Universidade do Minho
Escola de Engenharia
Departamento de Informática

Bruno Alexandre Alves Ferreira

Database Preservation Toolkit

A relational database conversion and normalization tool

November 2016

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Bruno Alexandre Alves Ferreira

Database Preservation Toolkit

A relational database conversion and normalization tool

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
Dr. José Carlos Leite Ramalho
Eng. Luís Francisco da Cunha Cardoso de Faria

November 2016

ACKNOWLEDGEMENTS

I would like to show my gratitude towards everyone that contributed to this dissertation:

To my supervisors, José Carlos Ramalho and Luís Faria, thank you most sincerely for your support, wisdom and guidance throughout the development of this work.

To KEEP SOLUTIONS, the company that co-funded the presented work and provided the excellent workspace in which it was developed. To Miguel Ferreira, Vitor Fernandes, Hélder Silva and Sébastien Leroux for their contributions to this work and their support. To all my company colleagues, for their support and their fellowship.

To the E-ARK project, who also co-funded this work. To all E-ARK partners. To the pilots from the Danish National Archives, the National Archives of Hungary, the National Archives of Estonia, National Archives of the Republic of Slovenia and the United States National Archives, who tested and provided feedback on the database preservation tools, their contributions were essential to this work. To Janet Delve, Co-ordinator of the E-ARK project, to Anders Bo Nielsen, Phillip Tømmerholt and Ann-Kristin Egeland from the Danish National Archives, to Zoltán Szatucsek, Zoltán Lux and József Mezei from the National Archives of Hungary, to Kuldar Aas and Lauri Rätsep from the National Archives of Estonia, and to Boris Domajnko from the National Archives of the Republic of Slovenia, who were kind enough to contribute their testimony about these tools and their development.

To everyone who presented these tools in conferences worldwide, and to everyone who contributed to this work via GitHub or email, by submitting code, issues and improvement suggestions, you have my gratitude.

Finally, I would like to thank my girlfriend, my parents, my brother and all of my family, for supporting and encouraging me on following my dreams.

You have my deepest gratitude,
Bruno Ferreira

ABSTRACT

Databases are one of the main technologies supporting organizations' information assets, and very often these databases contain information that is irreplaceable or prohibitively expensive to reacquire. The digital preservation field attempts to maintain this kind of information accessible and authentic for multiple decades, but the complexity commonly found in databases and the incompatibilities between database systems make it difficult to preserve this kind of digital object.

The Database Preservation Toolkit is a software that automates the migration of relational databases to the second version of the Software Independent Archiving of Relational Databases format. Furthermore, this flexible tool that supports the current most popular Relational Database Management Systems can also convert a preserved database back to a Database Management System, allowing for some special usage scenarios in an archival context. The conversion of databases between different formats, whilst retaining the databases' significant properties, poses a number of interesting issues, which are described in this document, along with their current solutions.

To complement the conversion software, the Database Visualization Toolkit is introduced, a software tool that provides access to preserved databases, enabling a consumer to quickly search and explore a database without knowing any query language. The viewer is capable of handling big databases as well, promptly presenting results of searching and filtering operations on millions of records.

This work covers the challenges of relational database preservation, and the development of a format and tools that play an important role in successfully preserving this kind of information.

RESUMO

As bases de dados são uma das principais tecnologias para armazenamento e gestão de informação digital de uma organização e, caso se perdesse, esta informação poderia ser de muito difícil ou dispendiosa recuperação. Para evitar este tipo de situações e despesas, a área da Preservação Digital tenta encontrar formas de manter a informação acessível e autêntica durante várias décadas, no entanto, devido à complexidade presente nas bases de dados e às incompatibilidades entre diferentes sistemas de base de dados, preservar bases de dados não é uma tarefa trivial.

O Database Preservation Toolkit é uma aplicação que automatiza a conversão de bases de dados relacionais para um formato especialmente desenhado para a sua preservação, o Software Independent Archiving of Relational Databases. Esta ferramenta é capaz de exportar bases de dados dos sistemas de gestão de base de dados mais populares, e também recuperar a base de dados para um sistema de gestão de base de dados, potencialmente diferente do seu sistema original. Esta flexibilidade na escolha dos formatos ou sistemas de entrada e saída faz com que a ferramenta possa ser usada para solucionar vários problemas no contexto da preservação de bases de dados.

Para complementar a ferramenta de conversão foi criada uma plataforma de visualização de bases de dados que estejam num formato de preservação, o Database Visualization Toolkit. Esta plataforma permite analisar os meta-dados da base de dados e pesquisar o seu conteúdo, sem requerer conhecimento especializado na área das bases de dados. A ferramenta foi desenhada para providenciar acesso a bases de dados de grandes dimensões, para ter a capacidade de apresentar rapidamente os resultados de pesquisas em milhões de registos.

O presente documento foca-se na preservação de bases de dados relacionais, abordando as principais dificuldades dessa atividade, assim como o desenvolvimento de um formato específico para preservação desses objetos, e descreve o desenvolvimento e funcionamento de ferramentas que têm um papel crucial na preservação deste tipo de objetos digitais.

CONTENTS

1 INTRODUCTION 1
- 1.1 Motivation 1
- 1.2 Objectives 1
- 1.3 Document Structure 2

2 STATE OF THE ART 3
- 2.1 Digital Preservation 3
  - 2.1.1 The Digital Object 3
  - 2.1.2 Long-term Digital Preservation 4
  - 2.1.3 Digital Preservation Strategies 5
  - 2.1.4 The OAIS Reference Model 7
  - 2.1.5 Significant Properties and Authenticity 9
- 2.2 Relational Databases 10
  - 2.2.1 Relational Data Model 10
  - 2.2.2 Structured Query Language Standard 12
  - 2.2.3 Accessing Information 16
- 2.3 Digital Preservation of Relational Databases 17
  - 2.3.1 The E-ARK Project 17
  - 2.3.2 Significant Properties 18
  - 2.3.3 Preservation Formats 20
  - 2.3.4 Migration Tools 21
  - 2.3.5 Viewers for Preserved Databases 23

3 METHODOLOGY 25
- 3.1 Usage scenarios 25
- 3.2 Requirements 26
- 3.3 Approach 28
- 3.4 Validation 30

4 SOLUTION 31
- 4.1 SIARD 2 31
  - 4.1.1 Database Metadata 32
  - 4.1.2 Database Contents 36
  - 4.1.3 SIARD in an OAIS Archive 36
  - 4.1.4 Handling LOBs 37
- 4.2 Database Preservation Toolkit 39
  - 4.2.1 Purpose and Requirements 39

4.2.2 Design 40
4.2.3 Development 42
4.2.4 Deployment 47
4.2.5 Testing 47
4.2.6 Validation 49
4.3 Database Visualization Toolkit 53
4.3.1 Purpose and Requirements 53
4.3.2 Design and Development 54
4.3.3 Deployment 56
4.3.4 Testing 56
4.3.5 Validation 56
5 VALIDATION 70
5.1 Methodology 70
5.2 Results 71
6 CONCLUSION AND FUTURE WORK 80

A SIARD 2 FORMAT DETAILS 88
A.1 Database metadata elements 88
A.2 Table metadata elements 89

1 INTRODUCTION

This chapter introduces the subject of relational database preservation, the main objectives that this work tries to tackle within this subject, and how this document is organised.

1.1 MOTIVATION

Databases are one of the main technologies supporting organizations' information assets. They are designed to store, organize and retrieve digital information, and are such a fundamental part of information systems that most would not be able to function without them (Connolly and Begg, 2004). Very often, the information contained in databases is irreplaceable or prohibitively expensive to reacquire, therefore steps must be taken to ensure that the information within databases is preserved.

It is common for governments to have databases with important and invaluable information, such as taxes, pensionary records and judicial information; this information is crucial to maintain fairness on citizens' rights and obligations. Other organizations, such as hospitals and laboratories, also have valuable medical information for researchers attempting to understand and solve health-related problems. This information is valuable must be kept authentic and accessible throughout several decades.

The digital preservation field tries to define the policies and processes necessary to keep information accessible and integrous. However, for databases this task is specially hard because they are complex objects and present unique challenges to traditional digital preservation methods. The heterogeneity of information in databases is difficult to transport through time, and the variety and popularity of Database Management Systems that make use of proprietary storage formats or use non-standard query languages further hinder the preservation process.

1.2 OBJECTIVES

This work aims to contribute to the digital preservation community by designing and developing the technology necessary to support the archival of relational databases, as well as the technology capable of providing access to the database data and metadata. Both technologies are meant to be used by information producers, archivists and researchers, supporting database preservation.

Chapter 3 details the objectives for each technology and their intended usage in a digital preservation context.

1.3 DOCUMENT STRUCTURE

This document is organised in 6 chapters:

The first chapter presents the introduction to the dissertation, briefly describing the motivations for the current work and its goals.

The state of the art is covered in the second chapter, describing fundamental preservation and database concepts, as well as analysing existing database preservation solutions.

The third chapter details the usage scenarios and requirements for the database archival tool and the database access tool.

The SIARD 2 format was created to maintain the information of databases and support the preservation process. Its development and the extent of its capabilities is described in chapter 4.1. The software artefacts Database Preservation Toolkit and Database Visualization Toolkit, used to preserve and provide access to preserved databases, are also presented in chapter 4.1. The validation of both tools is presented in chapter 5, in order to assess if and how they meet their objectives.

Finally, the sixth chapter presents conclusions that can be extracted from this work, the main contributions it provides and a description of future work.

2

STATE OF THE ART

This chapter covers the state of the art, context and essential concepts related to the long-term preservation of databases.

2.1 DIGITAL PRESERVATION

Here the digital preservation field is introduced, starting with the digital object concept, followed by long-term digital preservation and existing strategies to achieve it, and finally an analysis of current consensus on the characteristics that should be preserved.

2.1.1 *The Digital Object*

Digital preservation is the group of activities and processes guaranteeing that information will be continuously accessible, integrous and authentic throughout several decades. Information, in the digital preservation research field, is formally referred to as a *digital object*, which is *an information object, of any type of information or any format, that is expressed in digital form* (Thibodeau, 2002). Examples of digital objects include movies, music records, a patient's medical records, scientific research data, models of buildings or vehicles, software's source code and public administration databases.

Digital information use is currently widespread, mostly because it is easier to create, copy, edit and publish. But it is also fragile, due to requiring a proper technological environment to be correctly consumed. In opposition, an analogue object can be readily consumed, e.g. anyone can pick up, read and understand a newspaper article[1]; whilst an online version of the same newspaper article requires a complete set of technologies to be correctly consumed, starting with a screen where images can be formed and perceived by the human eye, those images would need to be provided by a software that can render the article from its byte representation (like a web browser), which exists on top of some kind of operating system which interacts with hardware parts like memory and processor; however, for this

1 Based on the assumption that one can physically pick up the newspaper, read the article text, understand the language and has some level of knowledge on the subject. But these are all problems that are common to both analogue and digital objects.

example of an online article, the bytes would have to be obtained from the internet, which is another set of complex technologies working together, and ultimately the article would be on some storage system in a remote data centre. (Faria, 2015)

To avoid dealing with this kind of complexity when discussing the digital object, it is often divided into three levels (Thibodeau, 2002):

PHYSICAL: The representation of the object as an inscription of signs on some physical medium, e.g. a CD or flash-drive;

LOGICAL: The binary coding of the information, the file format, which is interpreted by a software reader, e.g. a file in Portable Document Format (PDF) which can be interpreted by Acrobat Reader software;

CONCEPTUAL: The tangible unit of information that can be recognised and understood by a person, e.g. a journal, a book or a photo;

If any of the levels is damaged, the access to the next is endangered and may ultimately lead to the corruption of the conceptual object. Taking, as an example, a digital photo of a landscape, saved in PNG format[2] on a CD; the inability to read some bytes from the CD due to a scratch (Pohlmann, 1989) may make it impossible for a optical drive to read some bytes and the rendering software may be unable to process and correctly render the picture.

2.1.2 *Long-term Digital Preservation*

Concerning long-term digital preservation, in which the conceptual object needs to remain accessible and authentic for various decades (sometimes for undetermined time), and besides the previously mentioned threats, there is an additional threat from hardware, software and data formats deprecation, which may hinder or make it impossible to render the conceptual object. Reusing the previous example, a CD reader drive needs to be used to retrieve the PNG photo bytes from the CD. There is a risk that given enough time, technologies become obsolete and it would become increasingly difficult to find CD reader drives, threatening the logical and conceptual levels of the digital object. Regarding the logical object, the PNG format use is currently widespread and the format is very well documented, so even if the format were to disappear, the format specification could be used to create a software to interpret such format. This is not the case for some proprietary or unpopular formats which may be abandoned (e.g. for financial reasons) and would need to be reverse

2 Portable Network Graphics, an extensible file format for the lossless, portable, well-compressed storage of raster images. (Boutell, 1997)

engineered[3] to create new rendering software. (Ferreira, 2006; Wilson, 2007; Ramalho et al., 2007a)

2.1.3 *Digital Preservation Strategies*

Traditionally, *"preserving"* means *"to maintain (something) in its original state"*, and in digital preservation the "something" is the conceptual object, that is meant to be kept in its original state and without losses, independently of its physical format. One could try a different approach, preserving digital information and its physical container without modifications, but accessing this information in the future would become increasingly difficult, if not impossible. This problem led to the creation of different strategies to preserve digital information.

The scope of these strategies may vary, as different strategies pursue the preservation of the digital object at different levels. The strategies discussed in this section operate mainly on the logical level. Different approaches also exist because different researchers believe that simply preserving the conceptual object is too risky or error prone, and try to overcome this by preserving more levels of the digital object, such as the original bytes (logical level) or hardware (physical level). Only some of these strategies are relevant to this document, and are introduced below. (Ferreira, 2006; Lee et al., 2002; Wilson, 2007)

Emulation

Emulation-based strategies are supported by the use of an emulator, a software that simulates the digital object's original software and hardware environment. This strategy is important and may be the preferred one to preserve digital objects that include interactive aspects. As an example, it is common for digital games from discontinued consoles to be playable in a personal computer through the use of an emulator. But using this strategy might not be ideal, mainly because the knowledge needed to operate old applications and operating systems might also fade with time. A secondary reason to not rely only on this strategy is that all the problems (e.g. bugs and security issues) present on the original system may also be preserved and result in loss of information. An example of this could be a software bug or computer virus[4] that deleted essential system files that was also preserved alongside the original software; the harmful behaviour could then be triggered when accessing the software a couple decades later and permanently erase those essential system files.

3 Reverse engineering is the reproduction of another manufacturer's product following detailed examination of its construction or composition. This process is usually slow, expensive and may produce imperfect or incomplete results. (Chikofsky and Cross, 1990)

4 Computer virus is a malicious software that performs some kind of harmful activity on the infected computer.

This strategy may also bring additional legal problems to the preservation process, since the majority of software is protected by licenses and intellectual rights, limiting or prohibiting its emulation.

Emulation may become problematic, as the emulator is also affected by software deprecation. The necessary steps need to be taken to ensure the preservation of the digital object, such as creating a new emulator for the digital object or creating an emulator for the emulator (although this might start an unmaintainable cycle). The strategy also assumes that future users will be able to interact with discontinued software and operating systems, which might prove difficult. (Ferreira, 2006; Woodyard, 2000)

Migration

Migration-based strategies focus on the preservation of the conceptual level, migrating it between formats to keep it compatible with current technology. The migration process must ensure that no significant properties are lost, otherwise the preservation will fail. This strategy also requires constant monitoring of formats, to migrate those nearing deprecation. Despite these problems, format migration is the most widely accepted digital preservation strategy and the one with higher success rates. (Ferreira, 2006; Lee et al., 2002)

Policy: Migration to an analogue medium

Following a policy based on the migration of the digital object to a long-duration analogue medium and focusing on preserving that analogue medium can be preferable in some cases, e.g. preserving a 2D automotive model, created with modern Computer-aided design software, on paper. Using this policy may postpone having to handle the preservation of digital formats, however it is not possible to use this strategy for interactive or dynamic digital objects, since they do not possess an equivalent analogue representation. (The Commission on Preservation and Access and The Research Libraries Group, 1996)

Policy: Migration to a digital format

Following a policy based on the migration of specific digital object types (e.g. images) to a single digital format helps reduce format diversity in a repository and may help reduce preservation costs, since future interventions will have to handle fewer formats. While planning this strategy, special attention must be taken when choosing the normalization format, since a bad choice in this stage could lead to increased preservation costs and even information loss.

The chosen format should be able to maintain all the digital object's significant properties. Additionally, these formats should be international and open formats, based on open and widely supported technologies. The format should have these characteristics to have higher chance of being interpreted in the future. (Ferreira, 2006; Thibodeau, 2002; Hodge and Frangakis, 2004)

Encapsulation

This strategy attempts to preserve the unaltered digital object together with all the information needed for the future development of viewers, emulators and converters for the digital object. This strategy is very cost-efficient if the digital object will not be accessed for some years, avoiding the kind of costly maintenance tasks that are required by other strategies. The maintenance tasks for this strategy revolve around adding more documentation information, aiming to better describe the digital object's original technological environment.

Combining multiple strategies

It is noteworthy that multiple digital preservation strategies may be combined to increase the changes of successfully preserving the digital object. As a practical example, some traits from *encapsulation* strategies may be used to improve the emulation strategy, documenting the original environment and information format, and making it easier to create a new emulator some decades later.

2.1.4 *The OAIS Reference Model*

The Open Archival Information System (OAIS) reference model[5] identifies the different components in an archive focused on long-term digital preservation[6] and their responsibilities. An OAIS-compliant archive is an organization of people and systems that has accepted the responsibility to preserve information and make it available for a designated community, according to the OAIS reference model. (Lavoie, 2014; Consultative Committee for Space Data Systems, 2002)

Figure 1 illustrates, in a general way, the different components in an OAIS-compliant archive and information package exchanges between those components. The **producer** produces information and packages it in a Submission Information Package (SIP), which is a special package format designed to support delivering information (along with metadata) to the archive. The SIP is delivered to the archive and some other procedures initiate in order to archive the information. This step corresponds to the **ingestion phase**, although these events may also be separated[7] in a **pre-ingest phase**, taking place before the SIP is

5 The OAIS reference model is defined in ISO Standard 14721:2012 (E)

6 The OAIS reference model focuses mainly on preserving digital information, but the framework it provides can be expanded to cover the long-term preservation of information in non-digital form.

