch/dienstleistungen/00823/00825/index.html?lang=en`. Accessed: 2014-07.

[SFAb] Swiss Federal Archives SFA. The siard relational database archiving solution.

[SFAc] Swiss Federal Archives SFA. Siard webpage. `http://www.bar.admin.ch/dienstleistungen/00823/00825/index.html?lang=en`. Accessed: 2013-12.

[SFA11] Swiss Federal Archives SFA. Siard format description, 2011.

[Tana] Kelvin Tan. Lucenetutorial.com webpage. http://www.lucenetutorial.com. Accessed: 2014-08.

[Tanb] Kelvin Tan. Solrtutorial.com webpage. http://www.solrtutorial.com. Accessed: 2014-08.

[Tea] AngularJS Team. Angularjs webpage. https://angularjs.org. Accessed: 2014-08.

[Tho13] Hartwig Thomas. Siard suite manual. 2013.

[Wika] Wikipedia. Database. http://en.wikipedia.org/wiki/Database. Accessed: 2013-12.

[Wikb] Wikipedia. Sql. http://en.wikipedia.org/wiki/SQL. Accessed: 2013-12.

Apêndice A

Exemplo de ficheiro DBML

```
<?xml version="1.0" encoding="UTF-8"?>
<db name="test2" exportDate="2014-10-30T15:40:32Z"
    ↪ productName="PostgreSQL" productVersion="9.3.4"
    ↪ schemaVersion="0.2">
  <structure>
    <table id="mock_data" name="mock_data">
      <columns>
      <column id="mock_data.id" name="id" nillable="true"
    ↪ >
        <type originalTypeName="int4">
          <simpleTypeNumericExact precision="10" scale="0
    ↪ "/>
        </type>
      </column>
      <column id="mock_data.first_name" name="first_name"
    ↪  nillable="true">
        <type originalTypeName="varchar">
          <simpleTypeString length="50" variableLegth="
    ↪ true"/>
        </type>
      </column>
      <column id="mock_data.last_name" name="last_name"
    ↪ nillable="true">
        <type originalTypeName="varchar">
          <simpleTypeString length="50" variableLegth="
    ↪ true"/>
        </type>
      </column>
      <column id="mock_data.email" name="email" nillable=
    ↪ "true">
        <type originalTypeName="varchar">
          <simpleTypeString length="50" variableLegth="
    ↪ true"/>
        </type>
```

```
    </column>
    <column id="mock_data.country" name="country"
↪  nillable="true">
      <type originalTypeName="varchar">
        <simpleTypeString length="50" variableLegth="
↪  true"/>
      </type>
    </column>
    <column id="mock_data.ip_address" name="ip_address"
↪   nillable="true">
      <type originalTypeName="varchar">
        <simpleTypeString length="20" variableLegth="
↪  true"/>
      </type>
    </column>
    </columns>
    <keys>
    </keys>
  </table>
  <table id="mytable" name="mytable">
    <columns>
    <column id="mytable.id" name="id" nillable="false">
      <type originalTypeName="serial">
        <simpleTypeNumericExact precision="10" scale="0
↪  "/>
      </type>
    </column>
    <column id="mytable.name" name="name" nillable="
↪  true">
      <type originalTypeName="varchar">
        <simpleTypeString length="255" variableLegth="
↪  true"/>
      </type>
    </column>
    <column id="mytable.company" name="company"
↪  nillable="true">
      <type originalTypeName="varchar">
        <simpleTypeString length="255" variableLegth="
↪  true"/>
      </type>
    </column>
    <column id="mytable.mail" name="mail" nillable="
↪  true">
      <type originalTypeName="varchar">
        <simpleTypeString length="255" variableLegth="
↪  true"/>
      </type>
    </column>
    <column id="mytable.street" name="street" nillable=
↪  "true">
      <type originalTypeName="varchar">
```

```
        <simpleTypeString length="255" variableLegth="
↪ true"/>
      </type>
    </column>
    </columns>
    <keys>
    <pkey type="SIMPLE">
      <field name="id"/>
    </pkey>
    </keys>
  </table>
  <table id="test_table" name="test_table">
    <columns>
    <column id="test_table.id" name="id" nillable="true
↪ ">
      <type originalTypeName="int4">
        <simpleTypeNumericExact precision="10" scale="0
↪ "/>
      </type>
    </column>
    <column id="test_table.name" name="name" nillable="
↪ true">
      <type originalTypeName="varchar">
        <simpleTypeString length="240" variableLegth="
↪ true"/>
      </type>
    </column>
    </columns>
    <keys>
    </keys>
  </table>
</structure>
<data>
  <tableData id="mock_data">
    <row id="1">
      <cell id="mock_data.id.1">
        <s>1</s>
      </cell>
      <cell id="mock_data.first_name.1">
        <s>Juan</s>
      </cell>
      <cell id="mock_data.last_name.1">
        <s>Rivera</s>
      </cell>
      <cell id="mock_data.email.1">
        <s>jrivera0@parallels.com</s>
      </cell>
      <cell id="mock_data.country.1">
        <s>Antarctica</s>
      </cell>
      <cell id="mock_data.ip_address.1">
```

```
      <s>98.18.173.23</s>
    </cell>
  </row>
  <row id="2">
    <cell id="mock_data.id.2">
      <s>2</s>
    </cell>
    <cell id="mock_data.first_name.2">
      <s>Rose</s>
    </cell>
    <cell id="mock_data.last_name.2">
      <s>Barnes</s>
    </cell>
    <cell id="mock_data.email.2">
      <s>rbarnes1@ucoz.com</s>
    </cell>
    <cell id="mock_data.country.2">
      <s>Wallis and Futuna Islands</s>
    </cell>
    <cell id="mock_data.ip_address.2">
      <s>236.209.35.45</s>
    </cell>
  </row>
  <row id="3">
    <cell id="mock_data.id.3">
      <s>3</s>
    </cell>
    <cell id="mock_data.first_name.3">
      <s>Alan</s>
    </cell>
    <cell id="mock_data.last_name.3">
      <s>Rice</s>
    </cell>
    <cell id="mock_data.email.3">
      <s>arice2@google.es</s>
    </cell>
    <cell id="mock_data.country.3">
      <s>Guatemala</s>
    </cell>
    <cell id="mock_data.ip_address.3">
      <s>25.32.29.233</s>
    </cell>
  </row>
  <row id="4">
    <cell id="mock_data.id.4">
      <s>4</s>
    </cell>
    <cell id="mock_data.first_name.4">
      <s>Tina</s>
    </cell>
    <cell id="mock_data.last_name.4">
```

```
      <s>Henderson</s>
    </cell>
    <cell id="mock_data.email.4">
      <s>thenderson3@skype.com</s>
    </cell>
    <cell id="mock_data.country.4">
      <s>Northern Mariana Islands</s>
    </cell>
    <cell id="mock_data.ip_address.4">
      <s>240.193.56.61</s>
    </cell>
  </row>
  <row id="5">
    <cell id="mock_data.id.5">
      <s>5</s>
    </cell>
    <cell id="mock_data.first_name.5">
      <s>Irene</s>
    </cell>
    <cell id="mock_data.last_name.5">
      <s>Hall</s>
    </cell>
    <cell id="mock_data.email.5">
      <s>ihall4@google.com.au</s>
    </cell>
    <cell id="mock_data.country.5">
      <s>Equatorial Guinea</s>
    </cell>
    <cell id="mock_data.ip_address.5">
      <s>0.165.7.130</s>
    </cell>
  </row>
  <row id="6">
    <cell id="mock_data.id.6">
      <s>6</s>
    </cell>
    <cell id="mock_data.first_name.6">
      <s>Randy</s>
    </cell>
    <cell id="mock_data.last_name.6">
      <s>Watkins</s>
    </cell>
    <cell id="mock_data.email.6">
      <s>rwatkins5@quantcast.com</s>
    </cell>
    <cell id="mock_data.country.6">
      <s>Yugoslavia</s>
    </cell>
    <cell id="mock_data.ip_address.6">
      <s>179.76.224.70</s>
    </cell>
```

```
    </row>
    <!-- more rows -->
  </tableData>
  <tableData id="mytable">
    <row id="1">
      <cell id="mytable.id.1">
        <s>1</s>
      </cell>
      <cell id="mytable.name.1">
        <s>Beau Bean</s>
      </cell>
      <cell id="mytable.company.1">
        <s>Ut Associates</s>
      </cell>
      <cell id="mytable.mail.1">
        <s>mattis.Cras@Quisqueporttitoreros.com</s>
      </cell>
      <cell id="mytable.street.1">
        <s>P.O. Box 342, 2054 Dui. Rd.</s>
      </cell>
    </row>
    <row id="2">
      <cell id="mytable.id.2">
        <s>2</s>
      </cell>
      <cell id="mytable.name.2">
        <s>Robert Cooley</s>
      </cell>
      <cell id="mytable.company.2">
        <s>Enim Curabitur Associates</s>
      </cell>
      <cell id="mytable.mail.2">
        <s>Nunc@commodo.net</s>
      </cell>
      <cell id="mytable.street.2">
        <s>812-6194 Aliquam Ave</s>
      </cell>
    </row>
    <row id="3">
      <cell id="mytable.id.3">
        <s>3</s>
      </cell>
      <cell id="mytable.name.3">
        <s>Malik Bray</s>
      </cell>
      <cell id="mytable.company.3">
        <s>Luctus Ltd</s>
      </cell>
      <cell id="mytable.mail.3">
        <s>lorem@acmattissemper.co.uk</s>
      </cell>
```

```
        <cell id="mytable.street.3">
          <s>875-9577 Urna. Road</s>
        </cell>
      </row>
      <row id="4">
        <cell id="mytable.id.4">
          <s>4</s>
        </cell>
        <cell id="mytable.name.4">
          <s>Isaiah Parsons</s>
        </cell>
        <cell id="mytable.company.4">
          <s>Semper Tellus Id Corp.</s>
        </cell>
        <cell id="mytable.mail.4">
          <s>ridiculus.mus@ligulaNullam.org</s>
        </cell>
        <cell id="mytable.street.4">
          <s>P.O. Box 621, 3718 Sed, Street</s>
        </cell>
      </row>
      <row id="5">
        <cell id="mytable.id.5">
          <s>5</s>
        </cell>
        <cell id="mytable.name.5">
          <s>Jael Hester</s>
        </cell>
        <cell id="mytable.company.5">
          <s>Faucibus Orci Luctus Inc.</s>
        </cell>
        <cell id="mytable.mail.5">
          <s>sed@sitamet.co.uk</s>
        </cell>
        <cell id="mytable.street.5">
          <s>P.O. Box 819, 9337 Est Street</s>
        </cell>
      </row>
      <!-- more rows -->
    </tableData>
  </data>
</db>
```

Listagem A.1: Exemplo de ficheiro XML

Apêndice B

Schema do DBML: dbml-1.1.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ elementFormDefault="qualified">

  <xs:element name="type">
    <xs:annotation>
      <xs:documentation>The type of the column. A type
    ↪ can be simple or composed, composed
        types can be structure or array. Types are
    ↪ defined based on SQL:1999
      standard.</xs:documentation></xs:annotation>
    <xs:complexType>
      <xs:choice>
        <xs:element ref="simpleTypeString"/>
        <xs:element ref="simpleTypeNumericExact"/>
        <xs:element ref="simpleTypeNumericApproximate"/>
        <xs:element ref="simpleTypeBoolean"/>
        <xs:element ref="simpleTypeEnumeration"/>
        <xs:element ref="simpleTypeDateTime"/>
        <xs:element ref="simpleTypeInterval"/>
        <xs:element ref="simpleTypeBinary"/>
        <xs:element ref="composedTypeStructure"/>
        <xs:element ref="composedTypeArray"/>
      </xs:choice>
      <xs:attribute name="originalTypeName" use="optional
    ↪ "/>
      <xs:attribute name="description" use="optional"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="composedTypeArray">
    <xs:complexType>
      <xs:sequence>
```

```
      <xs:element ref="type" minOccurs="1" maxOccurs="1
↪ "/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="composedTypeStructure">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="type" minOccurs="1" maxOccurs="
↪ unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="simpleTypeString">
  <xs:annotation>
    <xs:documentation>Sequence of charecters drawn from
↪  sobre character repertoire (charset)
      This sequence is is either of fixed length, or of
↪  variable length up to some
      implementation-defined maximum</xs:documentation>
↪ </xs:annotation>
  <xs:complexType>
    <xs:attribute name="length" type="
↪ xs:nonNegativeInteger" use="required"/>
    <xs:attribute name="variableLegth" type="xs:boolean
↪ " use="required"/>
    <xs:attribute name="charSet" type="xs:string" use="
↪ optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="simpleTypeNumericExact">
  <xs:annotation>
    <xs:documentation>An exact numeric includes integer
↪  types and types with specified
      precision (number of digits) and scale (digits
↪ after the radix
    point)</xs:documentation></xs:annotation>
  <xs:complexType>
    <xs:attribute name="precision" type="
↪ xs:nonNegativeInteger" use="optional"/>
    <xs:attribute name="scale" type="
↪ xs:nonNegativeInteger" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="simpleTypeNumericApproximate">
  <xs:annotation>
```

```
    <xs:documentation>An approximate numeric is
  ↪ essentially a floating point and for each a
      precision may be optionally specified</
  ↪ xs:documentation></xs:annotation>
   <xs:complexType>
    <xs:attribute name="precision" type="
  ↪ xs:nonNegativeInteger" use="optional"/>
   </xs:complexType>
</xs:element>

<xs:element name="simpleTypeBoolean">
   <xs:annotation>
    <xs:documentation>A value of the Boolean data type
  ↪ is either true or false. The truth
      value of unknown is sometimes represented by the
  ↪ null value.</xs:documentation></xs:annotation>
</xs:element>

<xs:element name="simpleTypeEnumeration">
   <xs:annotation>
    <xs:documentation>A list of possible values for
  ↪ this field. Each value is represented by
      a string</xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="simpleTypeEnumerationOption"
  ↪ minOccurs="2" maxOccurs="unbounded"
      />
    </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="simpleTypeEnumerationOption">
   <xs:complexType>
    <xs:attribute name="name" type="xs:string" use="
  ↪ required"/>
    <xs:attribute name="description" use="optional"/>
   </xs:complexType>
</xs:element>

<xs:element name="simpleTypeDateTime">
   <xs:annotation><xs:documentation>Date and time
  ↪ according to ISO 8601.</xs:documentation></
  ↪ xs:annotation>
   <xs:complexType>
    <xs:attribute name="timeDefined" type="xs:boolean"
  ↪ use="required"/>
    <xs:attribute name="timeZoneDefined" type="
  ↪ xs:boolean" use="required"/>
   </xs:complexType>
```

```
</xs:element>

<xs:element name="simpleTypeInterval">
  <xs:annotation><xs:documentation>Time intervals
 ↪ according to ISO 8601.</xs:documentation></
 ↪ xs:annotation>
  <xs:complexType>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="START_END"/>
          <xs:enumeration value="START_DURATION"/>
          <xs:enumeration value="DURATION_END"/>
          <xs:enumeration value="DURATION"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="simpleTypeBinary">
  <xs:annotation>
    <xs:documentation>A value of binary string type (
 ↪ known as a binary large object, or
      BLOB) is a variable length sequence of octets, up
 ↪  to an implementation-defined
      maximum.</xs:documentation></xs:annotation>
  <xs:complexType>
    <xs:attribute name="formatRegistryName" type="
 ↪ xs:string" use="optional"/>
    <xs:attribute name="formatRegistryKey" type="
 ↪ xs:string" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="column">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="type" minOccurs="1" maxOccurs="1
 ↪ "/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
    <xs:attribute name="name" use="required"/>
    <xs:attribute name="nillable" type="xs:boolean" use
 ↪ ="optional"/>
    <xs:attribute name="description" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="columns">
  <xs:complexType>
```