7 This division is not part of the OAIS reference model, as it considers only an ingest phase.

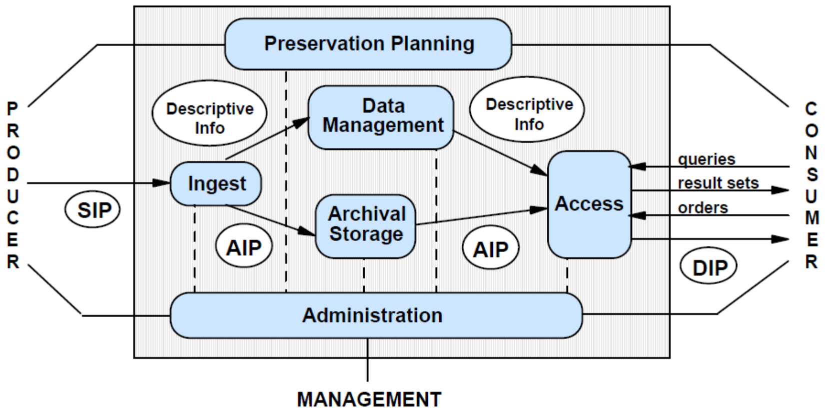


Figure 1.: OAIS functional model diagram (ISO Standard 14721:2012, E)

delivered to the archive, and an **ingest phase**, starting when the SIP is delivered to the archive.

After the SIP is delivered to the archive, the archive converts the SIP to an Archival Information Package (AIP) and transfers the newly created AIP to archival storage.

After the AIP has been archived, possibly multiple decades after the archiving action, a **consumer** (formally, a member of the designated community) may want to **access** the preserved information. The consumer, aided by the archivist, should be able to interact with the system and request some information. Since the AIP is not ready to be consumed, the archive must convert it into a Distribution Information Package (DIP), which includes the information ready to be experienced by the consumer. Depending on the information and preservation strategy, the DIP may take a few days to a few weeks to be concluded. For instance, in a best-case scenario, the DIP for a preserved digital photo may be available in a couple of days if the image format has been migrated to newer formats during the time it has been archived; in a worst-case scenario, the image format may have been discontinued[8] and a new software must be developed to render the image (resorting to documentation, reverse engineering, etc.), which might take months.

When the DIP is ready, it is made available to the consumer so that the digital object can be accessed, thus completing the **access** phase and successfully carrying the information through time. (Lavoie, 2014; Ferreira, 2006; Consultative Committee for Space Data Systems, 2002)

The DIP is used to distribute the information, e.g. by physically accessing a specific device, like a computer, containing the information or even the original system that was used decades ago. A different access methodology called dissemination may also be used, although it is not included in the OAIS reference model. In this methodology, the infor-

8 This can be avoided by monitoring the formats and technologies present in the archive to detect threats before they cause permanent damage. (Faria, 2015)

mation is not delivered in a package but rather given access to via some technology (e.g. displayed in a website or available via streaming). (Lagoze et al., 2005).

2.1.5 *Significant Properties and Authenticity*

Digital objects possess some properties that are essential to access the digital object in the future, these are the digital object's significant properties. These properties depend on the object type, organisation policies and the object's future purpose.

As a practical example, when preserving an online newspaper article the text is commonly a significant property; but other properties can be considered significant, such as the formatting (bold, italic, alignment, underline, emphasis) or the metadata (title, creator, creation and publication date). If the article contains links to other web pages then the links should also be preserved, along with any content on the linked pages.

Not all properties are significant, and while some may be significant for a purpose or organisation, others may be optional for other appliances. There is widespread agreement on the important role significant properties play in digital preservation processes, but no consensus on what is significant. In most cases, the ideal significant properties are the ones required by the designated community, but as there is no current way to determine those, organisations try to make educated guesses based on past experiences and opt for some property subset that is allowed by current technical and financial limitations. (Grace et al., 2009; Hockx-Yu and Knight, 2008; Dappert and Farquhar, 2009; Freitas and Ramalho, 2010)

For the purposes of this document, significant properties are defined as:

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

This definition connects the digital object with the concepts of authenticity (that an object has the capacity to be accepted as evidence of what it purports to record) and integrity (that there was no corruption of the digital object at any of its three levels). For the preservation to be successful, the digital object must maintain the information authenticity and integrity, even with the environment changes that are bound to happen. And the digital object itself can change at the byte stream level, as long as the conceptual object is still able to be experienced. (Wilson, 2007)

The following section covers relational database concepts, then significant properties for relational databases are discussed in section 2.3.

2.2 RELATIONAL DATABASES

Databases are complex digital objects that contain heterogeneous information, often accompanied by structural definitions and even documentation. Their complexity makes it difficult to preserve this kind of object whilst maintaining all its significant properties. Formally, a database is *a shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization*. But they are usually enclosed in a DBMS, which is *a software that enables users to define, create, maintain and control access to the database*; and provides specific languages to define and manipulate the database, which are usually some variation of SQL. (Connolly and Begg, 2004)

This section describes relational database features and their importance to understanding the database. Henceforth, the term DBMS is used to refer to RDBMS and *database* refers to *relational database* (unless explicitly stated otherwise), since the focus of this document is on relational databases. Databases are usually manipulated using a specific software, e.g. a business management application that allows adding and updating business information. This section will focus on database and DBMS, and will not cover application-specific logic.

2.2.1 *Relational Data Model*

The RDBMS is the dominant database software in use today[9] and is based on the relational data model proposed by Codd (1970). In this model, all data is structured in **tables** (or relations), **table columns** (or attributes) and **table rows** (or tuples) containing one value per column (corresponding to a **cell**). Each column has a specific **type** (or domain) which defines the set of allowable values for cells in the column. Figure 2 shows a basic example of this structure and its data, however it does not include the column types specification.

The relational model also introduces the concept of **relational keys**. Keys consist of a column, or a group of columns, used to uniquely identify each row in a table, and the model defines four different kinds of keys (Connolly and Begg, 2004):

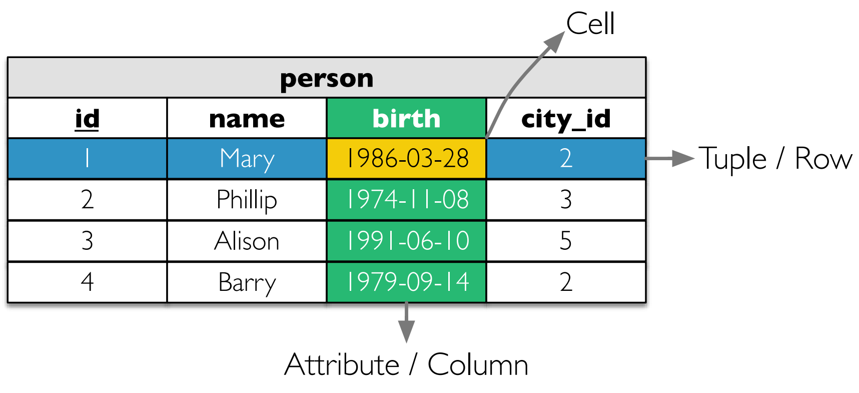
SUPERKEY

A group of columns used to uniquely identify each row. This key contains additional columns that are not needed to uniquely identify a row and therefore primary keys are used instead.

CANDIDATE KEY

A column or group of columns used to uniquely identify each row. This key differs

9 According to the ranking at `http://db-engines.com/en/ranking` (accessed on July 2016), where 7 of the top 10 DBMS use the relational model. The ranking was based on a scoring system in which 90% of the total score points in the top 10 belonged to RDBMSs.



person			
id	**name**	**birth**	**city_id**
1	Mary	1986-03-28	2
2	Phillip	1974-11-08	3
3	Alison	1991-06-10	5
4	Barry	1979-09-14	2

Cell
Tuple / Row
Attribute / Column

Figure 2.: An example of a relational database table, named `person`, and some data

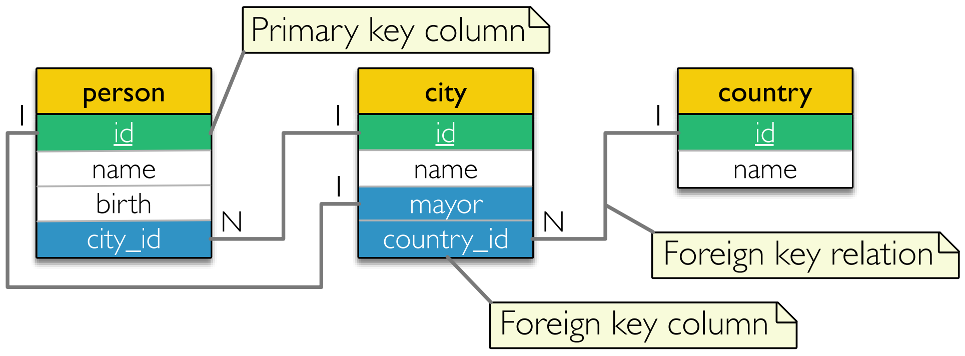


Figure 3.: Entity-relation Physical Database Model of a simple database schema

from the superkey because it is irreducible, meaning that all columns in the key are needed to uniquely identify a row. Several of these keys can exist for a table.

PRIMARY KEY
The candidate key that was selected to uniquely identify each row in a table.

FOREIGN KEY
A column or group of columns that matches a candidate key of some (possibly the same) table.

The above foreign key description hints at a very important concept of the relational model: the **foreign key relation**. Using a foreign key, two rows can be connected by matching a foreign key of a row to a candidate key in another row. To better explain this concept, a simplified relational database schema is depicted in figure 3.

In the database schema (figure 3) there are three tables, person, city and country, each having its own primary key (coloured green) composed of a single column. Three foreign keys are also represented (coloured blue), including the lines linking them to the target candidate key column. The "1" and "N" in those lines define the multiplicity, and thus the type, of the foreign key relation (i.e. *one to many* or *one to one*[10]). By starting on the `country` column in table `city`, one could read the foreign key relation as *"each city has one country, and one country can have many cities"*; as another example, the foreign key relation relating the `mayor` column in table `city` to the `id` column in table `person` could be read as *"each city has a mayor (who is a person), and one person can be mayor of one city"*.

In relational database model theory there are some **integrity rules** that should be enforced by the DBMS to ensure that the data is coherent[11]. These rules exist because cells in a database can have a value of `NULL`, representing unknown, not applicable or not yet defined data. As an example of the importance of the `NULL` value, in figure 2, if the birth date of a person is unknown, the value `NULL` should be used, because any other value (e.g. the date `0000-00-00`) will have a different meaning and possibly odd consequences (e.g. if the system was used to calculate the age, it would inform that a person with a birth date of `0000-00-00` is 2016 years old). Regarding the integrity rules themselves, there is an integrity rule specifying that the primary key of a table can not be `NULL`; and a referential integrity rule specifies that the value of a foreign key column must have a matching value on the related table or be `NULL`. These rules enforce some basic preconditions to simplify the use of foreign key relations.

The final database element defined in the relational model is the **view**, which is a virtual/derived table that is dynamically created from the underlying base tables when required. Views may be used to hide confidential information or join, transform and display information from multiple tables in a way that is easier to understand.

2.2.2 *Structured Query Language Standard*

The SQL ISO Standard 9075:2008 (E) defines the database elements and the operations on those elements. Some of these elements were introduced with the relation model, but others are closer to the DBMS and were not covered by the initial model.

Not all DBMS implementations (e.g. MySQL, Oracle, PostgreSQL) follow all parts of the standard. Some DBMSs deviate from the standard by adding functionality or implementing it in a different way. In some cases, the standard is purposely vague, to allow developers to choose how to implement the described functionality.

The **database** element is usually the base or root element of a database object, and contains information about users and their privileges, as well as schemas.

10 Conceptually, there is a third type of foreign key relation, the *many to many* type, but this kind must be transformed into multiple *one to many* foreign key relations as DBMSs do not support it.

11 The integrity, in relational database terms, consists in keeping data coherency in the database, and should not be confused with the integrity of the digital object.

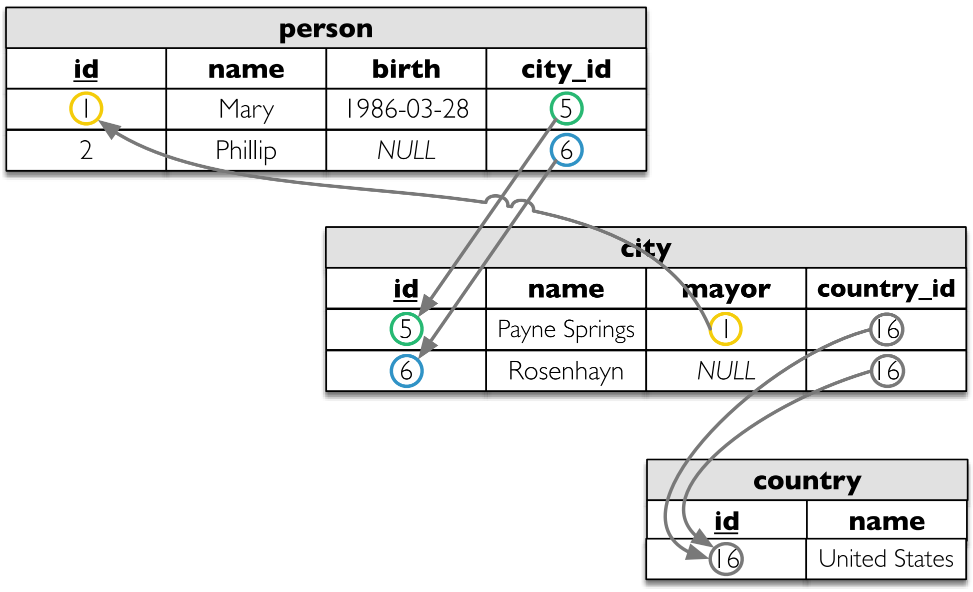


Figure 4.: Example data, detailing the foreign key relations

Although the **user** concept is present in the relational model, it is introduced as an entity that has access to at least some part of the database. The SQL ISO Standard 9075:2008 (E) details these entities further, by introducing **roles** and **privileges**. In practice, databases have a set of **users** who are allowed to manage certain parts of the database depending on their permissions. A user can be given the **privilege** to use or perform some action on a database element. Privileges can also be bundled in a **role** (e.g. administrator or researcher) to be easier to grant and manage.

The **schema** is used to separate tables and other database elements into logical groups that are easier to manage. A database can contain multiple schemas, although in some DBMSs (e.g. MySQL and SQLite) a schema is synonymous with a database and to logically separate database elements one would need to create multiple databases instead of multiple schemas. In this case, all database elements that are usually enclosed in a schema belong to the database instead. This difference is illustrated in figures 5 and 6.

Routines are sequences of instructions, written in SQL or a programming language like C, which can be invoked through a given name. Routines come in two types, **functions** and **procedures**, depending on whether they produce a value as result or not, respectively.

Changes to table contents may be limited by **constraints**. Constraints specify conditions that must hold true to ensure data integrity. Keys are a special case of constraints, since primary keys ensure that no other row in the same table has the same primary key value, and foreign keys ensure that a value in the source of a foreign key relation has a corresponding value on the foreign key target.

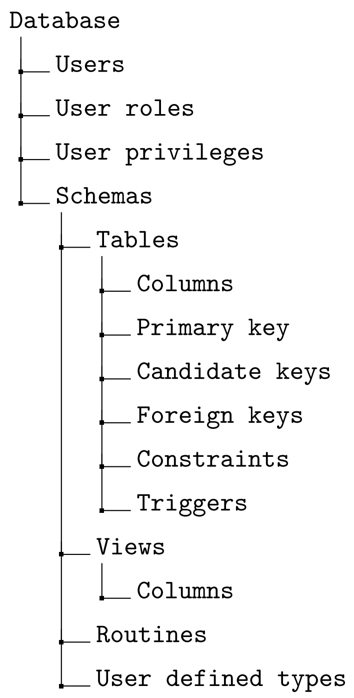


Figure 5.: Hierarchy of database elements, including schemas.

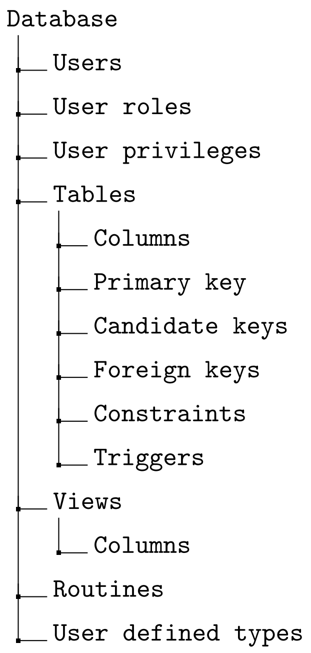


Figure 6.: Hierarchy of database elements, excluding schemas.

Triggers provide the functionality to execute arbitrary code (often through the use of procedures) when data is added, updated or removed from a table. As an example, triggers may be used to ensure that the foreign key source rows are deleted when the foreign key target row is deleted (e.g. deleting a movie should delete all related ratings).

The column definition must include its **data type**, which defines the range of values that can be present in cells of that column, e.g. a column of integer type must only contain integer (or `NULL`) cell values, and a column of a textual type will always contain a sequence of characters (or `NULL`) as cell values. Besides defining the type of data (text, integer numbers, floating point numbers, boolean, binary, etc.), data types often provide some form of parameterization through which the value range can be limited or amplified, e.g. a textual data type may include a parameter to define the upper limit on the number of characters for each cell value. There are many different data types, some are specific to a DBMS product, others exist across multiple or most DBMSs, but even for common data types, slight implementation differences may make their values incompatible across DBMS products.

Additionally, some DBMSs allow the database administrator to create **custom data types**, also known as User Defined Types (UDT). These data types can be used to represent complex data, e.g. to store an address, composed of a street name, number, postal code, and other information in a table, the database administrator could create a column for each address subcomponent, or create a custom data type that encloses this information and create a column of that type. Depending on the DBMS, new data types may be defined using a language specific to the DBMS or imported from an external library.

Most DBMSs include some data type that is able to store **binary sequences of arbitrary size**. These binary sequences tend to be larger (in bytes) than most cells, and thus they are said to be (or contain) BLOBs. A common use of these cells is to store files in the database, e.g. in a blog, they can be used to store the images displayed in a blog post. These BLOB cells are intrinsically bound to the software using the database and it may be difficult for other applications (like the DBMS) to understand the information represented by the binary data. When the arbitrary sized binary sequence is known to be text, the cells may be called CLOBs and the term LOB is then used to refer to both BLOBs and CLOBs.