```
    <xs:sequence>
      <xs:element ref="column" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="field">
  <xs:complexType>
    <xs:attribute name="name" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="fkey">
  <xs:complexType>
    <xs:attribute name="id" use="required"/>
    <xs:attribute name="name" use="required"/>
    <xs:attribute name="in" use="required"/>
    <xs:attribute name="ref" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="pkey">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="field" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="COMPOSITE"/>
          <xs:enumeration value="SIMPLE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="keys">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="pkey" minOccurs="0" maxOccurs="1
↪ "/>
      <xs:element ref="fkey" minOccurs="0" maxOccurs="
↪ unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="table">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element ref="columns" minOccurs="0"/>
      <xs:element ref="keys" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
    <xs:attribute name="name" use="required"/>
    <xs:attribute name="description" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="structure">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="table" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="db">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="structure"/>
      <xs:element ref="data"/>
    </xs:sequence>
    <xs:attribute name="name" use="required"/>
    <xs:attribute name="creationDate"  type="
↪ xs:dateTime" use="optional"/>
    <xs:attribute name="exportDate" type="xs:dateTime"
↪ use="optional"/>
    <xs:attribute name="productName" type="xs:string"
↪ use="optional"/>
    <xs:attribute name="productVersion" type="xs:string
↪ " use="optional"/>
    <xs:attribute name="
↪ defaultTransactionIsolationLevel" type="xs:int" use
↪ ="optional"/>
    <xs:attribute name="extraNameCharacters" type="
↪ xs:string" use="optional"/>
    <xs:attribute name="stringFunctions" type="
↪ xs:string" use="optional"/>
    <xs:attribute name="systemFunctions" type="
↪ xs:string" use="optional"/>
    <xs:attribute name="timeDateFunctions" type="
↪ xs:string" use="optional"/>
    <xs:attribute name="url" type="xs:string" use="
↪ optional"/>
    <xs:attribute name="supportsANSI92EntryLevelSQL"
↪ type="xs:boolean" use="optional"/>
    <xs:attribute name="supportsANSI92IntermediateSQL"
↪ type="xs:boolean" use="optional"/>
    <xs:attribute name="supportsANSI92FullSQL" type="
↪ xs:boolean" use="optional"/>
```

```
    <xs:attribute name="supportsCoreSQLGrammar" type="
  ↪ xs:boolean" use="optional"/>
    <xs:attribute name="schemaVersion" type="xs:string"
  ↪  use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="data">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tableData" maxOccurs="unbounded"
  ↪ />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="tableData">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="row" minOccurs="0" maxOccurs="
  ↪ unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="row">
  <xs:annotation>
    <xs:documentation>Container for the row data. A row
  ↪  has a list of
    cells.</xs:documentation></xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="cell" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="cell">
  <xs:annotation>
    <xs:documentation>Container for the cell data. The
  ↪ data correspondes to a simple or
      composite type. All types are nillable.</
  ↪ xs:documentation></xs:annotation>
  <xs:complexType>
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:element ref="s"/>
      <xs:element ref="c"/>
      <xs:element ref="b"/>
    </xs:choice>
```

```
      <xs:attribute name="id" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="s" type="xs:string" nillable="true">
    <xs:annotation>
      <xs:documentation>Container for simple or
    ↪ predefined data like string, integers,
        booleans, dates, etc.</xs:documentation></
    ↪ xs:annotation>
  </xs:element>

  <xs:element name="b" nillable="true">
    <xs:annotation>
      <xs:documentation>Container for simple or
    ↪ predefined data like string, integers,
      booleans, dates, etc.</xs:documentation></
    ↪ xs:annotation>
    <xs:complexType>
      <xs:attribute name="file" type="xs:string" use="
    ↪ required"/>
      <xs:attribute name="formatRegistryName" type="
    ↪ xs:string" use="optional"/>
      <xs:attribute name="formatRegistryKey" type="
    ↪ xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="c" nillable="true">
    <xs:annotation>
      <xs:documentation>Container for composite data like
    ↪  arrays or
        structures</xs:documentation></xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:choice minOccurs="1" maxOccurs="unbounded">
          <xs:element ref="s"/>
          <xs:element ref="c"/>
          <xs:element ref="b"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listagem B.1: *Schema* do DBML: dbml-1.1.xsd

Apêndice C

SIARD: exemplo *metadata.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="metadata.xsl"?>

<siardArchive xmlns="http://www.bar.admin.ch/xmlns/siard
    ↪ /1.0/metadata.xsd" xmlns:xsi="http://www.w3.org
    ↪ /2001/XMLSchema-instance" version="1.0"
    ↪ xsi:schemaLocation="http://www.bar.admin.ch/xmlns/
    ↪ siard/1.0/metadata.xsd metadata.xsd">
  <dbname>crm</dbname>
  <dataOwner>(...)</dataOwner>
  <dataOriginTimespan>(...)</dataOriginTimespan>
  <producerApplication>SiardEdit 1.19 Swiss Federal
    ↪ Archives, Berne, Switzerland, 2007, 2008</
    ↪ producerApplication>
  <archivalDate>2010-02-09</archivalDate>
  <messageDigest>MD5440AB2318FD146592223CF2A879ECAAF</
    ↪ messageDigest>
  <clientMachine>celsius.enterag.ch</clientMachine>
  <databaseProduct>
    Oracle Oracle9i Enterprise Edition Release 9.2.0.1.0
    ↪ -
    Production\u000AWith the Partitioning, OLAP and
    ↪ Oracle Data Mining
    options\u000AJServer Release 9.2.0.1.0 - Production
  </databaseProduct>
  <connection>jdbc:oracle:thin:@dbhost.enternet.
    ↪ ch:1521:SIARD1</connection>
  <databaseUser>CRM</databaseUser>
  <schemas>
    <schema>
      <name>CRM</name>
      <folder>schema0</folder>
      <tables>
        <table>
```

```
<name>ADDRESS</name>
<folder>table18</folder>
<description/>
<columns>
  <column>
    <name>AddressId</name>
    <type>DECIMAL(38)</type>
    <typeOriginal>NUMBER(38)</typeOriginal>
    <nullable>false</nullable>
  </column>
  <column>
    <name>Title</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>FirstNames</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>LastName</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Company</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Department</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Addition</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Address</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
```

```
</column>
<column>
  <name>Postbox</name>
  <type>CHARACTER VARYING(255)</type>
  <typeOriginal>VARCHAR2(255)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>ZipCode</name>
  <type>CHARACTER VARYING(10)</type>
  <typeOriginal>VARCHAR2(10)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>City</name>
  <type>CHARACTER VARYING(255)</type>
  <typeOriginal>VARCHAR2(255)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>PbZip</name>
  <type>CHARACTER VARYING(10)</type>
  <typeOriginal>VARCHAR2(10)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>PbCity</name>
  <type>CHARACTER VARYING(255)</type>
  <typeOriginal>VARCHAR2(255)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CountryId</name>
  <type>CHARACTER VARYING(2)</type>
  <typeOriginal>VARCHAR2(2)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CompanyPhone</name>
  <type>CHARACTER VARYING(255)</type>
  <typeOriginal>VARCHAR2(255)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>DirectPhone</name>
  <type>CHARACTER VARYING(255)</type>
  <typeOriginal>VARCHAR2(255)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>PrivatePhone</name>
```

```
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>MOBILEPHONE</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Fax</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Email</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Url</name>
    <type>CHARACTER VARYING(255)</type>
    <typeOriginal>VARCHAR2(255)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>LanguageId</name>
    <type>CHARACTER VARYING(1)</type>
    <typeOriginal>VARCHAR2(1)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Exclusions</name>
    <type>DECIMAL(38)</type>
    <typeOriginal>NUMBER(38)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>AddressTypes</name>
    <type>DECIMAL(38)</type>
    <typeOriginal>NUMBER(38)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>Subjects</name>
    <type>DECIMAL(38)</type>
    <typeOriginal>NUMBER(38)</typeOriginal>
    <nullable>true</nullable>
```

```
      </column>
      <column>
        <name>Comment</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>AuthorId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>ADDRESS_PK</name>
      <column>AddressId</column>
    </primaryKey>
    <rows>30336</rows>
  </table>
  <table>
    <name>ADDRESSCOLLECTION</name>
    <folder>table3</folder>
    <description/>
    <columns>
      <column>
        <name>AddressId</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>CollectionId</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>AuthorId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
```

```
            <type>TIMESTAMP</type>
            <typeOriginal>DATE</typeOriginal>
            <nullable>true</nullable>
          </column>
        </columns>
        <primaryKey>
          <name>ADDRESSCOLLECTION_PK</name>
          <column>AddressId</column>
          <column>CollectionId</column>
        </primaryKey>
        <foreignKeys>
          <foreignKey>
            <name>ADDRESSCOLLECTION_COLLECTION</name>
            <referencedSchema>CRM</referencedSchema>
            <referencedTable>COLLECTION</
↪ referencedTable>
            <reference>
              <column>CollectionId</column>
              <referenced>CollectionId</referenced>
            </reference>
            <deleteAction>RESTRICT</deleteAction>
            <updateAction>CASCADE</updateAction>
          </foreignKey>
        </foreignKeys>
        <rows>51626</rows>
      </table>
      <table>
        <name>ADDRESSTYPE</name>
        <folder>table11</folder>
        <description/>
        <columns>
          <column>
            <name>AddressTypeId</name>
            <type>DECIMAL(38)</type>
            <typeOriginal>NUMBER(38)</typeOriginal>
            <nullable>false</nullable>
          </column>
          <column>
            <name>Name</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>AuthorId</name>
            <type>CHARACTER VARYING(31)</type>
            <typeOriginal>VARCHAR2(31)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>LastModified</name>
```

```
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>ADDRESSTYPE_PK</name>
      <column>AddressTypeId</column>
    </primaryKey>
    <rows>5</rows>
  </table>
  <table>
    <name>AREA</name>
    <folder>table15</folder>
    <description/>
    <columns>
      <column>
        <name>AreaId</name>
        <type>CHARACTER VARYING(2)</type>
        <typeOriginal>VARCHAR2(2)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>AreaName</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>PartId</name>
        <type>CHARACTER VARYING(1)</type>
        <typeOriginal>VARCHAR2(1)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>AuthorId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>AREA_PK</name>
      <column>AreaId</column>
    </primaryKey>
```

```
      <foreignKeys>
        <foreignKey>
          <name>AREA_PART</name>
          <referencedSchema>CRM</referencedSchema>
          <referencedTable>PART</referencedTable>
          <reference>
            <column>PartId</column>
            <referenced>PartId</referenced>
          </reference>
          <deleteAction>RESTRICT</deleteAction>
          <updateAction>CASCADE</updateAction>
        </foreignKey>
      </foreignKeys>
      <rows>8</rows>
    </table>
    <table>
      <name>BRANCH</name>
      <folder>table12</folder>
      <description/>
      <columns>
        <column>
          <name>BranchId</name>
          <type>CHARACTER VARYING(2)</type>
          <typeOriginal>VARCHAR2(2)</typeOriginal>
          <nullable>false</nullable>
        </column>
        <column>
          <name>Name</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>AuthorId</name>
          <type>CHARACTER VARYING(31)</type>
          <typeOriginal>VARCHAR2(31)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>LastModified</name>
          <type>TIMESTAMP</type>
          <typeOriginal>DATE</typeOriginal>
          <nullable>true</nullable>
        </column>
      </columns>
      <primaryKey>
        <name>BRANCH_PK</name>
        <column>BranchId</column>
      </primaryKey>
      <rows>6</rows>
    </table>
```

```
<table>
  <name>CANTON</name>
  <folder>table16</folder>
  <description/>
  <columns>
    <column>
      <name>CantonId</name>
      <type>CHARACTER VARYING(2)</type>
      <typeOriginal>VARCHAR2(2)</typeOriginal>
      <nullable>false</nullable>
    </column>
    <column>
      <name>CantonGerman</name>
      <type>CHARACTER VARYING(255)</type>
      <typeOriginal>VARCHAR2(255)</typeOriginal>
      <nullable>true</nullable>
    </column>
    <column>
      <name>CantonFrench</name>
      <type>CHARACTER VARYING(255)</type>
      <typeOriginal>VARCHAR2(255)</typeOriginal>
      <nullable>true</nullable>
    </column>
    <column>
      <name>CantonItalian</name>
      <type>CHARACTER VARYING(255)</type>
      <typeOriginal>VARCHAR2(255)</typeOriginal>
      <nullable>true</nullable>
    </column>
    <column>
      <name>CantonEnglish</name>
      <type>CHARACTER VARYING(255)</type>
      <typeOriginal>VARCHAR2(255)</typeOriginal>
      <nullable>true</nullable>
    </column>
    <column>
      <name>AuthorId</name>
      <type>CHARACTER VARYING(31)</type>
      <typeOriginal>VARCHAR2(31)</typeOriginal>
      <nullable>true</nullable>
    </column>
    <column>
      <name>LastModified</name>
      <type>TIMESTAMP</type>
      <typeOriginal>DATE</typeOriginal>
      <nullable>true</nullable>
    </column>
  </columns>
  <primaryKey>
    <name>CANTON_PK</name>
    <column>CantonId</column>
```

```
      </primaryKey>
      <rows>27</rows>
    </table>
    <table>
      <name>COLLECTION</name>
      <folder>table8</folder>
      <description/>
      <columns>
        <column>
          <name>CollectionId</name>
          <type>DECIMAL(38)</type>
          <typeOriginal>NUMBER(38)</typeOriginal>
          <nullable>false</nullable>
        </column>
        <column>
          <name>Name</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>BranchId</name>
          <type>CHARACTER VARYING(2)</type>
          <typeOriginal>VARCHAR2(2)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>AuthorId</name>
          <type>CHARACTER VARYING(31)</type>
          <typeOriginal>VARCHAR2(31)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>LastModified</name>
          <type>TIMESTAMP</type>
          <typeOriginal>DATE</typeOriginal>
          <nullable>true</nullable>
        </column>
      </columns>
      <primaryKey>
        <name>COLLECTION_PK</name>
        <column>CollectionId</column>
      </primaryKey>
      <foreignKeys>
        <foreignKey>
          <name>COLLECTION_BRANCH</name>
          <referencedSchema>CRM</referencedSchema>
          <referencedTable>BRANCH</referencedTable>
          <reference>
            <column>BranchId</column>
            <referenced>BranchId</referenced>
```

```
        </reference>
        <deleteAction>RESTRICT</deleteAction>
        <updateAction>CASCADE</updateAction>
      </foreignKey>
    </foreignKeys>
    <candidateKeys>
      <candidateKey>
        <name>COLLECTION_UNIQUE</name>
        <column>Name</column>
      </candidateKey>
    </candidateKeys>
    <rows>102</rows>
  </table>
  <table>
    <name>CONSTRUCTION</name>
    <folder>table14</folder>
    <description/>
    <columns>
      <column>
        <name>CollectionId</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>ConstructionId</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>ActionId</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>ObjectType</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>ObjectId</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>ObjectTable</name>
        <type>CHARACTER VARYING(255)</type>
```