Some DBMSs support inserting multiple values of the same type in a cell. The column type is this case is an **array type**, but its use is discouraged as it goes against database normalization practices[12]. Nonetheless arrays are supported by the SQL ISO Standard 9075:2008 (E).

When planning a new database, its final plan or report can include some form of data dictionary, describing the semantics of each table, column, view, and other database elements; as well as justifications, observations and comments on those elements. These descriptions are invaluable information to understand the meaning and the data contained in the database. Some DBMSs provide special fields that allow database administrators to include this descriptive information near the database elements. The naming for those fields varies, being called *comment*, *remark* and *description*; but their purpose is the same. Even though this information has the potential to be very useful, most DBMSs do not support adding descriptive information[13] to most database elements.

Structured Query Language

One of the main focus of the SQL ISO Standard 9075:2008 (E) is to define the Structured Query Language (SQL), which used to instruct the DBMS to perform a variety of tasks on databases and interact with the database elements. These instructions are commonly referred to as SQL statements. Similarly to what happens with data types, DBMSs tend to implement some variation of the Standard SQL (called SQL flavours) to support functionality specific to the DBMS. The flavours may be partially or totally incompatible with each other, as they may also use slightly different language syntax.

Some special SQL statements are used to retrieve information from a database, those are called SQL queries. When high performance is important, a series of SQL statements (also

12 Specifically, using arrays is against the first normal form. The normal forms are several guidelines to reduce data redundancy and improve data integrity. (Codd, 1970, 1971)

13 PostgreSQL is an exception to this, since according to its documentation (available at `https://www.postgresql.org/docs/9.1/static/sql-comment.html`, accessed on August 2016) it does support adding descriptive metadata to most database elements.

know as a "batch") can be sent to the server at once, reducing the overhead needed to execute the statements.

2.2.3 *Accessing Information*

The DBMS works based on a client-server architecture. The server application receives the requests and interacts accordingly with the database, sending the responses back to the client application. The client application should be able to request database information from the server (through SQL queries) and display this information in some way.

Regarding client-side software, there is often a client software provided by the DBMS vendor or community, and for some DBMSs there are also reusable software components that simplify the connection and interaction with the database and that can be used by other client software tools.

Software developers may rely on a generic and reusable software component to manage distinct DBMSs using the same application code, e.g. the Java Database Connectivity (JDBC) or the Open Database Connectivity (ODBC). These software components provide an unified way for the application to interact with the database, and internally they use a DBMS-specific driver to translate the generic requests to the DBMS-specific ones (and the DBMS-specific responses into generic ones). (Hamilton et al., 1997; Geiger, 1995)

2.3 DIGITAL PRESERVATION OF RELATIONAL DATABASES

This section focuses on important topics and specific challenges related to preserving relational databases.

2.3.1 *The E-ARK Project*

The E-ARK project aims to harmonise currently fragmented archival approaches. It attempts to provide an overarching methodology addressing business and operational issues, and technical solutions for ingest, preservation and re-use. The project's main goal is to provide a single, scalable, robust approach capable of meeting the needs of diverse organisations, public and private, large and small, and able to support complex data types, specially relational databases.

By co-operating with commercial systems providers, E-ARK aims to create and pilot a pan-European methodology for electronic document archiving, synthesising existing national and international best practices, that will keep records and databases authentic and usable over time. The practices developed within the project attempt to reduce the risk of information loss due to unsuitable approaches to keeping and archiving digital information.

The project uses existing tools and services developed by partners to implement and pilot the methodology in various national contexts. This allows the institutions (and their clients) to assess, in an operational context, the suitability of those state-of-the-art technologies.

The pilots are an important part of the project, as they are piloting end-to-end OAIS-compliant e-archival services covering ingest, vendor-neutral archiving, and reuse of structured and unstructured data, thus covering both databases and records, addressing the needs of data subjects, owners and users. Furthermore, national archives running E-ARK pilot instances will serve as an example for others wanting to adopt the new e-archiving open system.

The pilot and methodology also focuses on the essential pre-ingest phase of data export from source systems. It is in this phase that the Database Preservation Toolkit is used to harvest representations of databases (to the SIARD 2 format) to be included in a SIP. The Database Preservation Toolkit is used on the pre-ingest because no other representations are suitable for the transfer of information to the archive. Access to the live database system is necessary in order to harvest all the necessary information directly from the source, i.e. the live database system, as, for example, the content of views may be very hard to replicate outside the original database system.

The pilot integrates tools currently in use in partner organisations, and provides a framework for providers of these and similar tools ensuring compatibility and interoperability.

The E-ARK project suggests the use of a "Free and Open Source Software" licensing model, like the Apache or GNU licensing models, for new software, which allows commercial suppliers to incorporate the project outputs (particularly the open interfaces for pre-ingest, ingest, archival, access and re-use) into their own systems, further enhancing their longevity.

The outputs of the project are sustained by the project partner The DLM Forum, comprising 22 national archives and associated commercial and technical providers. In addition, The Digital Preservation Coalition, also a project partner, promotes best practices in this area. (E-ARK Project, 2014)

2.3.2 *Significant Properties*

The E-ARK project attempted to define the significant properties for relational database objects, since there was no consensus on which properties were significant. After contributions from various institutions, public and private, as well as creators and users of existing database preservation formats, the project was able to decide on which properties are significant. This lead to the creation of a format suitable for use with a format migration preservation strategy that could maintain all these significant properties: the SIARD 2.

Contents

It is essential to save the database content information, i.e. the cell values. This content might have to be transformed to comply with the format guidelines, and that transformation is acceptable as long as no information is lost. For instance, the date and time "`Wed, 13th July 2016, 15:45, UTC`" can be converted to "`2016-07-13T15:45:00Z`"[14] but not to "`2016-07-13`" since the time and time zone would be lost. Other challenges include saving very large numbers or very precise floating point numbers, as they risk going over the representational limits.

The LOBs are also cell values and are thus considered significant.

Relational structure

The relational structure is all the information needed to preserve the relations between data in the database. It is composed by the schemas, the keys, the tables, their columns, their data types, etc., as they are essential to experience the relational aspects of a database in the future, because these aspects help understand concepts that may be implied by these relations.

14 The `Z` character indicates UTC time zone, and the `T` character serves to separate the date from the time. These rules are defined in the ISO Standard 8601:2004 (E).

Since user defined data types can also be used as column data types, they are also a significant property.

There is a threat to the preservation of databases arising from the diversity of SQL flavours and data types, since not all data types implemented by DBMSs are defined in the SQL ISO Standard 9075:2008 (E) and their definition might end up being lost in time if that DBMS becomes obsolete. To try to avoid losing data integrity, an equivalent ISO-compliant data type could be preserved alongside the original type, and the ISO-compliant type domain should support all the values that are supported by the original type. Choosing an incorrect data type could result in incomplete or inaccessible cell values, and for this reason it is really important from a digital preservation standpoint to select an adequate data type.

Behaviour information

The database's behaviour aspects are also significant properties, as they might be necessary to understand how the database operated. The behaviour aspects do not include interactions between the database and any external software, but rather constraints, triggers and other actions that occurred on an event-based manner.

The significant behaviour properties are users, roles, permissions, views, triggers, routines (functions and procedures) and constraints (such as disallowing the `NULL` value or disallowing integer values outside a predefined range).

Descriptive information

The descriptive information present in the database (i.e. descriptions for columns, data types, triggers, etc.) are also significant properties, although they can be considered redundant if the database's data dictionary is also preserved.

External software interactions

Databases usually exist to serve data to an application (e.g. a website), and one might argue that the original application needs to be preserved alongside the database to correctly experience the database information, i.e. to experience it as the original users experienced it before it was archived. Attempting to preserve the original software and database system would require using an emulation strategy, which would probably be more expensive (than format migration), it might be impossible if the software (or part of it) is protected by intellectual rights laws, and hinders the information reuse (because the original system is emulated, and it might not provide the functionality needed by the consumer). (Ferreira, 2006)

The SIARD 2 format was created to be used with a format migration strategy, therefore its specification does not define a way to save the original software that interacted with the database. Although the producer could describe the software interactions with the database as descriptive information (i.e. add it as a description for tables, columns, triggers, etc.). This documentation alongside other representations like SQL dumps and other formats is to be gathered in the SIP and then transported to the AIP. Documenting or preserving software is a complex subject with approaches of their own, and a whole research domain that studies it. The focus of SIARD 2, and of this work, is on the information within the DBMS, and that is generally acceptable. DBMS functionality supporting part of the application layer (e.g. forms in a Microsoft Access database) may be included via some other means (adding documentation, using a software preservation strategy, etc.).

For these reasons the external software may be considered a significant property, but it will not be considered as such in this work.

2.3.3 *Preservation Formats*

A few open formats have been created to preserve relational databases using a migration strategy. They are briefly analysed in this section, along with references to the software used to create the format.

DBML

The Database Markup Language (DBML) format, created at University of Minho, was designed to preserve a relational database's structure and data. This format saves supported database elements in a single XML file, which can then be validated against a corresponding XSD. It is used by the first version of the Repository of Authentic Digital Objects (RODA)[15] software to preserve databases at the Portuguese National Archives[16]. (Ramalho et al., 2007a,b; Jacinto et al., 2002)

SIARD

The Software Independent Archival of Relational Databases (SIARD) is an open file format specially designed for the long-term archiving of relational databases. It consists of several XML files that are packaged in a ZIP container[17]. The format specifies the use of one XML file for metadata information which format is based on the SQL ISO Standard 9075:1999 (E), and distributes the database contents over several XML files, one for each table. This format includes XSD files to validate the XML file contents and supports storing LOBs

15 Current version of RODA is available at
`http://www.roda-community.org/`

16 Direcção-Geral do Livro, dos Arquivos e das Bibliotecas

17 Using the ZIP-64 format without compression or compressed using the deflate compression algorithm (ZIP 6.3.3, 2012), and having extension "`.siard`".

as hexadecimal-encoded strings[18] or as individual files in specific directories inside the SIARD package (Swiss Federal Archives, 2008). The SIARD format was developed by the Swiss Federal Archives in 2007 and is being used by many archives worldwide. In 2013, the SIARD format became a Swiss E-Government Standard (eCH-0165) (Bruggisser et al., 2013).

SIARD-DK

The SIARD-DK is a variation of the SIARD format created by the Danish National Archives to fit their specific needs. It includes support for case files[19] that can be used as a common point between the preserved databases. This format is more flexible than the original SIARD regarding LOB file location, since the archives had a lot of these files and they needed a format that could store them more efficiently; they also needed a way to easily verify if any of the files were corrupted, so they added a file that list all files and their checksums. SIARD-DK can also be an uncompressed folder, whereas the original SIARD could only exist as a ZIP file, the motivation for this was to avoid using extra storage to extract the databases before use. (Danish National Archives, 2010)

ADDML

The Archival Data Description Markup Language (ADDML) is the format used by the Norwegian National Archives and the Swedish National Archives to describe collections of data files. The format stores general metadata about the original database format, system, and database structure; it also holds content as plain-text files. The format can also contain and reference files in other formats. (Norwegian National Archives, 2011)

Others

There are other formats that use some combination technologies, like XML, ZIP and CSV; but lacked adoption by the digital preservation community and their support was dropped.

2.3.4 *Migration Tools*

This section presents a brief description of several tools that can be used to convert databases between different formats.

18 A textual representation of a binary format in which each byte is translated into two hexadecimal digits (Gordon and Nadig, 1977).

19 A case file is "a file relating to one or more transactions performed totally or partly in a structured or partly-structured way, as a result of a concrete process or activity". (European Comission, 2008)

SIARD Suite

The SIARD Suite is a free closed-source Java software application developed by the Swiss Federal Archives to simplify the archiving of relational databases. The software supports Oracle, Microsoft SQL Server, MySQL, DB/2, Microsoft Access and SIARD 1 database formats. The suite also includes SIARD Edit which supports editing SIARD 1 metadata and allows basic viewing of database content.

KOST-Val

KOST-Val is an open-source tool created by KOST-CECO to validate files in SIARD 1 format. It can also validate other non-database formats such as JPEG, JP2, TIFF and PDF/A.

RODA

RODA is a complete digital repository solution that delivers functionality for all the main functional units of the OAIS reference model. RODA is capable of ingesting, managing and providing access to the various types of digital content produced by large corporations or public bodies. RODA is based on open-source technologies and is supported by existing standards such as the Open Archival Information System (OAIS), Metadata Encoding and Transmission Standard (METS), Encoded Archival Description (EAD), Dublin Core (DC) and PREMIS (Preservation Metadata).

The first RODA version included an embed Database Preservation Toolkit prototype that used DBML as the preservation format of choice, converting databases to DBML in the ingest phase, and in the access phase loading the database into a modified PhpMyAdmin instance that could display the database information.

Pentaho Data Integration

Pentaho's Data Integration is focused on preparing data to be fed to business intelligence tools, but it is possible to use it to convert database contents between different DBMSs and transform the data along the way. It provides visual tools to orchestrate the migration, but it lacks support for long-term preservation formats.

CHRONOS

CHRONOS is a commercial software application that archives databases for long-term readability and usability. The software creates ZIP archives containing database data in comma separated value (CSV) files and saves metadata (including object names, data types, data ranges, commentaries and constraints) in XML format. It supports partial and ongoing database archiving, although no specification of the internal format used by CHRONOS is currently available, posing a risk to the preservation process in case the company decides to drop support for the tool. (Brandl and Keller-Marxer, 2007; CSP Software GmbH & Co. KG, 2015)

DeepArc

DeepArc was developed by the National Library of France to transform relational database content into XML for archiving purposes. It is part of the International Internet Preservation Consortium tool suite for web archiving. DeepArc is an open-source graphical editor which allows users to map an existing relational data model to one or several target data models, specified as XML Schemas. The purpose of this tool is to migrate database structure and content to an open-source, structured format that will create or retain the link between the document and its information.

HPAIO

Hewlett-Packard Application Information Optimizer (HPAIO) is a software application created by HP to relocate inactive data from production systems and legacy databases while preserving data integrity and access. It archives data as XML or CSV documents but embeds binary image files within the XML. Over time, this approach could hinder the monitoring of these embedded objects to prevent format obsolescence. (Fitzgerald, 2013)

2.3.5 *Viewers for Preserved Databases*

This section lists some tools that can be used to explore and search a database kept in a preservation format.

SOFIA

SOFIA (Search and Find in Archives) can load AIPs containing databases, create the respective DIPs and present them in a way that allows searching and filtering database records. This tool is developed and used by the Danish National Archives.

This solution loads the preserved database from SIARD-DK to a Microsoft SQL Server instance, and then the database can be accessed and searched using a specific client application. Sofia supports adding new views and search forms, used by non-technical personnel to retrieve desired information from the database.

Modified phpMyAdmin

A modified phpMyAdmin[20] version was used by RODA to display archived database information. The original phpMyAdmin is an open-source tool that can be used to manage databases in a MySQL system. The modified version aimed to use the existing presentation capabilities from phpMyAdmin, but disallow changes to the database; this way the database could be accessed and searched without any kind of changes to the database (e.g. adding or modifying records or tables, or removing information).

RODA used an embed Database Preservation Toolkit prototype to convert a DBML database to a MySQL system, and then provided a modified phpMyAdmin instance for consumers to access the database.

Having introduced the digital preservation field, with special focus on the preservation of relational databases, the next chapter focuses on describing the use-cases and requirements for database preservation tools.

20 *phpMyAdmin* is a free open-source software tool intended to handle the administration of MySQL over the Web. More information about this tool is available at `https://www.phpmyadmin.net/`.

3 METHODOLOGY

This chapter details the problem and its requirements, as well as discussing some possible solutions and choosing the most conceivable one. Lastly, the method used to validate the solution is described.

3.1 USAGE SCENARIOS

The E-ARK project defined multiple use-cases for the pre-ingest and access tools with respect to database archiving: (Alföldi et al., 2014; Alföldi and Réthy, 2016)

The information producer using the migration tool during the pre-ingest phase

During the pre-ingest phase, the producer can use the migration tool to convert a database to SIARD 2. After adding some documentation and other information, the producer can deliver the database to the archive. This procedure is depicted in the left part of figure 7 and the following usage scenarios correspond to the right part of the same figure.

The consumer using the database viewer to explore a database

The archivist can set up the database viewer to provide access to a specific database and allow the consumer to search and filter database records at will.

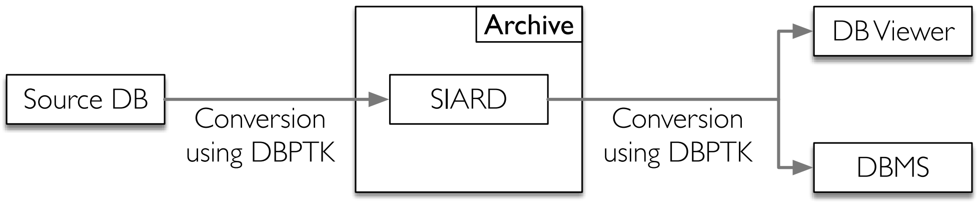


Figure 7.: General usage scenario for an archive

The archivist preparing a database for access or re-use

The archivist (or a database specialist) can use the database migration tool to convert the preserved database to a live DBMS and perform changes in the database. These changes might include rearranging data and remove sensitive information. The archivist should be able to migrate the modified database to SIARD 2 and submit the new SIARD 2 to the archive as a different version of the same database, or provide access to the modified database, or a combination of both.

The researcher performing complex queries and analysis

The researcher may initially act as a consumer, requesting access and exploring databases until a suitable database is found. The researcher could then request access to the suitable database in SIARD 2 format, obtaining it in a DIP, and use the migration tool to add it to a DBMS with advanced analytical capabilities. This allows a researcher to use Data Mining and Online Analytical Processing (OLAP) techniques to research archived databases.

Global use-case perspective

The diagram in figure 8 details how the use-cases and the tools' tasks are combined. All the use-cases from pre-ingest and access phases have been merged in this general use-case, to provide a global perspective on the expected work and information flow.

3.2 REQUIREMENTS

From the use-cases, the E-ARK project defined broad requirements that pre-ingest and access tools must fulfil, covering the typical goals and corresponding archival scenarios. This work focuses on handling the migration of relational databases to SIARD format (in the pre-ingest phase) and providing access to the database information (in the access phase). (Alföldi et al., 2014; Alföldi and Réthy, 2016)

The requirements for this work were derived from the requirements defined by E-ARK, because some of the requirements were too generic or too specific and needed to be normalised with regard to the scope of this work.