```
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>ObjectCondition</name>
            <folder>lob7</folder>
            <type>CHARACTER LARGE OBJECT</type>
            <typeOriginal>CLOB</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>Description</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>AuthorId</name>
            <type>CHARACTER VARYING(31)</type>
            <typeOriginal>VARCHAR2(31)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>LastModified</name>
            <type>TIMESTAMP</type>
            <typeOriginal>DATE</typeOriginal>
            <nullable>true</nullable>
          </column>
        </columns>
        <primaryKey>
          <name>CONSTRUCTION_PK</name>
          <column>CollectionId</column>
          <column>ConstructionId</column>
        </primaryKey>
        <foreignKeys>
          <foreignKey>
            <name>CONSTRUCTION_COLLECTION</name>
            <referencedSchema>CRM</referencedSchema>
            <referencedTable>COLLECTION</
↪ referencedTable>
            <reference>
              <column>CollectionId</column>
              <referenced>CollectionId</referenced>
            </reference>
            <deleteAction>RESTRICT</deleteAction>
            <updateAction>CASCADE</updateAction>
          </foreignKey>
        </foreignKeys>
        <rows>16607</rows>
      </table>
      <table>
```

```
    <name>COOPERATIVE</name>
    <folder>table13</folder>
    <description/>
    <columns>
      <column>
        <name>CooperativeId</name>
        <type>CHARACTER VARYING(4)</type>
        <typeOriginal>VARCHAR2(4)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>CooperativeName</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>AuthorId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>COOPERATIVE_PK</name>
      <column>CooperativeId</column>
    </primaryKey>
    <rows>10</rows>
  </table>
  <table>
    <name>COUNTRY</name>
    <folder>table5</folder>
    <description/>
    <columns>
      <column>
        <name>CountryId</name>
        <type>CHARACTER VARYING(2)</type>
        <typeOriginal>VARCHAR2(2)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>Name</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
```

```
        </column>
        <column>
          <name>AuthorId</name>
          <type>CHARACTER VARYING(31)</type>
          <typeOriginal>VARCHAR2(31)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>LastModified</name>
          <type>TIMESTAMP</type>
          <typeOriginal>DATE</typeOriginal>
          <nullable>true</nullable>
        </column>
      </columns>
      <primaryKey>
        <name>COUNTRY_PK</name>
        <column>CountryId</column>
      </primaryKey>
      <rows>178</rows>
    </table>
    <table>
      <name>CantonSource</name>
      <folder>table0</folder>
      <description/>
      <columns>
        <column>
          <name>CantonId</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>false</nullable>
        </column>
        <column>
          <name>CantonGerman</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>CantonFrench</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>CantonItalian</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>CantonEnglish</name>
```

```
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>CantonSource_PK</name>
      <column>CantonId</column>
    </primaryKey>
    <rows>27</rows>
  </table>
  <table>
    <name>DBUSER</name>
    <folder>table10</folder>
    <description/>
    <columns>
      <column>
        <name>DbUserId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>DbPassword</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>FirstNames</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastName</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>RoleId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>BranchId</name>
        <type>CHARACTER VARYING(2)</type>
        <typeOriginal>VARCHAR2(2)</typeOriginal>
        <nullable>true</nullable>
```

```
      </column>
      <column>
        <name>AuthorId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>DBUSER_PK</name>
      <column>DbUserId</column>
    </primaryKey>
    <foreignKeys>
      <foreignKey>
        <name>DBUSER_BRANCH</name>
        <referencedSchema>CRM</referencedSchema>
        <referencedTable>BRANCH</referencedTable>
        <reference>
          <column>BranchId</column>
          <referenced>BranchId</referenced>
        </reference>
        <deleteAction>RESTRICT</deleteAction>
        <updateAction>CASCADE</updateAction>
      </foreignKey>
    </foreignKeys>
    <rows>43</rows>
  </table>
  <table>
    <name>DUPLICATE</name>
    <folder>table17</folder>
    <description/>
    <columns>
      <column>
        <name>DuplicateId</name>
        <type>DECIMAL(38)</type>
        <typeOriginal>NUMBER(38)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>Company</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
```

```
            <name>FirstNames</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>LastName</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>Department</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>Addition</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>Address</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>City</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>ZipCode</name>
            <type>CHARACTER VARYING(10)</type>
            <typeOriginal>VARCHAR2(10)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>PostBox</name>
            <type>CHARACTER VARYING(255)</type>
            <typeOriginal>VARCHAR2(255)</typeOriginal>
            <nullable>true</nullable>
          </column>
        </columns>
        <rows>86</rows>
      </table>
      <table>
```

```
      <name>ISOCountry</name>
      <folder>table2</folder>
      <description/>
      <columns>
        <column>
          <name>CountryId</name>
          <type>CHARACTER VARYING(2)</type>
          <typeOriginal>VARCHAR2(2)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>FName</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>EName</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>DName</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
      </columns>
      <rows>178</rows>
    </table>
    <table>
      <name>PART</name>
      <folder>table1</folder>
      <description/>
      <columns>
        <column>
          <name>PartId</name>
          <type>CHARACTER VARYING(1)</type>
          <typeOriginal>VARCHAR2(1)</typeOriginal>
          <nullable>false</nullable>
        </column>
        <column>
          <name>PartName</name>
          <type>CHARACTER VARYING(255)</type>
          <typeOriginal>VARCHAR2(255)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>AuthorId</name>
          <type>CHARACTER VARYING(31)</type>
```

```
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>PART_PK</name>
      <column>PartId</column>
    </primaryKey>
    <rows>3</rows>
  </table>
  <table>
    <name>REGION</name>
    <folder>table9</folder>
    <description/>
    <columns>
      <column>
        <name>RegionId</name>
        <type>CHARACTER VARYING(4)</type>
        <typeOriginal>VARCHAR2(4)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>RegionName</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>AreaId</name>
        <type>CHARACTER VARYING(2)</type>
        <typeOriginal>VARCHAR2(2)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>AuthorId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
```

```
      </columns>
      <primaryKey>
        <name>REGION_PK</name>
        <column>RegionId</column>
      </primaryKey>
      <foreignKeys>
        <foreignKey>
          <name>REGION_AREA</name>
          <referencedSchema>CRM</referencedSchema>
          <referencedTable>AREA</referencedTable>
          <reference>
            <column>AreaId</column>
            <referenced>AreaId</referenced>
          </reference>
          <deleteAction>RESTRICT</deleteAction>
          <updateAction>CASCADE</updateAction>
        </foreignKey>
      </foreignKeys>
      <rows>18</rows>
    </table>
    <table>
      <name>SHIFT</name>
      <folder>table7</folder>
      <description/>
      <columns>
        <column>
          <name>AddressId</name>
          <type>DECIMAL(38)</type>
          <typeOriginal>NUMBER(38)</typeOriginal>
          <nullable>true</nullable>
        </column>
        <column>
          <name>PrevAddressId</name>
          <type>DECIMAL(38)</type>
          <typeOriginal>NUMBER(38)</typeOriginal>
          <nullable>true</nullable>
        </column>
      </columns>
      <rows>30336</rows>
    </table>
    <table>
      <name>SUBJECT</name>
      <folder>table6</folder>
      <description/>
      <columns>
        <column>
          <name>SubjectId</name>
          <type>DECIMAL(38)</type>
          <typeOriginal>NUMBER(38)</typeOriginal>
          <nullable>false</nullable>
        </column>
```