The requirements for the tool providing the conversion of relational databases (in the pre-ingest phase) are:

1. The tool should support the extraction of relational databases from live systems to the SIARD format;

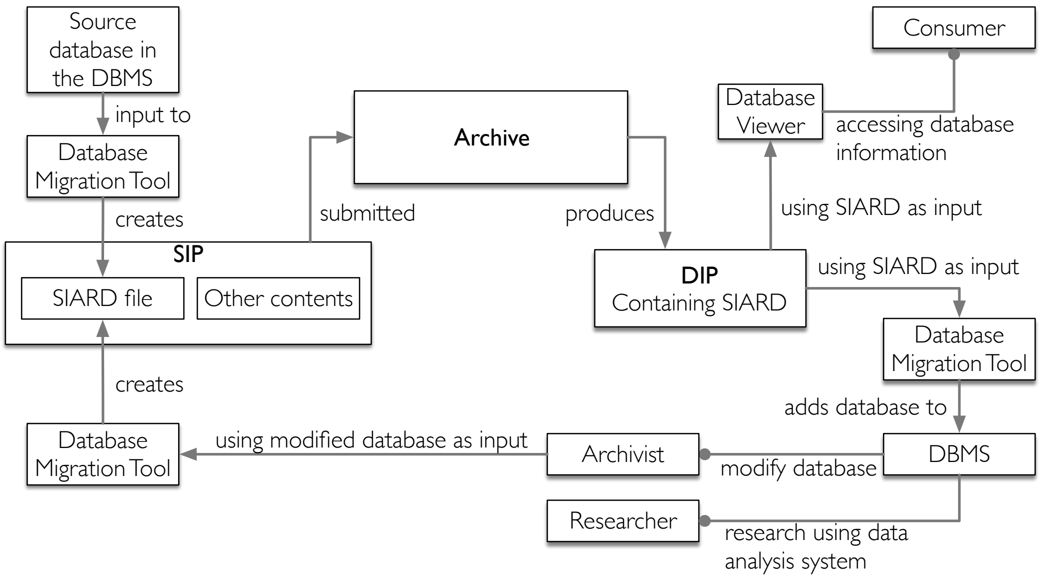


Figure 8.: Combined E-ARK database ingestion and access use-cases

2. The tool should support extracting databases from the current most popular RDBMSs;
3. The tool should support transforming the database from SIARD back to a RDBMS.

The requirements for the tool providing access to the database information are:

1. It must be possible to transform and package the database in a DIP, to provide access to the database information via a package or via a disseminator;
2. An information consumer must be able to access the database information;
3. An advanced consumer, like a researcher, must be able to re-use the database. For instance, a researcher could load the dataset into a data analysis or analytical processing system in order to extract usable information;
4. A consumer may request a database containing sensitive information that can not be freely distributed. In this case, an archivist must be able to create a new version of the database that excludes the sensitive information, and provide access to the new version;
5. An archivist must be able to create a new version of the preserved database that introduces some modifications, like rearranging the data to be easily understandable by a consumer, and provide access to the new version;
6. Any new versions of preserved databases may be preserved for later re-use without the need to re-create them.

3.3 APPROACH

This section describes the accepted approaches to build the tools, as well as significant alternatives.

For the database migration tool, the most significant design choice was already implied in the use-cases: using the same tool to convert the databases to the preservation format and from the preservation format back to the DBMS. This way some code could be shared for both conversion directions, reducing the work needed to maintain the tool.

However, multiple design approaches were considered for the database viewer. The three most significant approaches are described in this section, starting with a simple approach, followed by an approach that was implemented in a couple European archives and then describing the accepted design.

First approach

The first approach is to directly access the database information in the SIARD file, which is distributed across multiple XML files. Some technology like XPath(Clark et al., 1999) could be used to retrieve the required information from the XML files. This approach would not use any kind of preparation of the database or indexing of information.

The main problem of this approach is the time it would take to search information in big tables. Since XPath is a generic technology to select nodes in XML, and since the information would not have been previously indexed or optimized in any way, the XPath process would need to parse the whole XML file (or files) for every search.

This approach might work for demonstration databases or for demonstration purposes, but it would not be feasible to use with a real database.

A different XPath-based approach could be used, one that would pre-process the SIARD file and index it to provide faster XPath queries. This kind of solution would be similar to the accepted solution, as it also pre-processes the information to be easier to retrieve.

Second approach

The second approach attempts to provide access to preserved databases by converting the SIARD database to a DBMS database and using some client application to search the database on the DBMS (via search forms or SQL).

This approach has been implemented in the Portuguese National Archives, by loading the databases to MySQL and providing access to the databases via a modified *phpMyAdmin* web interface.

This approach has also been implemented in the Danish National Archives, by loading the databases to a Microsoft SQL Server instance and providing access to the database via a custom software tool, SOFIA[1].

This approach relies on a DBMS product and the functionality it provides. This may lead to data loss, as some features from the original DBMS might not be supported by the DBMS in which the database was loaded.

This approach also relies on using a RDBMS to hold information from multiple databases, which is an approach that would need special adjustments to each database, because most RDBMS serving a large database need some special adjustments to handle high volume data, and this system would have to serve multiple large databases. The main problem concerning both these approaches is the resources needed to run all databases in a single system and the difficulty to horizontally scale the application's data layer. Both mentioned implementations suffer from this problem, sometimes taking a couple of minutes to perform a search in one database, or even failing to complete the search at all. (Pokorny, 2013)

The accepted approach

The accepted approach, similarly to the second approach, attempts to provide access to preserved databases by converting the SIARD database to a different system and using some client application to access the database. However, this approach considers using NoSQL[2] technologies to provide faster searches and horizontal scalability[3].

To provide access to the database, a custom client application would need to be created, or the tool could be designed as a web application, in which case a browser would be used to access the web interface.

This approach is an improvement compared to the second approach, whilst retaining similar functionality. Using a NoSQL data layer, this approach would not be able to support using SQL to query the database, but SQL support may be available by using the database migration tool to convert the SIARD database to a RDBMS supporting SQL querying.

1 SOFIA stands for *Search and Find in Archives* and is used to load and provide access to the databases.

2 Not-only SQL (NoSQL) technologies provide a mechanism for storage and retrieval of data which is modelled in means other than the relational model. (Pokorny, 2013)

3 Adding processing power to a system might be achieved by scaling vertically, by increasing the available resources to a machine in the system, or by scaling horizontally, by increasing the number of machines in the system. Scaling horizontally introduces some advantages like fault tolerance (the failure of one machine may not be noticeable to a client using the system) at the expense of having to manage a group of machines instead of a single one. (Michael et al., 2007)

3.4 VALIDATION

The validation process is detailed in three separate phases. The first two validation phases cover the specific requirements for each tool and are present in sections 4.2.6 (for the Database Preservation Toolkit) and 4.3.5 (for the Database Visualization Toolkit). A global validation is detailed in chapter 5, covering the E-ARK requirements and focusing on the validation provided by the E-ARK pilots, via testing according to use-cases in real-world environments.

Having defined the use-cases and requirements for database preservation tools, the next chapter will introduce the database preservation format and tools developed in this work.

4 SOLUTION

The selected strategy to archive databases is the migration of information into a database preservation format, that would be accepted by archives as a representation of the database[1]. The preservation format is a new version of the SIARD format, created using feedback from current users of the SIARD format and its creators, as well as users of other database preservation formats.

The selected strategy to provide access to databases is to develop a database viewer that uses scalable information retrieval software. The database information is pre-processed and loaded to the scalable system, and the graphical user interface provides a way to search the database that is aimed at non-technical personnel.

This chapter details the second version of the SIARD format, which is the selected database preservation format. The Database Preservation Toolkit which is the tool to convert databases to and from database preservation formats. And the Database Visualization Toolkit, which is used to provide access to archived databases.

4.1 SIARD 2

The second version of the Software Independent Archiving of Relational Databases (SIARD) format emerged from lessons learnt by creators and users of database preservation formats. The format aims to be backwards compatible, so it shares multiple features with its first version. It is based on an updated version of the SQL standard, SQL ISO Standard 9075:2008 (E), to support a wider range of database elements, such as arrays and user defined types. Whenever further functionality is necessary, SIARD relies on stable and widely accepted technologies, for instance Unicode (The Unicode Consortium, 1996) and URI (Masinter et al., 2005).

SIARD supports adding descriptions to virtually every database element (database, schema, table and their keys, columns, triggers and constraints, user defined types and their fields, routines and their parameters, views and their columns, users, roles and privileges). From

1 The format preference policy is defined by the archive. A common example of a format preference policy is the preference for the Portable Document Format (PDF) over the Microsoft Word 97 format.

```
/ ............................................................ zip file root
├── header/ ............................... folder for database metadata
│   ├── metadata.xml
│   ├── metadata.xsd
│   └── version/
│       └── 2.0/ ............ empty folder indicating SIARD version 2.0
└── content/ ................................ folder for database content
    └── schemaM/ ......... M is an integer that uniquely identifies the schema
        └── tableN/ ......... N is an integer that uniquely identifies the table
            ├── tableN.xml .................... same N used in tableN/
            └── tableN.xsd .................... same N used in tableN/
```

Figure 9.: Basic SIARD 2 directory structure.

a preservation standpoint, this feature is very important, as it not only allows producers to fully describe the database but also keeps that information efficiently organized and close to the elements themselves.

Since DBPTK was the first tool to automate the migration of databases to the SIARD 2 format, many suggestions to improve the format were made during the tool's development. Some of the suggestions were to clarify certain parts of the format specification, making it stricter and removing ambiguities. Other suggestions were to change the specification to make it easier to implement and overcome certain technical limitations when migrating the database to SIARD. All suggestions were considered and used to improve the specification (which was still in development at the time), and the author of this work is also a named contributor in the SIARD 2 standard specification, in recognition of his contributions to the format.

The SIARD format treats the whole database as a single document to be archived. In its most basic form, it is a ZIP file that contains a hierarchy of folders and files of XML and XSD (XML Schema) format, illustrated in figure 9. The XML files inside the SIARD ZIP file hold database metadata information and contents, and the accompanying XSD files can be used to validate the XML file contents.

4.1.1 *Database Metadata*

The `metadata.xml` file contains database descriptive, structural and behavioural information, organized in an hierarchical manner. Listing 1 introduces a trimmed down example of a `metadata.xml` file for the Sakila database[2], which will also be used in following examples,

2 Available at `https://dev.mysql.com/doc/sakila/en/`.

and illustrates the kind of database metadata information that can be present in a SIARD file.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<siardArchive version="2.0"
  xmlns="http://www.bar.admin.ch/xmlns/siard/2.0/metadata.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd metadata.xsd" >
  <dbname>sakila</dbname>
  <description>
    The Sakila sample database was initially developed by Mike Hillyer, a former member of
↪   the MySQL AB documentation team, and is intended to provide a standard schema that can
↪   be used for examples in books, tutorials, articles, samples, and so forth. The Sakila
↪   sample database also serves to highlight some features of MySQL such as Views, Stored
↪   Procedures, and Triggers. The Sakila sample database is designed to represent a DVD
↪   rental store.
  </description>
  <archiver>Bruno Ferreira</archiver>
  <archiverContact>email: pg29030@alunos.uminho.pt</archiverContact>
  <dataOwner>MySQL team</dataOwner>
  <dataOriginTimespan>Early 2005 to March 2006</dataOriginTimespan>
  <lobFolder>content</lobFolder>
  <producerApplication>Database Preservation Toolkit</producerApplication>
  <archivalDate>2016-05-24+01:00</archivalDate>
  <clientMachine>young</clientMachine>
  <databaseProduct>MySQL 5.5.5-10.1.11-MariaDB-1~trusty</databaseProduct>
  <schemas>
    <schema>
      <name>sakila</name>
      <folder>schema1</folder>
      <description>
        This schema encloses all tables form the original MySQL database.
      </description>
      <tables>
        (tables are omitted in this example)
      </tables>
    </schema>
  </schemas>
  <users>
    <user>
      <name>root@localhost</name>
      <description>Database administrator user</description>
    </user>
  </users>
</siardArchive>
```

Listing 1: Example of a metadata.xml file (omitting tables)

Although not all the elements in listing 1 are mandatory according to the SIARD 2 specification, the optional elements should be used when information is available, in order to improve the quality of the preserved database. The list of metadata elements supported by SIARD 2 and their description is available in appendix A.1.

In the SIARD metadata XML file, the `schema` element must have a name and a list of tables, and can optionally include a textual description and lists of views, routines and UDTs associated with the schema. The "`schema`M" element in figure 9, corresponding to the schema folder name, is also specified in this element and is used to locate table content files (see section 4.1.2).

The list of tables contains a sequence of table elements like the one in listing 2. A description of the characteristics of the table element in the SIARD metadata XML file is available in appendix A.2.

As shown in the listings, tables are enclosed in schema elements. Besides containing the tables, the schema element has a name, a description, and three lists: of types (UDTs), views and routines.

The **type** element contains very technical properties (based on the ones present in the SQL ISO Standard 9075:2008 (E)) used to describe an advanced or structured data type, and needed to understand the UDT in the future.

The **routine** element is composed of a name, a description, its parameters (each having a name, description, type information, and mode[3]), the original routine source code (in whatever language it may be written), and the routine code expressed in SQL that complies with ISO Standard 9075:2008 (E).

The **view** element has a name, a description and the list of columns in the view (each column is structurally identical to table columns). The element may also contain the original view definition expressed in (flavoured) SQL and the same definition expressed in SQL that complies with ISO Standard 9075:2008 (E).

The SIARD 2 format enforces the presence of some elements and properties that are considered essential, such as the database contents and relational structure. The technical details related to the database's behaviour are not mandatory as one could argue that a good description would be worth more than a query expressed in a deprecated (and maybe undocumented) SQL flavour. The `metadata.xsd` file can be used to check the `metadata.xml` and ensure that all required elements are present and valid.

3 Possible modes are "`IN`", "`OUT`" and "`INOUT`"; representing input parameters, output parameters or parameters that are used as both input and output.

```xml
<table>
  <name>address</name>
  <folder>table2</folder>
  <description>This table contains addresses</description>
  <columns>
    <column>
      <name>address_id</name>
      <type>SMALLINT</type>
      <typeOriginal>SMALLINT UNSIGNED</typeOriginal>
      <nullable>false</nullable>
      <description>The address unique id.</description>
    </column>
    <column>
      <name>address</name>
      <type>CHARACTER VARYING(50)</type>
      <typeOriginal>VARCHAR</typeOriginal>
      <nullable>false</nullable>
      <description>First address line</description>
    </column>
    <column>
      <name>address2</name>
      <type>CHARACTER VARYING(50)</type>
      <typeOriginal>VARCHAR</typeOriginal>
      <nullable>true</nullable>
      <description>Second address line</description>
    </column>
    <column>
      <name>district</name>
      <type>CHARACTER VARYING(20)</type>
      <typeOriginal>VARCHAR</typeOriginal>
      <nullable>false</nullable>
      <description>Address district</description>
    </column>
    <column>
      <name>city_id</name>
      <type>SMALLINT</type>
      <typeOriginal>SMALLINT UNSIGNED</typeOriginal>
      <nullable>false</nullable>
      <description>Address city (id)</description>
    </column>
    (other columns omitted)
  </columns>
  <primaryKey>
    <name>PRIMARY</name>
    <description>A unique address identifier.</description>
    <column>address_id</column>
  </primaryKey>
  <foreignKeys>
    <foreignKey>
      <name>fk_address_city</name>
      <referencedSchema>sakila</referencedSchema>
      <referencedTable>city</referencedTable>
      <reference>
        <column>city_id</column>
        <referenced>city_id</referenced>
      </reference>
      <deleteAction>NO ACTION</deleteAction>
      <updateAction>CASCADE</updateAction>
      <description>Relates an address to its city</description>
    </foreignKey>
  </foreignKeys>
  <candidateKeys>
    <candidateKey>
      <name>PRIMARY</name>
      <description>A unique identifier column that was created for that
 ↪  purpose.</description>
      <column>address_id</column>
    </candidateKey>
  </candidateKeys>
  <rows>603</rows>
</table>
```

Listing 2: An example of a table expressed in the metadata.xml file

4.1.2 *Database Contents*

The `table`N`.xml` files (see figure 9) correspond to each of the database tables and hold the content of the rows and cells from that table. Each table XML file is accompanied by a XSD file that can be used to perform some validation on the database contents according to their definition, including presence (cells of columns marked as "not null" must be present in all rows), value type (including date, time, number and binary sequence validation), and cell values length.

The listing 3 illustrates, in practice, how contents are arranged in a `table`N`.xml` file. Each row in the table is stored in a **row** element, and inside that element are multiple **c**X elements corresponding to the contents of the X^{th} column in that row.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<table
  xsi:schemaLocation="http://www.admin.ch/xmlns/siard/2.0/schema1/table2.xsd table2.xsd"
  xmlns="http://www.admin.ch/xmlns/siard/2.0/schema1/table2.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <row>
    <c1>1</c1>
    <c2>47 MySakila Drive</c2>
    <c4>Alberta</c4>
    <c5>300</c5>
    (other columns omitted)
  </row>
  <row>
    <c1>2</c1>
    <c2>28 MySQL Boulevard</c2>
    <c4>QLD</c4>
    <c5>576</c5>
    (other columns omitted)
  </row>
  <row>
    <c1>3</c1>
    <c2>23 Workhaven Lane</c2>
    <c4>Alberta</c4>
    <c5>300</c5>
    (other columns omitted)
  </row>
  (remaining rows omitted)
</table>
```

Listing 3: An excerpt of Sakila's table "address" as present in the `table`N`.xml` file

The `table`N`.xml` file also contains information about the LOB cells, which are described in section 4.1.4.

4.1.3 *SIARD in an OAIS Archive*

As a long-term storage format for relational databases, the SIARD format is designed independently of package structures such as the SIP, AIP and DIP in the OAIS reference model. However, it is assumed that a database in SIARD format is archived as part of an information package, together with other documents related to the database (e.g. database

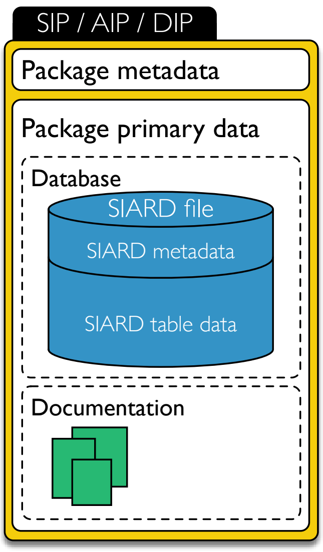


Figure 10.: Diagram of an information package containing a SIARD file (Faria et al., 2016)

documentation, business documents relevant to understanding the database, manuals of the applications that interacted with the database, etc.), as depicted in figure 10.

Following this design, it is possible to use the SIARD format in a OAIS-compliant archive, independently of the information package format being used.

4.1.4 *Handling LOBs*

Databases containing a large number of LOBs tend to be larger and so they must be handled in a specific and scalable way. The SIARD specification indicates three different strategies for storing LOBs:

INLINE
: Store the LOB inside the `table`N`.xml`, as a sequence of hexadecimal digits.

INSIDE
: Store the LOB in a separate file inside the SIARD archive.

OUTSIDE
: Store the LOB in a separate file in the file system, outside the SIARD archive.

The “inline” and “inside” strategies are defined in the SIARD specification. The “outside” strategy is simply referred but not detailed, and while this allows greater freedom and storing LOBs in multiple locations or storage devices outside the SIARD archive, it also includes a greater risk of losing the files. To avoid this, and to fit conflicting requirements from different national archives, an "External LOB placement recommendation" was created.

The LOBs are always included or referenced in the `table`N`.xml` file, along with their length (in bytes for BLOBs and in characters for CLOBs) and a checksum (provided by the "digestType" and the "messageDigest" attributes) string that can be used to check the byte integrity of the LOBs.

4.2 DATABASE PRESERVATION TOOLKIT

This section describes the Database Preservation Toolkit (DBPTK) artefact, its purpose and requirements, its design and development phases, and how the tool was tested. The validation phase is described in chapter 5.

4.2.1 *Purpose and Requirements*

The manual creation of a SIARD file is impractical, as manually obtaining all the information from the database and saving it to SIARD is very error prone and time consuming. Therefore, there is a need for a tool to provide this functionality in an automated way, requiring as little input as possible from the information producer and avoid as much human-related errors as possible.

A few requirements were laid out before the development phase whilst others were added after user feedback. These requirements are listed below. In these requirements, the term "convert" means to migrate a database, whilst keeping it authentic and preserving its significant properties. The software must:

Req. 1: Be compatible with both Linux and Windows operating systems.

Req. 2: Be open-source, to make it easier to maintain and use in the future.

Req. 3: Be able to access the DBMS using an encrypted connection, to ensure the information security.

Req. 4: Be executable in remote environments that are accessible only via command line (i.e. without a graphical interface).

Req. 5: Convert databases from the most popular RDBMSs to SIARD 1 and SIARD 2 preservation formats.

Req. 6: Convert databases from multiple preservation formats back to a DBMS.

Req. 7: Export to SIARD 2 must support specifying some tables that should be excluded from the conversion process.

Req. 8: Support adding more input and output systems and formats.

Req. 9: Convert structural information without data loss, performing (lossless) conversions when necessary.

Req. 10: Convert content information without data loss, performing (lossless) transformations when necessary.

Req. 11: Convert descriptive information associated with various database elements.

Req. 12: Convert behavioural information; saving it as descriptive information when it is unsupported by the target system.

Req. 13: Be capable of handling databases containing millions of rows and hundreds of Gigabytes in size.

Req. 14: Report and document any possible data loss, the report must include all the information that was changed during the conversion.

4.2.2 *Design*

The DBPTK uses a modular approach, allowing the combination of an import module and an export module to enable the conversion between database formats. The import module is responsible for retrieving the database information (metadata and data), whilst the export module transcribes the database information to a target database format. Each module supports the reading or writing of a particular database format or DBMS and functions independently, making it easy to plug in new modules to add support for more DBMS and database formats. The conversion functionality is provided by the composition of data import with data export, as depicted in figure 11. (Ferreira et al., 2016)

The most popular relational DBMSs are[4]: Oracle, MySQL, Microsoft SQL Server, PostgreSQL, DB2 and Microsoft Access. From that list, Oracle, MySQL, Microsoft SQL Server, PostgreSQL and Microsoft Access were planned to be supported as input systems, and Oracle, MySQL, Microsoft SQL Server and PostgreSQL were chosen as the supported output systems.

The main framework that supports the import and export modules includes an internal data structure based on the SQL ISO Standard 9075:2008 (E) that is able to maintain all the information from the database. During the conversion, the import module extracts the database to these internal data structures and the export module converts them to some output system or format.

To address scalability and performance concerns, the software was designed to use an information streaming approach, in which the import module retrieves and pushes data to the export module, which outputs the data to some output format or system. Using this method, the application ensures minimal processing resources are used during the conversion.

Regarding the order of operations, the import module should obtain all the database metadata (schemas, tables, types, users, etc.) and pass it to the export module. The import

4 According to the ranking at `http://db-engines.com/en/ranking` (accessed on July 2016).

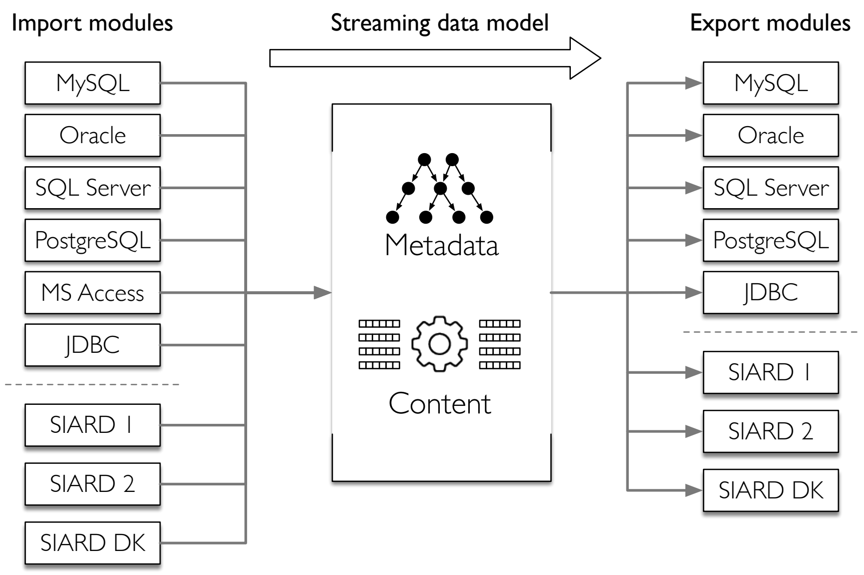


Figure 11.: Application architecture overview

module should then proceed to pass database data to the export module in an event-driven way. The sequence of operations is as follows:

1. The import module tells the export module to initialize the output **database**;
2. The import module obtains the **database metadata** and passes it to the export module;
3. The import module iterates over the **list of schemas**, and for each **schema**:
    1. The import module tells the export module to make the necessary preparations to handle that specific **schema**;
    2. The import module iterates over the **list of tables**, and for each **table**:
        1. The import module tells the export module to make the necessary preparations to handle that specific **table**;
        2. The import module obtains each **row** in the table and passes them, one at a time, to the export module. The export module should handle outputting the **row** to the output system or format.
        3. The import module tells the export module that all the **table**'s contents have been processed.
    3. The import module tells the export module that all the **schema**'s contents have been processed.
4. The import module tells the export module that the **database** has been processed.

When the export module receives a certain event, it should execute the necessary actions on the output format or system. When the export module finishes processing the current element, the import module advances to the next element until the whole database has been converted.

4.2.3 *Development*

The DBPTK currently supported DBMSes are Oracle, MySQL, PostgreSQL and Microsoft SQL Server; all of these DBMSs are supported as input and output systems. A Microsoft Access import module was also developed, to support having these databases as input to the application. All these modules use the Java Database Connectivity (JDBC) modules as a generic starting point, and then deviate as much as needed to account for functionality specific to each DBMS.

The base JDBC import and export modules, may enable conversion to or from any DBMS, given the correct configurations and dependencies (such as the appropriate JDBC implementation for the DBMS). Being DBMS-agnostic, this technology can be used to connect to a wide range of database products, however some specific features (e.g. data types) may not be supported and may hinder a fully successful conversion. Using this technology further simplifies adding modules, as they can inherit the base functionality from JDBC and override functionality that should work in a different way.

The SIARD modules are an essential part of DBPTK, being used to allow the conversion of databases to and from SIARD 1 and SIARD 2 database preservation formats. The SIARD export modules also support filtering out some database tables.

As all the import and export modules can be used interchangeably, any import module can provide information to any export module, and thus the software can be used to convert databases from DBMSs to preservation formats and from preservation formats back into DBMSs.

Regarding security and data privacy concerns, current import and export modules default to using secure (encrypted) connections to DBMSes if such a connection is supported by the DBMS.

Figure 11 depicts an overview of the information flow in the application, as explained in the design sub-section, with import modules as information providers, extracting the information from a source and mapping it into an internal application model; and export modules implementing the inverse process, by mapping information from the internal model to the target DBMS. This mapping may be specific for each DBMS module, as most DBMSes have specific or incompatible features.

In the first phase of the conversion, the database metadata (descriptive, structural and behavioural information) is fetched by the import module and transcribed to the target database management system or format by the export module. This phase is followed by the conversion of database contents. Using streaming and optimizing interactions with the database, the contents are converted, record by record, with low computing resource requirements. Finally, system resources are released, concluding the execution. While this is a common overview of a typical execution, specific modules can slightly diverge from this approach to improve performance and error handling.

The conversion is prioritized by firstly converting the database content information, secondly trying to keep the database structural metadata identical to the original database, and thirdly attempting to translate the database behavioural metadata to the target database. In practical terms, this means that in cases where the target DBMS does not support the data type used in the original database, an equivalent or less restrictive data type is used; this changes the database structure metadata, but avoids database content information losses. The database behavioural information is the last priority in the conversion because it is prone to failure (with a warning) due to source DBMSs that do not check the invariants and constraints imposed by behaviour like primary and foreign keys, or views which have DBMS-specific queries, incompatible with target DBMS[5] nor by the SQL ISO Standard 9075:2008 (E).

The internal application structure

Figure 12 introduces an overview of a database structure as a hierarchy. As most databases fit this structure entirely or partially, it is the structure used by all import and export modules, i.e. the internal data structure to which all the information is converted by the import module, and that the export module uses to migrate information to the output format or system. This structure is based on the SQL ISO Standard 9075:2008 (E), however there are some database systems, e.g. MySQL, that do not fit this structure entirely, as they have no schemas. In these cases, all the information that would be accommodated in a schema is moved up to the database component, resulting in a slightly different component that performs as both a database and a single schema, depicted in figure 13. DBPTK import modules work around this issue by treating the schema-less database as if it were a database containing a single schema, moving any tables, views, routines and user defined types to this schema.

Handling different SQL idioms

Most DBMSes implement SQL with slight deviations from the SQL standard. These derived query languages are commonly referred to as SQL flavours and make it difficult to create a set of queries compatible with the majority of DBMSes. To create queries, there is a query generator, based on the SQL standard, and serving as a base for a few flavour-specific

5 The conversion of SQL statements between different DBMSs is outside the scope of this work.

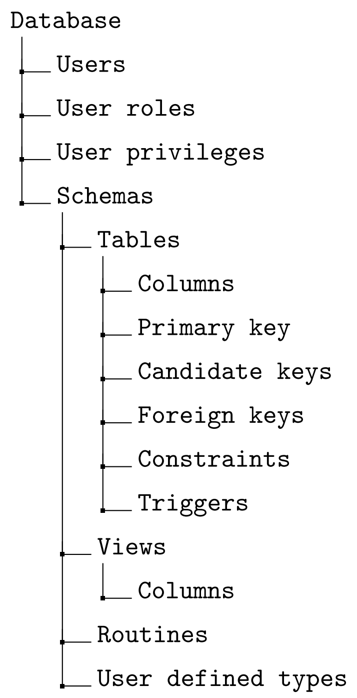


Figure 12.: Database structure hierarchy.

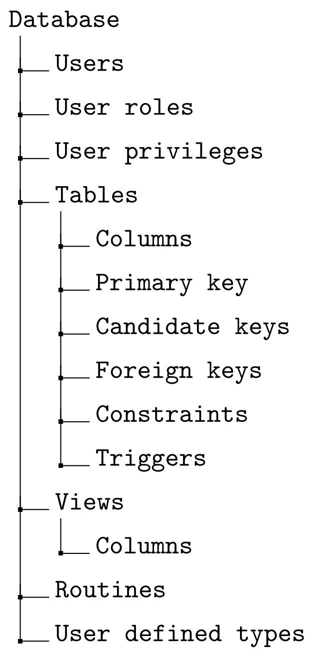


Figure 13.: Schema-less database structure hierarchy.

query generators. The import and export modules use the most specialized SQL generator considering the DBMS SQL flavour, guaranteeing equivalent functionality across different DBMSes.

SQL flavours often include new SQL data types or alias to standard SQL data types, but internal data types used in DBPTK are based on SQL standard data types. During the database conversion process, the import module maps the data types to appropriate internal data types, and the export module does the inverse process, by mapping the internal data types to data types supported by the target database management system or format. The aforementioned process is observable in figure 14.

Most flavoured SQL types are automatically converted to standard SQL types as they are obtained by the import modules, but there are a few cases that need to be handled specifically for each DBMS. An example of such case is the `YEAR` MySQL data type[6], depicted in figure 14, which the import module first perceives as representing a date, but is in fact a 4 digit numeric type (corresponding to the SQL ISO Standard 9075:2008 (E) standard type "`NUMERIC(4)`"). Since PostgreSQL `NUMERIC(4)` data type definition follows the SQL standard, that designation is used for the target data type.

The data type precision (or size) and scale usually corresponds to the first and second parameters of the data type definition. However, the semantics for those parameters may

6 MySQL `YEAR` data type documentation available at `http://dev.mysql.com/doc/refman/5.7/en/year.html`

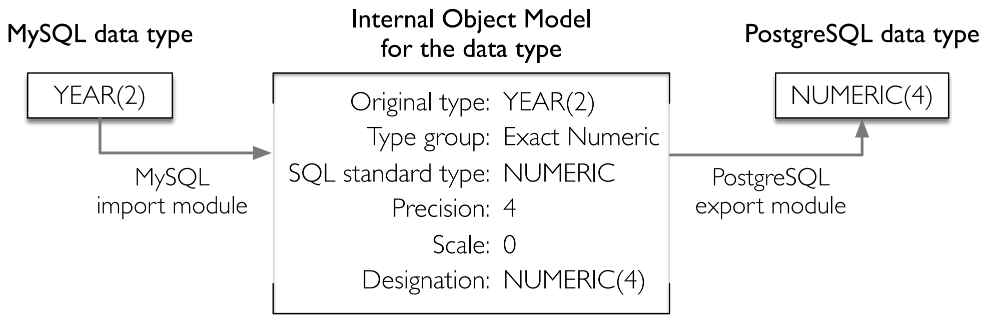


Figure 14.: Conversion of MySQL YEAR data type to PostgreSQL

also vary with the SQL flavour, requiring, for those cases, a specialized interpretation and conversion to an internal standard representation.

Due to prioritizing the database content information over the database structural metadata, the data type conversion does not ensure that the target type will be the same as the original type, but rather a data type that can be used to represent all values that can be represented by the original data type, without any data losses (i.e. the target data type domain contains original data type domain). An example of this could be the conversion of a data type `VARCHAR(500)` (capable of holding a variable length sequence of up to 500 characters) to an hypothetical DBMS in which the maximum number of characters supported by the `VARCHAR` data type is 200. In this case, the module would choose a `TEXT` data type (provided that it had the capacity to hold 500 or more characters), ensuring that all character sequences could be represented by the target data type without any loss of data.

The modules may also opt for using a different data type instead of a deprecated one. In some cases, the data type is changed to an alias data type with the exact same functionality, such as the `NUMERIC` and `DECIMAL` types on MySQL and Microsoft SQL Server.

Regarding database contents, some values obtained from the database may not be in a standard format and must be converted before use in the export module. Common examples of such values are the `DATE`, `DATETIME`, `TIMESTAMP` and other date and time data types. Due to basing its internal structure on the date and time ISO Standard[7], the DBPTK must convert some dates that are not provided in this format. Specific examples include the `YEAR` MySQL data type that must be converted to a numeric value in the range 1970 to 2069, inclusive.

The conversion of SQL statements between different DBMSs is outside the scope of this work.

7 The ISO Standard for Representation of Dates and Times. (ISO Standard 8601:2004, E)

Report data transformations and potential data loss

The DBPTK produces a report on each conversion registering all transformations and potential data losses during the conversion. The report is a text file, annotated with non-intrusive markdown[8] (to support an optional improved reading method by converting the markdown-annotated text to HTML, without reducing the readability of the plain-text report), which begins with the conversion start time and parameters, and then contains one transformation (e.g. of a data type) or potential data loss per line.

This file is important in an archival workflow to quickly ascertain the correctness of the created SIARD file.

Conversion performance optimizations

An optimization common to most DBMS export modules postpones adding database behavioural information until after the database content information is completely converted. If the behavioural information was to be added before the database content conversion, all inserted table records would be subject to a series of validations, such as primary key uniqueness or foreign keys presence, upon being inserted in the target database. Postponing the addition of these elements executes those validations only once, thus reducing the time needed to insert a table record in the target database. This approach also simplifies error handling when invariants were not being respected in the original DBMS.

Other optimizations are implemented, such as streaming LOBs from the database directly to the export module, whenever possible, instead of copying them to a temporary folder on the local system.

Handling LOBs

Large cell values are converted by the DBPTK using little computing resources, by never having the whole large object in the computer memory. This is achieved through streaming the LOB bytes from the source database to the target database. To achieve this, the import module does not immediately extract the LOB contents from the DBMS, instead retrieving an unique identifier for that LOB; when the export module starts to handle that specific LOB, the import module immediately resolves that identifier to a byte stream and provides the stream to the export module, so it can read from the stream and write the LOB to the output format or system.

Using this method, only a small portion of the large object is kept in the computer memory by the DBPTK, corresponding to the section being transferred (streamed) from the import module to the export module.

8 Markdown is a text-to-HTML conversion tool and annotation language for web writers. Markdown allows writing using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML). More information is available at `https://daringfireball.net/projects/markdown/`

When the unique LOB identifier can not be obtained, the import module streams the LOB to a temporary file, and when the export module starts to handle that specific LOB, the import module provides a stream that reads the bytes from the temporary file (and deletes it afterwards). This backup strategy is important because some DBMSs can not provide the LOB unique identifier.