```
      <column>
        <name>Name</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>AuthorId</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2(31)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>LastModified</name>
        <type>TIMESTAMP</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>SUBJECT_PK</name>
      <column>SubjectId</column>
    </primaryKey>
    <rows>8</rows>
  </table>
  <table>
    <name>ZIPCODE</name>
    <folder>table4</folder>
    <description/>
    <columns>
      <column>
        <name>ZipCode</name>
        <type>CHARACTER VARYING(10)</type>
        <typeOriginal>VARCHAR2(10)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>City</name>
        <type>CHARACTER VARYING(255)</type>
        <typeOriginal>VARCHAR2(255)</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>CantonId</name>
        <type>CHARACTER VARYING(2)</type>
        <typeOriginal>VARCHAR2(2)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>RegionId</name>
        <type>CHARACTER VARYING(4)</type>
```

```
            <typeOriginal>VARCHAR2(4)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>CooperativeId</name>
            <type>CHARACTER VARYING(4)</type>
            <typeOriginal>VARCHAR2(4)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>AuthorId</name>
            <type>CHARACTER VARYING(31)</type>
            <typeOriginal>VARCHAR2(31)</typeOriginal>
            <nullable>true</nullable>
          </column>
          <column>
            <name>LastModified</name>
            <type>TIMESTAMP</type>
            <typeOriginal>DATE</typeOriginal>
            <nullable>true</nullable>
          </column>
        </columns>
        <primaryKey>
          <name>ZIPCODE_PK</name>
          <column>ZipCode</column>
          <column>City</column>
        </primaryKey>
        <foreignKeys>
          <foreignKey>
            <name>ZIPCODE_CANTON</name>
            <referencedSchema>CRM</referencedSchema>
            <referencedTable>CANTON</referencedTable>
            <reference>
              <column>CantonId</column>
              <referenced>CantonId</referenced>
            </reference>
            <deleteAction>RESTRICT</deleteAction>
            <updateAction>CASCADE</updateAction>
          </foreignKey>
          <foreignKey>
            <name>ZIPCODE_COOP</name>
            <referencedSchema>CRM</referencedSchema>
            <referencedTable>COOPERATIVE</
↪ referencedTable>
            <reference>
              <column>CooperativeId</column>
              <referenced>CooperativeId</referenced>
            </reference>
            <deleteAction>RESTRICT</deleteAction>
            <updateAction>CASCADE</updateAction>
          </foreignKey>
```