This approach also moves the complexity of handling different LOB sources to the import module, and the export module only needs to handle reading the LOB bytes from a stream and outputting them in some way.

Support for different user interfaces

The import and export modules have been designed to self-describe their needed parameters. Based on this description, the command-line interface can handle the different parameters supported by the database import and export modules, being also able to list them and validate them when the application is executed.

This design produces code that is easier to maintain, since the developer can change the parameters for a module and the changes are automatically reflected in the command-line interface (or any other user interface adopting this strategy).

Other user interfaces could be designed to use the information provided by the modules, simplifying the interface development and maintenance. This greatly simplifies the work needed to develop, for instance, a graphical user interface.

4.2.4 *Deployment*

The application can be executed on any operating system running Java 7 or higher. At least 512MB of RAM are recommended, along with enough free disk space to store the outputs of the application, e.g. the SIARD files.

To archive a database, the Database Preservation Toolkit can be downloaded and executed on a machine that can access (locally or via network) the intended database. Executing the tool without parameters produces the extensive list of parameters supported for each import and export modules, which can be used to understand which parameters need to be used to convert a database format or system to a different one. The software, documentation and up-to-date examples are available in the application website:
`http://www.database-preservation.com`.

4.2.5 *Testing*

To ascertain the quality of the conversions made using DBPTK, a testing system was developed. The system was named *roundtrip testing* and is used to check if converting a database

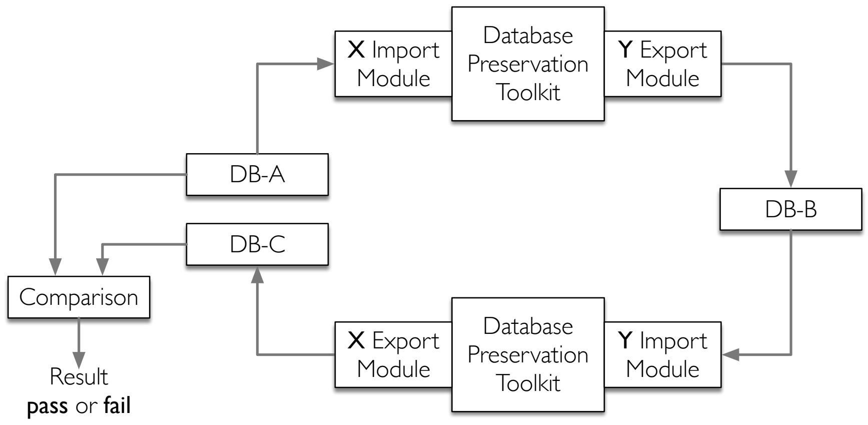


Figure 15.: The *roundtrip test*

to a different database management system or format and then converting it back to the original database management system or format results in any data changes or losses. For most purposes, this test is equivalent to an identity test, validating the completeness and correctness of the conversions.

The *roundtrip test* is described below and illustrated in figure 15.

1. Create a new database, *DB-A*, in a database management system or format, *DBMS X*, with the intended test data (e.g. a table containing LOBs);
2. Use DBPTK to convert *DB-A* to a different database management system or format, *DBMS Y*, creating database *DB-B*;
3. Use DBPTK to convert *DB-B* back to *DBMS X*, creating database *DB-C*;
4. Compare *DB-A* and *DB-C*, via a textual database *dump*[9]. The test passes if the database content information is unchanged.

Using this method, four modules are tested (two import modules and two export modules). The *roundtrip test* fails if any database content information is changed or lost during the conversions.

It is noteworthy that the comparison step accomodates a few expected and well-defined changes, e.g. the change of a type name to a normalized type name. This exception exists to accommodate for aliased data types and any incompatibilities between database management systems or formats. An example of this is the `YEAR(2)` data type from MySQL, which is changed to `NUMERIC(4)` when converting to PostgreSQL (see figure 14) and would be

9 A textual database *dump* is a text file describing the metadata and contents of a database, possibly using non-standard SQL and other DBMS specific instructions. This does not impact the comparison, since these files are always generated by the same system: *DBMS X*.

created as `NUMERIC(4)` when converting the database back to MySQL. In the roundtrip test it would be possible to use the original data type, since the test is using the same database system product and version, but that is not the intended use for the DBPTK. The DBPTK is intended to support information migration between highly different database systems, possibly separated by decades of development, and as such it should not be attempting to use the original data type in the target system.

The DBPTK test suite contains the roundtrip tests, along with unit and integration tests. These tests are executed when the application is compiled, ensuring that any change that breaks existing functionality is noticed. The code is also submitted to a continuous integration and continuous analysis systems, to ensure the tests also pass in different environment and to inform the developers of variations regarding the code quality.

4.2.6 *Validation*

This section details how each requirement was validated, i.e. how it was proven that the artefact achieves the requirements.

Requirement 1: the software must be compatible with both Linux and Windows operating systems

The DBPTK was used to successfully convert databases using different operating systems, specifically:

- Windows 8.1
- Windows 10
- Mac OS X
- Ubuntu 14.04
- Ubuntu 15.10

This was achieved by developing the software using the Java programming language, designed to execute on the Java Virtual Machine which can be run on most operating systems, and by not using any kind of operating system specific dependency.

Requirement 2: the software must be open-source, to make it easier to maintain and use in the future

The software is open-source, hosted on GitHub, and available[10] for anyone to download, change and contribute. It is licensed under the LGPL v3.0 license[11] which is a license used in Free/*Libre* and Open Source Software (FLOSS).

10 Available at `https://github.com/keeps/db-preservation-toolkit`
11 Full license available at `https://www.gnu.org/licenses/lgpl-3.0.en.html`.

Requirement 3: the software must be able to access the DBMS using an encrypted connection, to ensure the information security

The software was able to use an encrypted connection to MySQL and Microsoft SQL Server DBMSs using the HTTPS protocol, ensuring that the information could not be intercepted by third parties. The software did not behave differently while using an encrypted connection. (Fielding et al., 1999)

Requirement 4: it must be executable in remote environments that are accessible only via command line (i.e. without a graphical interface)

The software was developed as a command line tool, and there is currently no other way to execute the tool except using the command line. Since it is possible to access remote command line environments, via technologies like SSH, it is also possible to execute the DBPTK in those environments. (Ylonen and Lonvick, 2006)

Requirements covering the conversion between formats and systems

This section covers the following requirements:

Req. 5: Convert databases from the most popular[12] RDBMSs to SIARD 1 and SIARD 2 preservation formats;

Req. 6: Convert databases from multiple preservation formats back to a DBMS;

Req. 9: Convert structural information without data loss, performing (lossless) conversions when necessary.

Req. 10: Convert content information without data loss, performing (lossless) transformations when necessary.

Req. 11: Convert descriptive information associated with various database elements.

Req. 12: Convert behavioural information; saving it as descriptive information when it is unsupported by the target system.

Requirements 5 and 6 are proven to be achieved using the *roundtrip tests*, as they ensure the conversion as well as the correctness of the conversion to and from SIARD 1 and SIARD 2 preservation formats.

The following roundtrip tests were executed:

- PostgreSQL DBMS converted to SIARD 1 and back to PostgreSQL;

12 The most popular databases and which were chosen to be support as inputs and outputs is discussed in section 4.2.2.

- PostgreSQL DBMS converted to SIARD 2 and back to PostgreSQL;
- MySQL DBMS converted to SIARD 1 and back to MySQL;
- MySQL DBMS converted to SIARD 2 and back to MySQL;
- Microsoft SQL Server DBMS converted to SIARD 1 and back to Microsoft SQL Server;
- Microsoft SQL Server DBMS converted to SIARD 2 and back to Microsoft SQL Server;
- Oracle DBMS converted to SIARD 1 and back to Oracle;
- Oracle DBMS converted to SIARD 2 and back to Oracle;

Due to the lack of a Microsoft Access export module, the Microsoft Access module was used to convert databases in that format to SIARD 1 and SIARD 2, proceeding to the manual validation of the completeness and correction of this conversion.

The described tests ensured the conversion of content information as well as descriptive information, ensuring that no data nor documentation was lost. Regarding structural information, the *roundtrip test* is capable of expecting certain changes in specific types, making sure that only equivalent type changes are made, and that the remainder of the structure is correctly and completely migrated.

The behavioural information was manually verified to be in the SIARD files, since its conversion to a target DBMS is not assured.

Requirement 7: exporting to SIARD 2 must support specifying some tables that should be excluded from the conversion process

An additional export module was developed to achieve this requirement. It creates a list of the tables in the database and an option in the SIARD 2 export module to use that file to filter the exported tables. Combining these modules produces the following work flow:

1. Use DBPTK with any import module and the "table list" export module. This creates a text file listing all tables in the database;
2. Manually edit the file with a text editor, making sure to remove the names of the tables that should not be migrated;
3. Use DBPTK with the same import module as before, and the SIARD 2 export module, configuring it to use the table list.

After these steps, the result is a SIARD 2 file containing only the tables that were listed in the table list file.

Requirement 8: support adding more input and output systems and formats

Since the future use of DBPTK may require new modules to be developed for other DBMSs, it should include some mechanisms to simplify that process and attempt to simplify the process of developing a new module.

The application uses the modular approach, establishing the order in which the modules interact and the internal data structure used to transfer data from the import to the export module. By complying with this design, adding a new import module is a matter of retrieving the data from the database format or system, saving it to the appropriate internal structure objects and send them in the expected order to the export module; and adding a new export module is a matter of expecting to receive the objects from the import module in a specific order and outputting them to the target format or system. Also, there is a contract in the form of a couple of interfaces that modules must inherit.

As long as the databases support the relational database model, the creation of new modules requires minimal changes in the main application part, requiring only the creation of code to interact with the database and the internal data structure used by the application.

Requirement 9: be capable of handling databases containing millions of rows and hundreds of Gigabytes in size

At KEEP SOLUTIONS, the DBPTK was successfully used to migrate their biggest client database (containing more than 100 Gigabytes and more than a billion rows) to SIARD 2. This conversion took approximately 80 minutes on an Intel Core i7 machine with 16 gigabytes of RAM (although the application was limited to using 2 gigabytes) and approximately 500 gigabytes of available memory in an Hard Disk Drive.

Requirement 14: report and document any possible data loss, the report must include all the information that was changed during the conversion.

Any changes to the data types and problems during the conversion (that may have caused some data loss) are documented in a report file that is generated by DBPTK for every database migration.

This section concludes the description of the Database Preservation Toolkit artefact. The next section will cover the Database Visualization Toolkit, followed by the *Validation* chapter, which focuses on validating the tools using the E-ARK pilots.

4.3 DATABASE VISUALIZATION TOOLKIT

This section describes the Database Visualization Toolkit (DBVTK) artefact, its purpose and requirements, its design and development phases, and how the tool was tested. The validation phase is described in chapter 5.

4.3.1 *Purpose and Requirements*

The preservation process is only successful if there is a way for the designated community to access the archived objects, and this is true for databases as well. Towards this objective, the Database Visualization Toolkit was developed, to provide access to databases, allowing archivists and consumers to preview, explore and retrieve information from preserved databases. The software is aimed at end-users with little or no experience with SQL, and its main goal is to enable non-technical users to quickly find a set of records and provide means to export and print these records.

As with DBPTK, a few requirements were laid out before the development phase whilst others were added after hearing user feedback, they are listed below.

In these requirements, "the software" refers to the DBVTK as a whole, and the "web interface" is the presentation layer accessible through a web browser. Although some "web interface" functionality may need special support from "the software", that implication is omitted in the requirements, as the "web interface" requirements try to avoid making assumptions about implementation details.

Software Requirements

Req. 1: The software must be a Web Application accessible both in an Intranet environment and over the Internet, via a web interface.

Req. 2: The application's data layer must be scalable, supporting millions of records and quick searches on those records.

Req. 3: Support database representations based on SIARD 2 format.

Req. 4: It must not include technologies protected by intellectual rights laws, as they could hinder deploying the software in a user's environment.

Web Interface Requirements

Req. 5: Show the database metadata (structural, behavioural and descriptive) information present in SIARD 2.

Req. 6: Show the list of databases when first accessed, allowing the user to click a database in the list to browse it.

Req. 7: Support searching the whole database and individual tables.

Req. 8: Support filtering by values of specific columns when searching individual tables.

Req. 9: Support sorting search results using the values of a column.

Req. 10: Support saving searches and make those searches accessible via a specific URI.

Req. 11: Support downloading search results as a CSV file.

Req. 12: Have a page dedicated to display a single record and provide links to related records via foreign key relations.

Req. 13: Support downloading LOBs present in the SIARD file.

Req. 14: Display a diagram that can help identifying the most important tables in a database schema.

4.3.2 *Design and Development*

The Database Visualization Toolkit is designed to be a scalable web-service that is able to serve multiple archived databases. It is optimized in order to provide almost instantaneous responses to searches on millions of records. It uses a client-server approach to provide access to databases. Its three main components are illustrated in figure 16:

- The web interface, with which the user interacts to access the database;
- The server-side application, providing a business logic layer between the web interface and the data layer;
- The data layer, where database information is stored and indexed.

It would be difficult to build a scalable data layer using RDBMS technologies (e.g. MySQL), because they are not capable of handling tens of databases containing millions of records each. This problem was experienced by users of the Sofia software, as the system would slow down significantly when multiple databases had been loaded, taking a few minutes to execute a query. After analysing potential candidates, Apache Solr[13] emerged as the chosen data layer platform. Apache Solr is an open source enterprise search platform. It was chosen for its versatility, scalability, and ability to provide almost instantaneous responses to searching and filtering queries on millions of records. Solr can also function in

13 Apache Solr is available at `http://lucene.apache.org/solr/`.

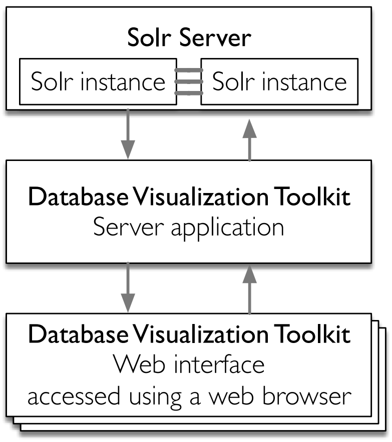

Solr Server
Solr instance
Solr instance
Database Visualization Toolkit
Server application
Database Visualization Toolkit
Web interface
accessed using a web browser

Figure 16.: DBVTK application architecture overview

a distributed manner, as a cluster of Solr servers working together to improve the whole system's performance, this is called a Solr Cloud system.

The Solr platform is used to index preserved database records and provide searching and filtering functionality on those records.

In Solr, searchable data is stored into collections, which are data indexes that are optimized to provide fast searching functionality. To save databases into Solr, there are two main types of collections used: one for databases, containing database metadata; and one for tables, containing table data. Information in a collection is stored in documents, which are groups of key-value pairs. To access a database in Solr, the databases collection is searched, obtaining the database metadata and information about its tables (including information needed to identify the collections containing their data). When searching a specific table, the database information is used to identify the corresponding collection to search, Solr is used to search that collection and then the search results are shown in the web interface.

To handle the migration of databases in SIARD 2 format to Solr, the most fitting strategy is to develop a DBPTK Solr export module and use that module to load databases into the Solr server. The DBVTK is then responsible to retrieve and render the information in the web interface.

Since both the DBVTK and the Solr export module interact with Solr, both software pieces were designed to use a common component to add and retrieve information from Solr. This makes the code more maintainable, since changes to the way information is stored in Solr would automatically be picked up by both the DBPTK Solr export module and the DBVTK.

4.3.3 *Deployment*

To deploy the Database Visualization Toolkit, a ZIP package was created with everything needed to execute the tool:

- Interactive command line scripts, for Linux and Windows, to easily start Solr and DBVTK;
- The **Solr server**, pre-configured to be used by DBVTK;
- The **Tomcat server** pre-loaded with DBVTK, to serve the DBVTK as a Web Application;
- The **Database Preservation Toolkit**, to add databases to the DBVTK.

After downloading this package, the user simply needs to execute the `start.sh` or `start.bat` script (depending on the operating system) to start the system, and execute the `stop.sh` or `stop.bat` to shut it down. Some instructions are provided when the script is executed, like how to add new databases and how to access the web interface.

4.3.4 *Testing*

Some unit tests were developed, to ensure that covered methods would perform as expected. The tests are executed every time the application is compiled, ensuring that any change that breaks existing functionality is noticed.

4.3.5 *Validation*

To display the required information, all software components must support and be able to carry the information over to the next component, specifically:

- During the conversion from the DBMS to SIARD, in the pre-ingest phase:
  1. The DBPTK DBMS import module must be able to retrieve the information from the DBMS;
  2. The DBPTK SIARD export module must be able to output the information to the SIARD file.
- During the conversion from SIARD to Solr, in the access phase:
  3. The DBPTK SIARD import module must be able to retrieve the information from the SIARD;

4. The DBPTK Solr export module must be able to add the information to the Solr server.

- When the consumer is accessing the information, in the access phase:

5. The DBVTK server side component must be able to retrieve the information from the Solr server;
6. The DBVTK web interface component must be able to show the information in the required manner.

This section details how each requirement was validated, i.e. how it was proven that the artefact achieves the requirements, sometimes mentioning some of the above software components.

Requirement 1: The software must be a Web Application accessible both in an Intranet environment and over the Internet, via a web interface

The software was designed as a web application accessible via a web interface. With the appropriate network configuration, it can be accessed in an Intranet environment, over the Internet, or both.

Requirement 2: The application's data layer must be scalable, supporting millions of records and quick searches on those records

The DBVTK data layer is Solr, which is a scalable and very fast search engine. Solr can also scale horizontally, distributing the searching functionality amongst a group of Solr servers working as a single instance, to provide faster response times with very large databases.

Requirement 3: Support database representations based on SIARD 2 format

The DBPTK Solr export module is used to add databases to Solr, this way any system or format supported by a DBPTK import module can be migrated to Solr. However, only the conversion from SIARD 2 to Solr was tested.