```
            <foreignKey>
              <name>ZIPCODE_REGION</name>
              <referencedSchema>CRM</referencedSchema>
              <referencedTable>REGION</referencedTable>
              <reference>
                <column>RegionId</column>
                <referenced>RegionId</referenced>
              </reference>
              <deleteAction>RESTRICT</deleteAction>
              <updateAction>CASCADE</updateAction>
            </foreignKey>
          </foreignKeys>
          <rows>4794</rows>
        </table>
      </tables>
    </schema>
  </schemas>
  <users>
    <user>
      <name>CRM</name>
    </user>
  </users>
  <roles>
    <role>
      <name>CONNECT</name>
      <admin/>
    </role>
    <role>
      <name>RESOURCE</name>
      <admin/>
    </role>
  </roles>
</siardArchive>
```

Listagem C.1: SIARD: exemplo *metadata.xml*

Apêndice D

Schema de *metadata.xml*: *metadata.xsd*

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- $Workfile: metadata.xsd $
    ↪ ********************************** -->
<!-- Metadata schema for SIARD 1.0
    ↪                 -->
<!-- Version     : $Id: metadata.xsd 1205 2010-06-17 16
    ↪ :54:52Z hartwig $ -->
<!-- Application: SIARD Suite
    ↪                 -->
<!--     Software-Independent Archival of Relational
    ↪ Databases         -->
<!-- Platform    : XML 1.0, XML Schema 2001
    ↪                 -->
<!-- Description: This XML schema definition defines the
    ↪ structure  -->
<!--     of the metadata in the SIARD format
    ↪                 -->
<!-- ********************************************** -->
<!-- Copyright   :  2007, Swiss Federal Archives, Berne,
    ↪ Switzerland -->
<!-- ********************************************** -->
<xs:schema id="metadata"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.bar.admin.ch/xmlns/siard/1.0/metadata
    ↪ .xsd"
  targetNamespace="http://www.bar.admin.ch/xmlns/siard
    ↪ /1.0/metadata.xsd"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
```

```
<!-- root element of an XML file conforming to this XML
 ↪  schema -->
<xs:element name="siardArchive">
  <xs:complexType>
    <xs:annotation>
      <xs:documentation>
        Root element of meta data of the SIARD archive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- name of the archived database -->
      <xs:element name="dbname" type="mandatoryString"/
 ↪ >
      <!-- short free form description of the database
 ↪ content -->
      <xs:element name="description" type="xs:string"
 ↪ minOccurs="0"/>
      <!-- name of person responsible for archiving the
 ↪  database -->
      <xs:element name="archiver" type="xs:string"
 ↪ minOccurs="0"/>
      <!-- contact data (telephone number or email
 ↪ address) of archiver -->
      <xs:element name="archiverContact" type="
 ↪ xs:string" minOccurs="0"/>
      <!-- name of data owner (section and institution
 ↪ responsible for data)
           of database when it was archived -->
      <xs:element name="dataOwner" type="
 ↪ mandatoryString"/>
      <!-- time span during which data where entered
 ↪ into the database -->
      <xs:element name="dataOriginTimespan" type="
 ↪ mandatoryString"/>
      <!-- name and version of program that generated
 ↪ the metadata file -->
      <xs:element name="producerApplication" type="
 ↪ xs:string" minOccurs="0"/>
      <!-- date of creation of archive (automatically
 ↪ generated by SIARD) -->
      <xs:element name="archivalDate" type="xs:date"/>
      <!-- message digest code over all primary data in
 ↪  folder "content" -->
      <xs:element name="messageDigest" type="digestType
 ↪ "/>
      <!-- DNS name of client machine from which SIARD
 ↪ was running for archiving -->
      <xs:element name="clientMachine" type="xs:string"
 ↪  minOccurs="0"/>
      <!-- name of database product and version from
 ↪ which database originates -->
```

```
      <xs:element name ="databaseProduct" type="
  ↪ xs:string" minOccurs="0"/>
      <!-- connection string used for archiving -->
      <xs:element name="connection" type="xs:string"
  ↪ minOccurs="0"/>
      <!-- database user used for archiving -->
      <xs:element name="databaseUser" type="xs:string"
  ↪ minOccurs="0"/>
      <!--  list of schemas in database  -->
      <xs:element name="schemas" type="schemasType"/>
      <!-- list of users in the archived database -->
      <xs:element name="users" type="usersType"/>
      <!-- list of roles in the archived database -->
      <xs:element name="roles" type="rolesType"
  ↪ minOccurs="0"/>
      <!-- list of privileges in the archived database
  ↪ -->
      <xs:element name="privileges" type="
  ↪ privilegesType" minOccurs="0"/>
    </xs:sequence>
    <!-- constraint: version number must be 1.0 -->
    <xs:attribute name="version" type="versionType" use
  ↪ ="required" />
  </xs:complexType>
</xs:element>

<!-- complex type schemas -->
<xs:complexType name="schemasType">
  <xs:annotation>
    <xs:documentation>
      List of schemas
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="schema" type="schemaType"
  ↪ minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type schema -->
<xs:complexType name="schemaType">
  <xs:annotation>
    <xs:documentation>
      Schema element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the schema -->
    <xs:element name="name" type="xs:string" />
    <!-- archive name of the schema folder -->
    <xs:element name="folder" type="fsName"/>
```

```
    <!-- description of the schema's meaning and
  ↪ content -->
    <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
    <!-- list of tables in the schema -->
    <xs:element name="tables" type="tablesType"/>
    <!-- list of views in the schema -->
    <xs:element name="views" type="viewsType" minOccurs
  ↪ ="0"/>
    <!-- list of routines in the archived database -->
    <xs:element name="routines" type="routinesType"
  ↪ minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type tables -->
<xs:complexType name="tablesType">
  <xs:annotation>
    <xs:documentation>
      List of tables
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="table" type="tableType" minOccurs
  ↪ ="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type table -->
<xs:complexType name="tableType">
  <xs:annotation>
    <xs:documentation>
      Table element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the table -->
    <xs:element name="name" type="xs:string"/>
    <!-- archive name of the table folder -->
    <xs:element name="folder" type="fsName"/>
    <!-- description of the tables meaning and content
  ↪ -->
    <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
    <!-- list of columns of the table -->
    <xs:element name="columns" type="columnsType"/>
    <!--  primary key -->
    <xs:element name="primaryKey" type="primaryKeyType"
  ↪  minOccurs="0"/>
    <!--  foreign keys  -->
```

```
    <xs:element name="foreignKeys" type="
  ↪ foreignKeysType" minOccurs="0"/>
    <!-- candidate keys (unique constraints) -->
    <xs:element name="candidateKeys" type="
  ↪ candidateKeysType" minOccurs="0"/>
    <!-- list of (check) constraints -->
    <xs:element name="checkConstraints" type="
  ↪ checkConstraintsType" minOccurs="0"/>
    <!-- list of triggers -->
    <xs:element name="triggers" type="triggersType"
  ↪ minOccurs="0"/>
    <!-- number of rows in the table -->
    <xs:element name="rows" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type views -->
<xs:complexType name="viewsType">
  <xs:annotation>
    <xs:documentation>
      List of views
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="view" type="viewType" minOccurs="
  ↪ 1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type view -->
<xs:complexType name="viewType">
  <xs:annotation>
    <xs:documentation>
      View element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the view -->
    <xs:element name="name" type="xs:string" />
    <!-- SQL query string defining the view -->
    <xs:element name="query" type="xs:string" minOccurs
  ↪ ="0"/>
    <!-- original query string defining the view -->
    <xs:element name="queryOriginal" type="xs:string"
  ↪ minOccurs="0"/>
    <!-- description of the view's meaning and content
  ↪ -->
    <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
    <!-- list of columns of the view -->
    <xs:element name="columns" type="columnsType"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <!-- complex type columns -->
  <xs:complexType name="columnsType">
    <xs:annotation>
      <xs:documentation>
        List of columns
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="column" type="columnType"
   ↪ minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type column -->
  <xs:complexType name="columnType">
    <xs:annotation>
      <xs:documentation>
        Column element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the column -->
      <xs:element name="name" type="xs:string" />
      <!-- archive name of the lob folder -->
      <xs:element name="folder" type="fsName" minOccurs="
   ↪ 0"/>
      <!-- SQL:1999 data type of the column -->
      <xs:element name="type" type="xs:string" />
      <!-- original data type of the column -->
      <xs:element name="typeOriginal" type="xs:string"
   ↪ minOccurs="0"/>
      <!-- default value -->
      <xs:element name="defaultValue" type="xs:string"
   ↪ minOccurs="0"/>
      <!-- nullability -->
      <xs:element name="nullable" type="xs:boolean"/>
      <!-- unique, references, check column constraints
           are stored as table constraints -->
      <!-- description of the column's meaning and
   ↪ content -->
      <xs:element name="description" type="xs:string"
   ↪ minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type primaryKey -->
  <xs:complexType name="primaryKeyType">
    <xs:annotation>
```

```
      <xs:documentation>
        primaryKey element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the primary key -->
      <xs:element name="name" type="xs:string" minOccurs=
  ↪ "0" />
       <!-- description of the primary key's meaning and
  ↪ content -->
       <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
      <!-- columns belonging to the primary key -->
      <xs:element name="column" type="xs:string"
  ↪ minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<!-- complex type foreignKeys -->
<xs:complexType name="foreignKeysType">
    <xs:annotation>
      <xs:documentation>
        List of foreign key constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="foreignKey" type="foreignKeyType"
  ↪  minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>

<!-- complex type foreignKey -->
<xs:complexType name="foreignKeyType">
    <xs:annotation>
      <xs:documentation>
        foreignKey element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the foreign key -->
      <xs:element name="name" type="xs:string" />
      <!--  referenced schema -->
      <xs:element name="referencedSchema" type="xs:string
  ↪ "/>
      <!-- referenced table -->
      <xs:element name="referencedTable" type="xs:string"
  ↪ />
      <!--  references -->
      <xs:element name="reference" type="referenceType"
  ↪ minOccurs="1" maxOccurs="unbounded"/>
      <!-- match type (FULL, PARTIAL, SIMPLE) -->
```

```
      <xs:element name="matchType" type="matchTypeType"
  ↪ minOccurs="0"/>
      <!-- ON DELETE action e.g. ON DELETE CASCADE -->
      <xs:element name="deleteAction" type="xs:string"
  ↪ minOccurs="0"/>
      <!-- ON UPDATE action e.g. ON UPDATE SET DEFAULT --
  ↪ >
      <xs:element name="updateAction" type="xs:string"
  ↪ minOccurs="0"/>
      <!-- description of the foreign key's meaning and
  ↪ content -->
      <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type reference -->
  <xs:complexType name="referenceType">
    <xs:annotation>
      <xs:documentation>
        reference element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- referencing column -->
      <xs:element name="column" type="xs:string"/>
      <!-- referenced column (table.column) -->
      <xs:element name="referenced" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type candidateKeys -->
  <xs:complexType name="candidateKeysType">
    <xs:annotation>
      <xs:documentation>
        List of candidate key (unique) constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="candidateKey" type="
  ↪ candidateKeyType" minOccurs="1" maxOccurs="
  ↪ unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type candidateKey -->
  <xs:complexType name="candidateKeyType">
    <xs:annotation>
      <xs:documentation>
        candiate key (unique) element in siardArchive
      </xs:documentation>
```

```
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the candidate key -->
      <xs:element name="name" type="xs:string"/>
       <!-- description of the candidate key's meaning
  ↪ and content -->
       <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
      <!-- columns belonging to the candidate key -->
      <xs:element name="column" type="xs:string"
  ↪ minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type check constraints -->
  <xs:complexType name="checkConstraintsType">
    <xs:annotation>
      <xs:documentation>
        List of check constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="checkConstraint" type="
  ↪ checkConstraintType" minOccurs="1" maxOccurs="
  ↪ unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type check constraint -->
  <xs:complexType name="checkConstraintType">
    <xs:annotation>
      <xs:documentation>
        Check constraint element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the constraint -->
      <xs:element name="name" type="xs:string"/>
      <!-- check condition -->
      <xs:element name="condition" type="xs:string"/>
      <!-- description of the constraint's meaning and
  ↪ content -->
      <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type triggers -->
  <xs:complexType name="triggersType">
    <xs:annotation>
      <xs:documentation>
```

```
        List of triggers
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="trigger" type="triggerType"
 ↪ minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type trigger -->
  <xs:complexType name="triggerType">
    <xs:annotation>
      <xs:documentation>
        Trigger element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the trigger -->
      <xs:element name="name" type="xs:string" />
      <!--  action time -->
      <xs:element name="actionTime" type="actionTimeType"
 ↪ />
      <!--  trigger event INSERT, DELETE, UPDATE [OF <
 ↪ trigger column list>] -->
      <xs:element name="triggerEvent" type="xs:string"/>
      <!--  alias list <old or new values alias> -->
      <xs:element name="aliasList" type="xs:string"
 ↪ minOccurs="0"/>
      <!--  triggered action -->
      <xs:element name="triggeredAction" type="xs:string"
 ↪ />
      <!-- description of the trigger‘s meaning and
 ↪ content -->
      <xs:element name="description" type="xs:string"
 ↪ minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type routines -->
  <xs:complexType name="routinesType">
    <xs:annotation>
      <xs:documentation>
        List of routines
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="routine" type="routineType"
 ↪ minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
```

```
<!-- complex type routine -->
<xs:complexType name="routineType">
  <xs:annotation>
    <xs:documentation>
      Routine
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of routine in schema -->
    <xs:element name="name" type="xs:string"/>
    <!-- description of the routines's meaning and
 ↪ content -->
    <xs:element name="description" type="xs:string"
 ↪ minOccurs="0"/>
    <!-- original source code (VBA, PL/SQL, ...)
 ↪ defining the routine -->
    <xs:element name="source" type="xs:string"
 ↪ minOccurs="0"/>
    <!--  SQL:1999 body of routine  -->
    <xs:element name="body" type="xs:string" minOccurs=
 ↪ "0"/>
    <!--  routine characteristic -->
    <xs:element name="characteristic" type="xs:string"
 ↪ minOccurs="0"/>
    <!--  SQL:1999 data type of the return value (for
 ↪ functions) -->
    <xs:element name="returnType" type="xs:string"
 ↪ minOccurs="0"/>
    <!--  list of parameters -->
    <xs:element name="parameters" type="parametersType"
 ↪  minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type parameters -->
<xs:complexType name="parametersType">
  <xs:annotation>
    <xs:documentation>
      List of parameters of a routine
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="parameter" type="parameterType"
 ↪ minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type parameter -->
<xs:complexType name="parameterType">
  <xs:annotation>
    <xs:documentation>
```

```
      Parameter of a routine
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!--  name of parameter  -->
  <xs:element name="name" type="xs:string"/>
  <!--  mode of parameter (IN, OUT, INOUT) -->
  <xs:element name="mode" type="xs:string"/>
  <!--  SQL:1999 type of argument -->
  <xs:element name="type" type="xs:string"/>
    <!-- original data type of the argument -->
    <xs:element name="typeOriginal" type="xs:string"
 ↪ minOccurs="0"/>
    <!-- description of the parameter's meaning and
 ↪ content -->
    <xs:element name="description" type="xs:string"
 ↪ minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type users -->
<xs:complexType name="usersType">
  <xs:annotation>
    <xs:documentation>
      List of users
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="user" type="userType" minOccurs="
 ↪ 1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type user -->
<xs:complexType name="userType">
  <xs:annotation>
    <xs:documentation>
      User
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- user name -->
    <xs:element name="name" type="xs:string"/>
    <!-- description of the user's meaning and content
 ↪ -->
    <xs:element name="description" type="xs:string"
 ↪ minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type roles -->
```

```
<xs:complexType name="rolesType">
  <xs:annotation>
    <xs:documentation>
      List of roles
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="role" type="roleType" minOccurs="
 ↪ 1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type role -->
<xs:complexType name="roleType">
  <xs:annotation>
    <xs:documentation>
      Role
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- role name -->
    <xs:element name="name" type="xs:string"/>
    <!-- role ADMIN (user or role) -->
    <xs:element name="admin" type="xs:string"/>
    <!-- description of the role‘s meaning and content
 ↪ -->
    <xs:element name="description" type="xs:string"
 ↪ minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type privileges -->
<xs:complexType name="privilegesType">
  <xs:annotation>
    <xs:documentation>
      List of grants
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="privilege" type="privilegeType"
 ↪ minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type privilege -->
<xs:complexType name="privilegeType">
  <xs:annotation>
    <xs:documentation>
      Grant (incl. grant of role)
    </xs:documentation>
  </xs:annotation>
```

```
    <xs:sequence>
      <!-- privilege type (incl. ROLE privilege or "ALL
  ↪ PRIVILEGES" -->
      <xs:element name="type" type="xs:string"/>
      <!-- privilege object (may be omitted for ROLE
  ↪ privilege) -->
      <xs:element name="object" type="xs:string"
  ↪ minOccurs="0"/>
      <!-- GRANTED BY -->
      <xs:element name="grantor" type="xs:string"/>
      <!-- user list of users or roles or single value "
  ↪ PUBLIC" -->
      <xs:element name="grantee" type="xs:string"/>
      <!-- optional option "GRANT" or "ADMIN" -->
      <xs:element name="option" type="privOptionType"
  ↪ minOccurs="0"/>
      <!-- description of the grant's meaning and content
  ↪  -->
      <xs:element name="description" type="xs:string"
  ↪ minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- simple type for digest string -->
  <xs:simpleType name="digestType">
    <xs:annotation>
      <xs:documentation>
        digestType must be empty or prefixed by MD5 oder
  ↪ SHA1
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:pattern value="((MD5|SHA-1).*)?"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for version number -->
  <xs:simpleType name="versionType">
    <xs:annotation>
      <xs:documentation>
        versionType must be constrained to "1.0"
        for conformity with this XLM schema
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="1.0"/>
    </xs:restriction>
  </xs:simpleType>
```

```
<!-- simple type for privilege option -->
<xs:simpleType name="privOptionType">
  <xs:annotation>
    <xs:documentation>
      privOptionType must be "ADMIN" or "GRANT"
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:enumeration value="ADMIN"/>
    <xs:enumeration value="GRANT"/>
  </xs:restriction>
</xs:simpleType>

<!-- simple type for mandatory string
     which must contain at least 1 character -->
<xs:simpleType name="mandatoryString">
  <xs:annotation>
    <xs:documentation>
      mandatoryType must contain at least 1 character
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="preserve"/>
    <xs:minLength value="1" />
  </xs:restriction>
</xs:simpleType>

<!-- simple type of a filesystem (file or folder) name
 ↪ -->
<xs:simpleType name="fsName">
  <xs:annotation>
    <xs:documentation>
      fsNames may only consist of ASCII characters and
 ↪ digits
      and must start with a non-digit
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="([a-z]|[A-Z])([a-z]|[A-Z]|[0-9])
 ↪ .*" />
    <xs:minLength value="1" />
  </xs:restriction>
</xs:simpleType>

<!-- simple type for action time of a trigger -->
<xs:simpleType name="actionTimeType">
  <xs:annotation>
    <xs:documentation>
      actionTime is BEFORE or AFTER
    </xs:documentation>
```

```
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="BEFORE" />
      <xs:enumeration value="AFTER" />
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for match type of a foreign key -->
  <xs:simpleType name="matchTypeType">
    <xs:annotation>
      <xs:documentation>
        matchType is FULL, PARTIAL or SIMPLE
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="FULL" />
      <xs:enumeration value="PARTIAL" />
      <xs:enumeration value="SIMPLE" />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Listagem D.1: *Schema* de *metadata.xml*: *metadata.xsd*

Apêndice E

SIARD: exemplo *table.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<table
   xsi:schemaLocation="http://www.admin.ch/xmlns/siard
    ↪ /1.0/schema0/table6.xsd table6.xsd"
   xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/
    ↪ table6.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <row><c1>2</c1><c2>Musik</c2><c4>2003-04-11T12:29:10
    ↪ .000000000</c4></row>
   <row><c1>4</c1><c2>Pop</c2><c4>2003-04-11T12:29:10
    ↪ .000000000</c4></row>
   <row><c1>8</c1><c2>Tanz</c2><c4>2003-04-11T12:29:10
    ↪ .000000000</c4></row>
   <row><c1>16</c1><c2>Theater</c2><c4>2003-04-11T12:29:10
    ↪ .000000000</c4></row>
   <row><c1>32</c1><c2>Literatur</c2><c4>2003-04-11
    ↪ T12:29:10.000000000</c4></row>
   <row><c1>64</c1><c2>Kleinkunst</c2><c4>2003-04-11
    ↪ T12:29:10.000000000</c4></row>
   <row><c1>128</c1><c2>Neue Medien</c2><c4>2003-04-11
    ↪ T12:29:10.000000000</c4></row>
   <row><c1>256</c1><c2>Soziale Themen</c2><c4>2003-04-11
    ↪ T12:29:10.000000000</c4></row>
</table>
```

Listagem E.1: SIARD: exemplo *table.xml*