Requirement 4: It must not include technologies protected by intellectual rights laws, as they could hinder deploying the software in a user's environment

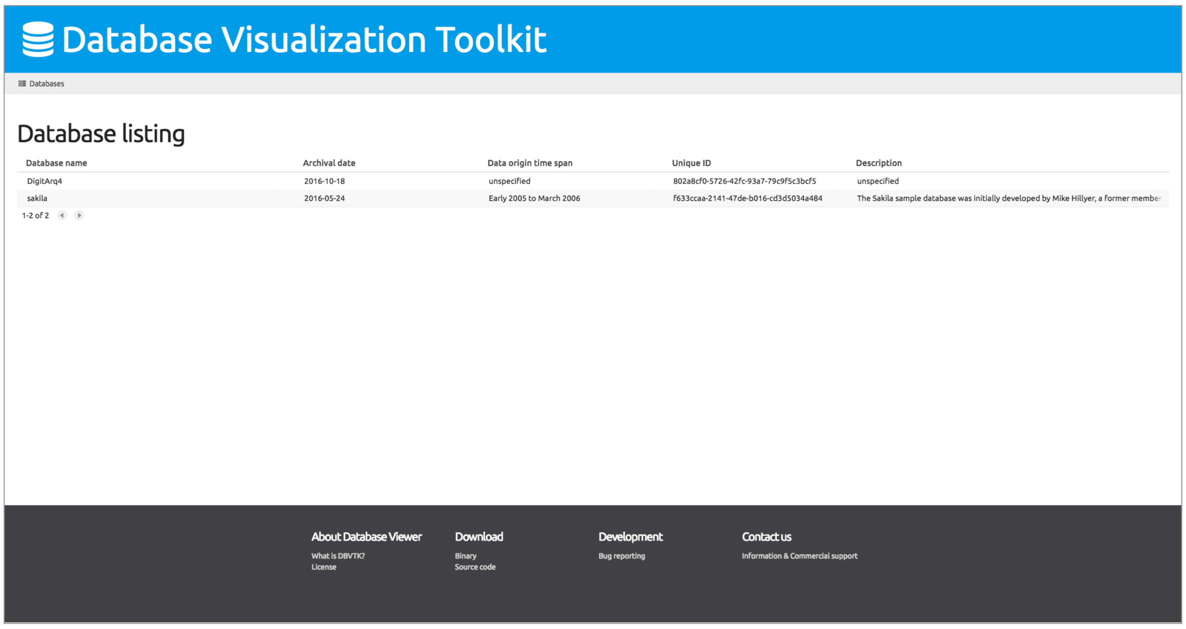
The solution is implemented using the Java programming language and is designed to run on the Java Runtime Environment, neither of which impose any license restrictions on created software. The software depends on Solr and Google Web Toolkit (GWT), which are both distributed under the Apache Version 2.0 License[14], and also on DBPTK, which is

14 Full license available at `https://www.apache.org/licenses/LICENSE-2.0`.

distributed under the LGPL v3.0 license[15]. Both of these licensing models are preferred by E-ARK for software developed in the project and do not hinder deploying the tools. The DBVTK artefact needs to be executed in a web server, like Apache Tomcat, that effectively loads DBVTK and provides access to it over the network. Apache Tomcat is also distributed under the Apache Version 2.0 License.

The DBVTK, similarly to the DBPTK, is also licensed under the LGPL v3.0 license.

Requirements 5 and 6: Show the database metadata (structural, behavioural and descriptive) information present in SIARD 2; and show the list of databases when first accessed, allowing the user to click a database in the list to browse it



Database listing

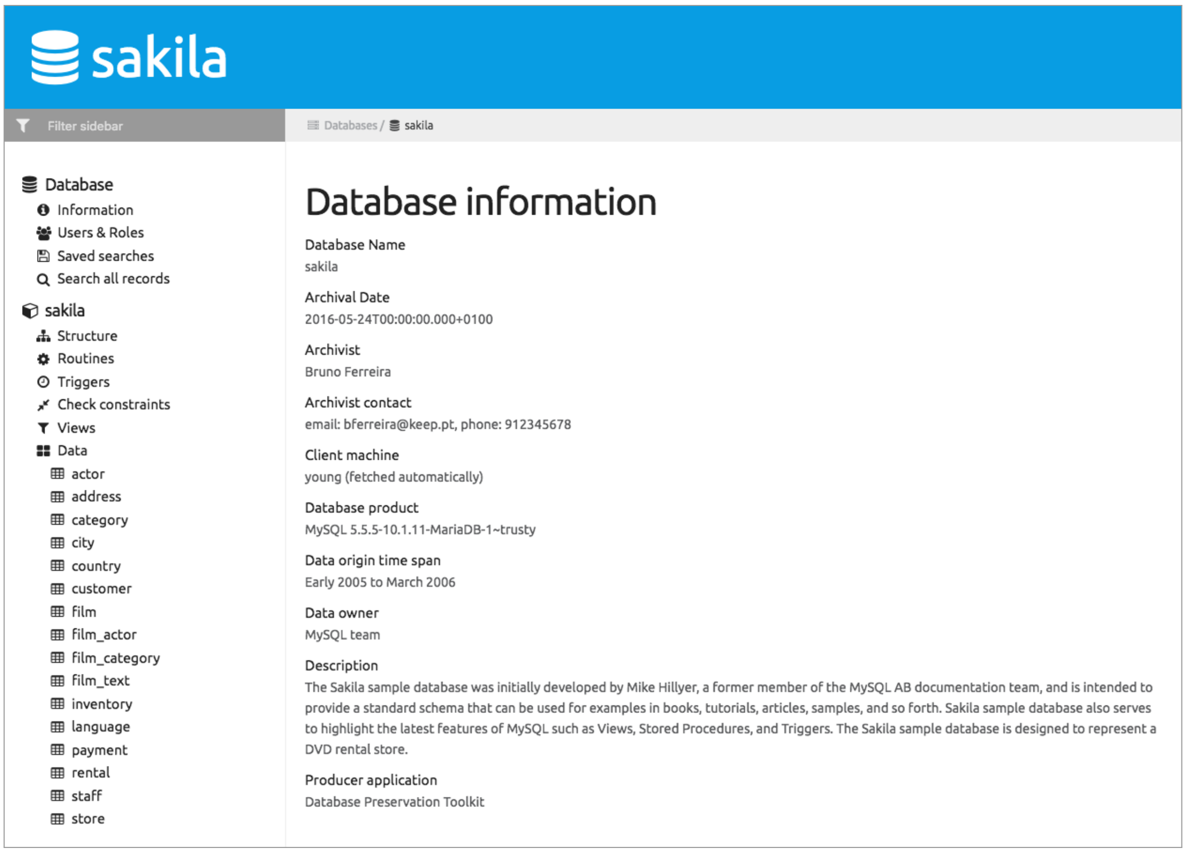
Database name	Archival date	Data origin time span	Unique ID	Description
DigitArq4	2016-10-18	unspecified	802a8cf0-5726-42fc-93a7-79c9f5c3bcf5	unspecified
sakila	2016-05-24	Early 2005 to March 2006	f633ccaa-2141-47de-b016-cd3d5034a484	The Sakila sample database was initially developed by Mike Hillyer, a former member

1-2 of 2

Figure 17.: Database listing (with two databases)

Figure 17 shows the list of databases that are currently loaded in the system and ready to be accessed. Clicking an item in that list shows descriptive metadata about the whole database, while a sidebar appears on the left side of the page with hyperlinks that allow the consumer to access information about other database elements. The hyperlinks direct the user to pages where metadata for different database elements is shown, because showing all the database metadata in a single web page would make the page bulky and difficult to read. Some examples of pages displaying database metadata are shown in figures 18, 19 and 20.

15 Full license available at `https://www.gnu.org/licenses/lgpl-3.0.en.html`.



Database information

Database Name
sakila

Archival Date
2016-05-24T00:00:00.000+0100

Archivist
Bruno Ferreira

Archivist contact
email: bferreira@keep.pt, phone: 912345678

Client machine
young (fetched automatically)

Database product
MySQL 5.5.5-10.1.11-MariaDB-1~trusty

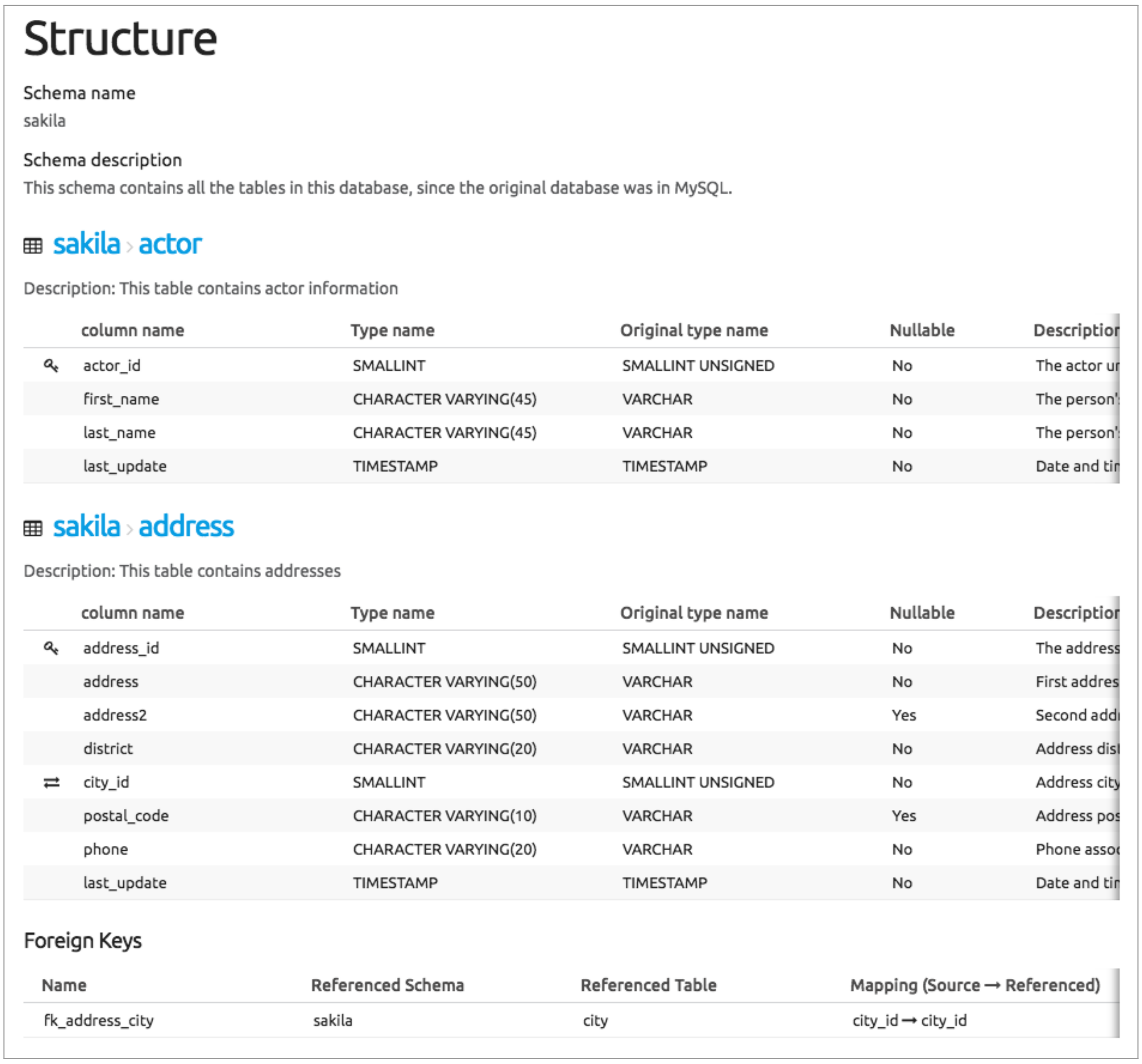
Data origin time span
Early 2005 to March 2006

Data owner
MySQL team

Description
The Sakila sample database was initially developed by Mike Hillyer, a former member of the MySQL AB documentation team, and is intended to provide a standard schema that can be used for examples in books, tutorials, articles, samples, and so forth. Sakila sample database also serves to highlight the latest features of MySQL such as Views, Stored Procedures, and Triggers. The Sakila sample database is designed to represent a DVD rental store.

Producer application
Database Preservation Toolkit

Figure 18.: Database descriptive metadata



Structure

Schema name
sakila

Schema description
This schema contains all the tables in this database, since the original database was in MySQL.

sakila › actor

Description: This table contains actor information

	column name	Type name	Original type name	Nullable	Descriptio
🔑	actor_id	SMALLINT	SMALLINT UNSIGNED	No	The actor u
	first_name	CHARACTER VARYING(45)	VARCHAR	No	The person'
	last_name	CHARACTER VARYING(45)	VARCHAR	No	The person'
	last_update	TIMESTAMP	TIMESTAMP	No	Date and ti

sakila › address

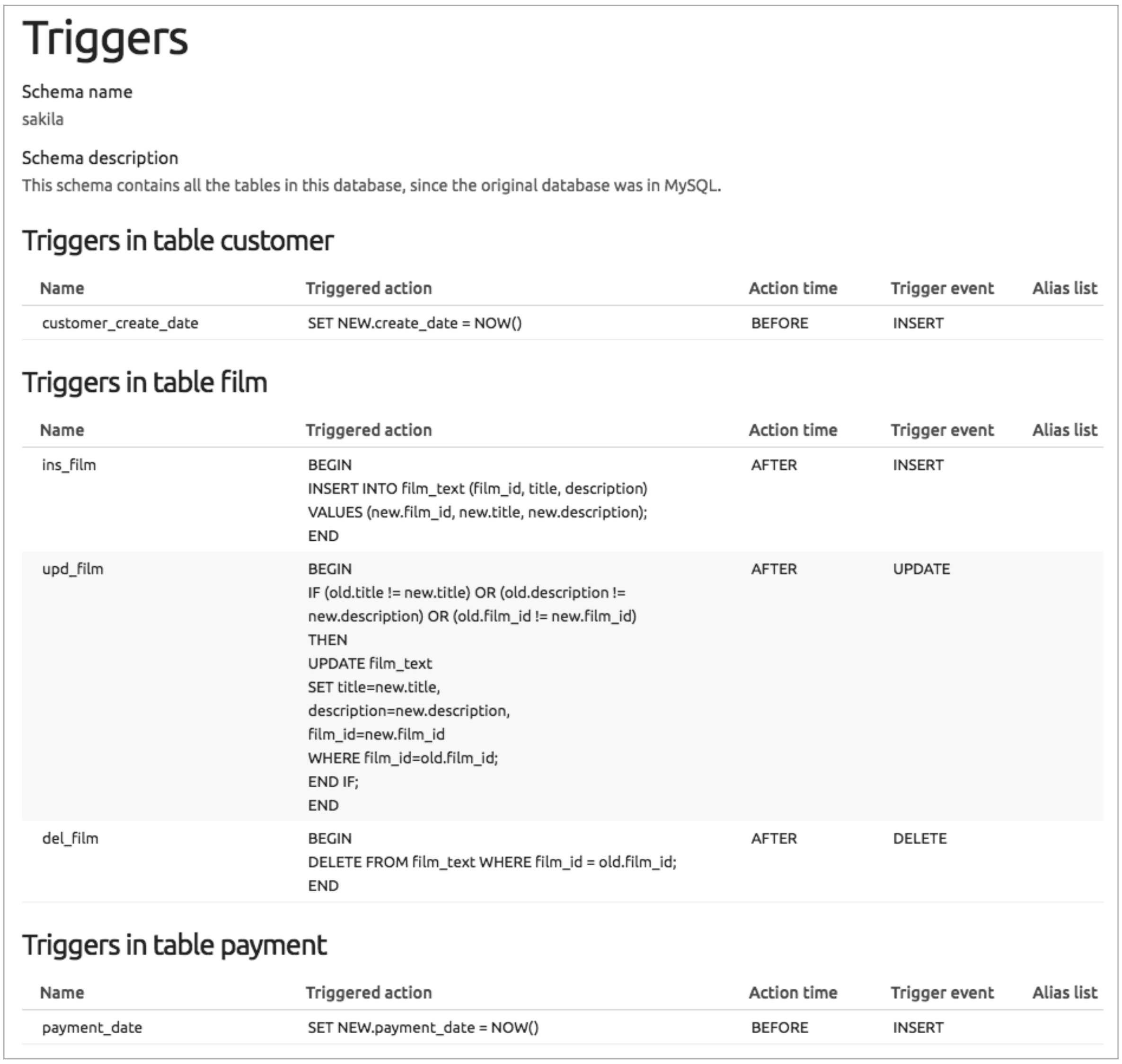
Description: This table contains addresses

	column name	Type name	Original type name	Nullable	Descriptio
🔑	address_id	SMALLINT	SMALLINT UNSIGNED	No	The address
	address	CHARACTER VARYING(50)	VARCHAR	No	First addres
	address2	CHARACTER VARYING(50)	VARCHAR	Yes	Second add
	district	CHARACTER VARYING(20)	VARCHAR	No	Address dis
⇄	city_id	SMALLINT	SMALLINT UNSIGNED	No	Address city
	postal_code	CHARACTER VARYING(10)	VARCHAR	Yes	Address pos
	phone	CHARACTER VARYING(20)	VARCHAR	No	Phone asso
	last_update	TIMESTAMP	TIMESTAMP	No	Date and ti

Foreign Keys

Name	Referenced Schema	Referenced Table	Mapping (Source → Referenced)
fk_address_city	sakila	city	city_id → city_id

Figure 19.: Database structural metadata



Triggers

Schema name
sakila

Schema description
This schema contains all the tables in this database, since the original database was in MySQL.

Triggers in table customer

Name	Triggered action	Action time	Trigger event	Alias list
customer_create_date	SET NEW.create_date = NOW()	BEFORE	INSERT	

Triggers in table film

Name	Triggered action	Action time	Trigger event	Alias list

ins_film	BEGIN
	INSERT INTO film_text (film_id, title, description)
	VALUES (new.film_id, new.title, new.description);
	END
		AFTER	INSERT	

upd_film	BEGIN
	IF (old.title != new.title) OR (old.description !=
	new.description) OR (old.film_id != new.film_id)
	THEN
	UPDATE film_text
	SET title=new.title,
	description=new.description,
	film_id=new.film_id
	WHERE film_id=old.film_id;
	END IF;
	END
		AFTER	UPDATE	

del_film	BEGIN
	DELETE FROM film_text WHERE film_id = old.film_id;
	END
		AFTER	DELETE	

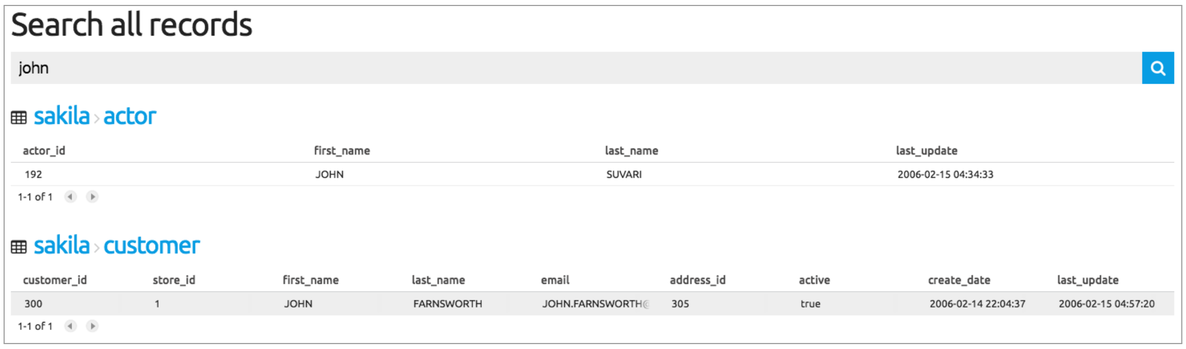
Triggers in table payment

Name	Triggered action	Action time	Trigger event	Alias list
payment_date	SET NEW.payment_date = NOW()	BEFORE	INSERT	

Figure 20.: Database behavioural metadata

Requirement 7: Support searching the whole database and individual tables

Using the searching and filtering capabilities provided by Solr, and loading the database information in a way that makes the most of those capabilities, the DBVTK allows a consumer to search the database. The consumer can search the whole database, and doing so will show records grouped according to the table in which they appear (illustrated by figure 21 or search a specific table (depicted in figure 22).



Search all records

john

sakila › actor

actor_id	first_name	last_name	last_update
192	JOHN	SUVARI	2006-02-15 04:34:33

1-1 of 1

sakila › customer

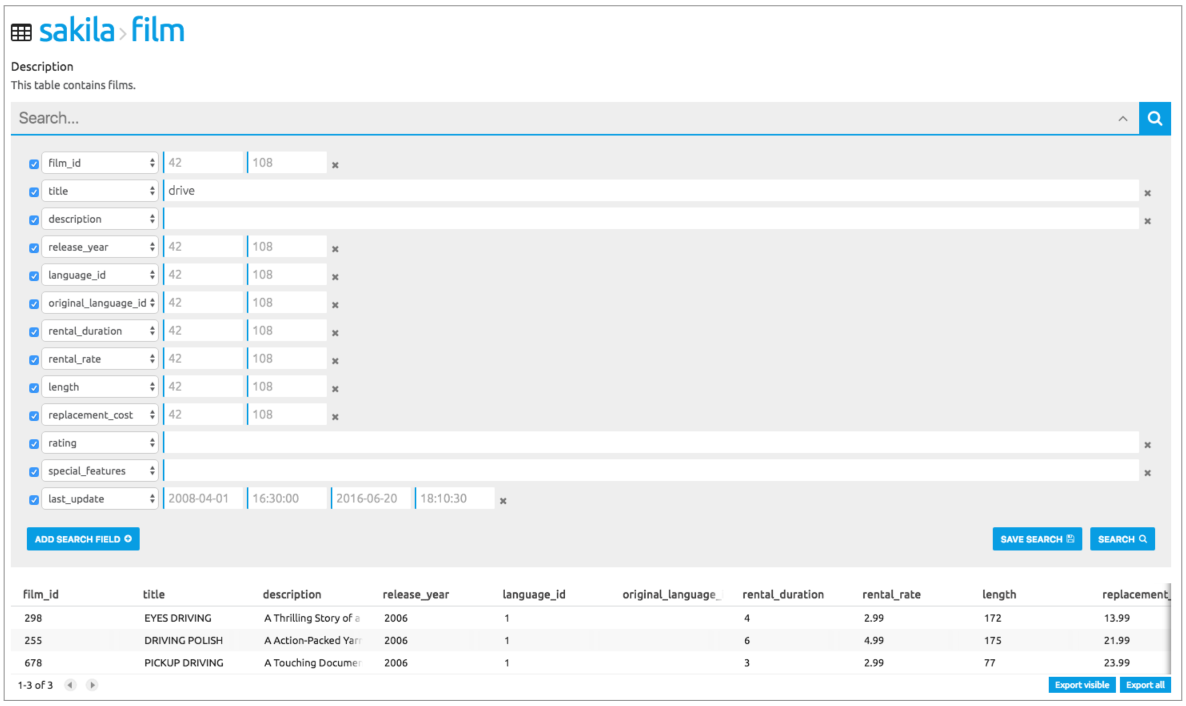
customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
300	1	JOHN	FARNSWORTH	JOHN.FARNSWORTH@	305	true	2006-02-14 22:04:37	2006-02-15 04:57:20

1-1 of 1

Figure 21.: Searching all records in a database

Requirement 8: Support filtering by values of specific columns when searching individual tables

The web interface also supports searching values in specific columns, either by providing an exact value or by defining a range of accepted values, as depicted in figure 22. This functionality is referred to as the advanced search.



sakila › film

Description
This table contains films.

Search...

✓ film_id	42	108		
✓ title	drive			
✓ description				
✓ release_year	42	108		
✓ language_id	42	108		
✓ original_language_id	42	108		
✓ rental_duration	42	108		
✓ rental_rate	42	108		
✓ length	42	108		
✓ replacement_cost	42	108		
✓ rating				
✓ special_features				
✓ last_update	2008-04-01	16:30:00	2016-06-20	18:10:30

film_id	title	description	release_year	language_id	original_language_	rental_duration	rental_rate	length	replacement_
298	EYES DRIVING	A Thrilling Story of a	2006	1		4	2.99	172	13.99
255	DRIVING POLISH	A Action-Packed Yar	2006	1		6	4.99	175	21.99
678	PICKUP DRIVING	A Touching Documer	2006	1		3	2.99	77	23.99

1-3 of 3

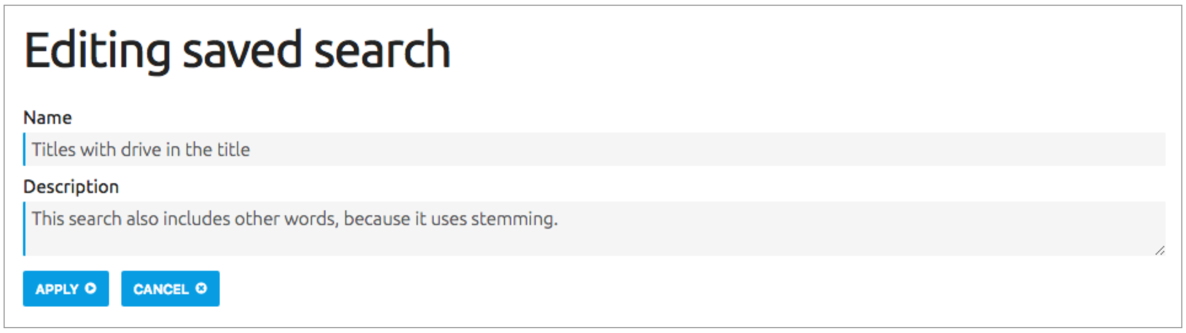
Figure 22.: Searching all record in a table

Requirement 9: Support sorting search results using the values of a column

The tables shown in figures 21 and 22 also support sorting the results using the values of a column. Clicking the column header once sorts the values in ascending order and clicking it again sorts the values in descending order.

Requirement 10: Support saving searches and make those searches accessible via a specific URI

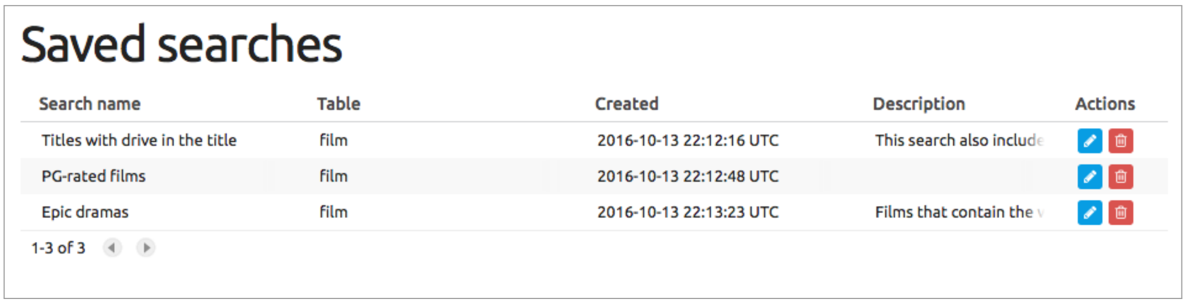
The advanced search, shown in figure 22, includes a button to save the search for later re-use. Upon saving the search, the user is shown a form to add a name and a description to the saved search. Upon submitting a name and description, the query will be displayed in a list containing the saved queries for the current database. Clicking an item on this list will open the URI that identifies this saved search, display the page to search a specific table and execute the search. This page shows the search results along with the original search parameters, allowing the consumer to change some of the search parameters and search again. This functionality is depicted in figures 23 and 24.



Editing saved search

Name	Titles with drive in the title
Description	This search also includes other words, because it uses stemming.

Figure 23.: Adding a name and description to a saved search



Saved searches

Search name	Table	Created	Description	Actions
Titles with drive in the title	film	2016-10-13 22:12:16 UTC	This search also include	
PG-rated films	film	2016-10-13 22:12:48 UTC		
Epic dramas	film	2016-10-13 22:13:23 UTC	Films that contain the	

1-3 of 3

Figure 24.: Saved searches listing for the database

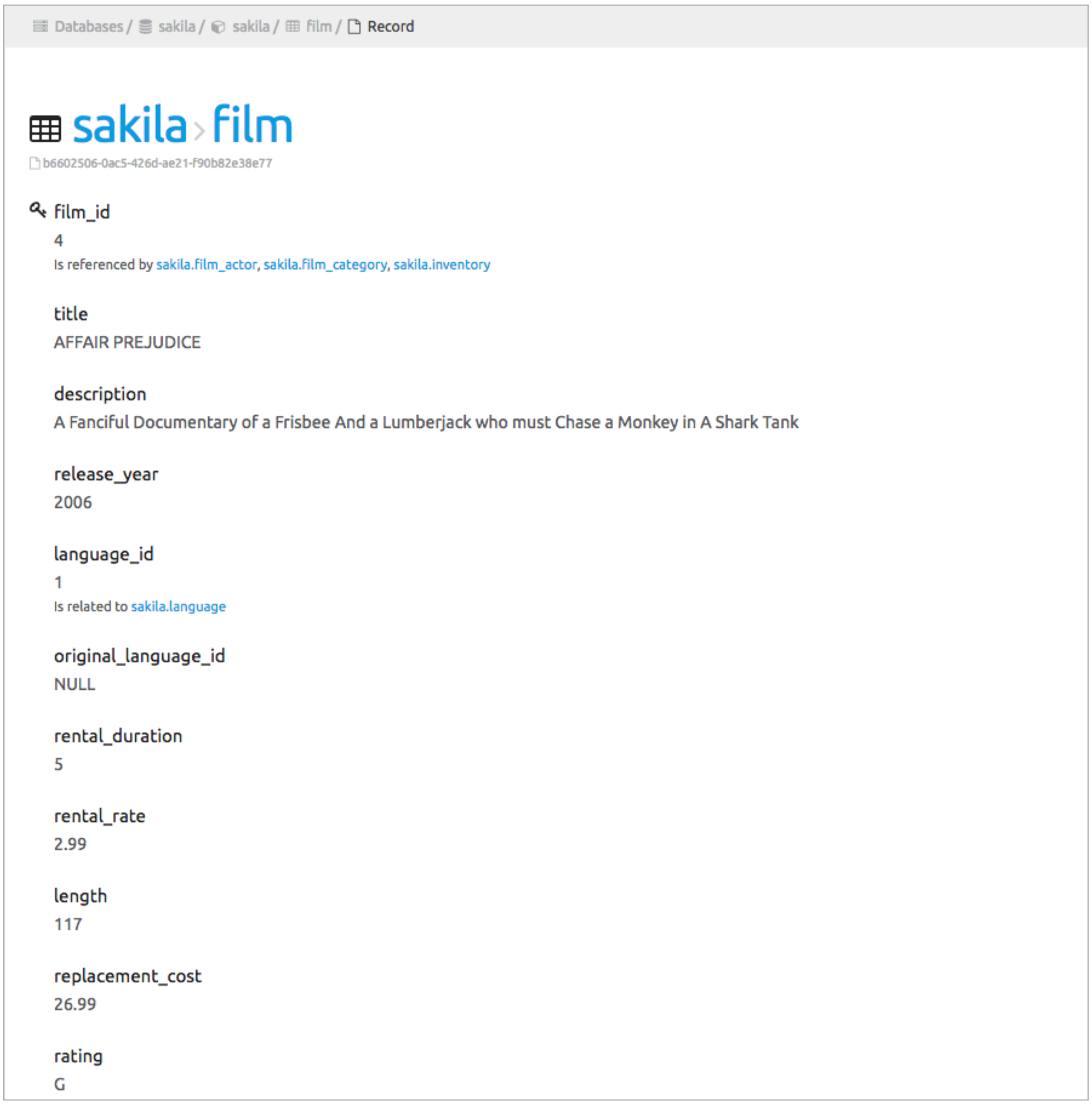
Requirement 11: Support downloading search results as a CSV file

Below the search results, in figure 22, there are two buttons that provide the functionality to export the search results to CSV. There are two buttons because the search results are paginated, which means that only a subset of the results are shown and a button must be clicked to advance to the "next page" and view the next subset of results. One of the buttons exports the currently visible result subset to CSV and the other exports all the search results.

The search results pagination avoids overloading the server, by ensuring that just a few records are loaded at a time. Even when sending the CSV export, data is obtained and sent in chunks, to avoid loading all the records at once.

Requirement 12: Have a page dedicated to display a single record and provide links to related records via foreign key relations

The single record page displays information in a list, to effectively show cell contents of various lengths. If the cell is related to other cells via a foreign key, by pointing to other records or by having other records point to it, a link appears below the cell contents that allows the navigation to the related records via a specific foreign key. Clicking this link directs the user to the list of related records, or to a single record if there is only one related record. This functionality is depicted in figures 25 and 26. Figure 26 was obtained by clicking the relation in the column "`film_id`" to the table "`sakila.inventory`", this performed a search for a specific value in the table "`sakila.inventory`" (shown in the figure).


Databases / sakila / sakila / film / Record

sakila > film

b6602506-0ac5-426d-ae21-f90b82e38e77

film_id
4
Is referenced by sakila.film_actor, sakila.film_category, sakila.inventory

title
AFFAIR PREJUDICE

description
A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank

release_year
2006

language_id
1
Is related to sakila.language

original_language_id
NULL

rental_duration
5

rental_rate
2.99

length
117

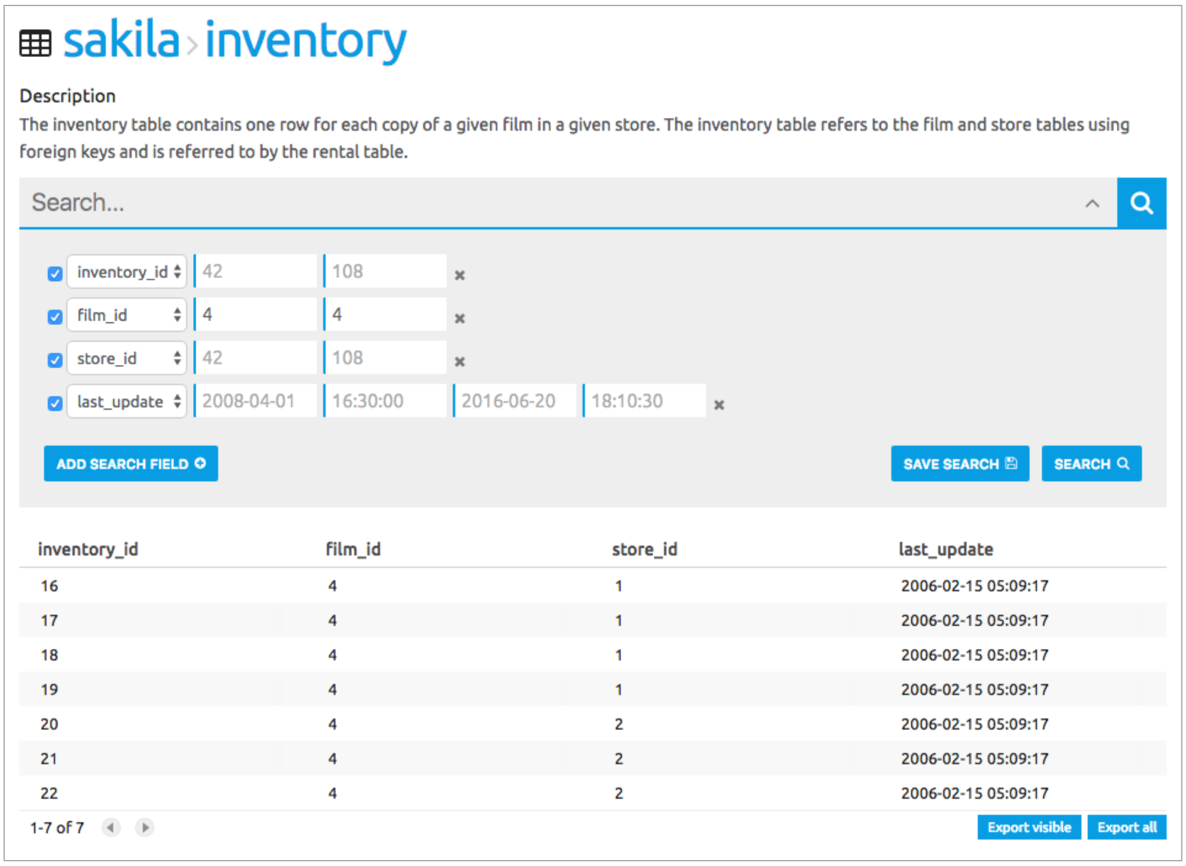
replacement_cost
26.99

rating
G

Figure 25.: Displaying a single record with foreign key relations

Requirement 13: Support downloading LOBs present in the SIARD file

In the DBVTK web interface, the LOB cells are displayed as a link that the consumer can click to download the corresponding LOB file. This is illustrated in figure 27, where the LOB present in column "`picture`" has been downloaded by clicking the "Download LOB" link.



Description

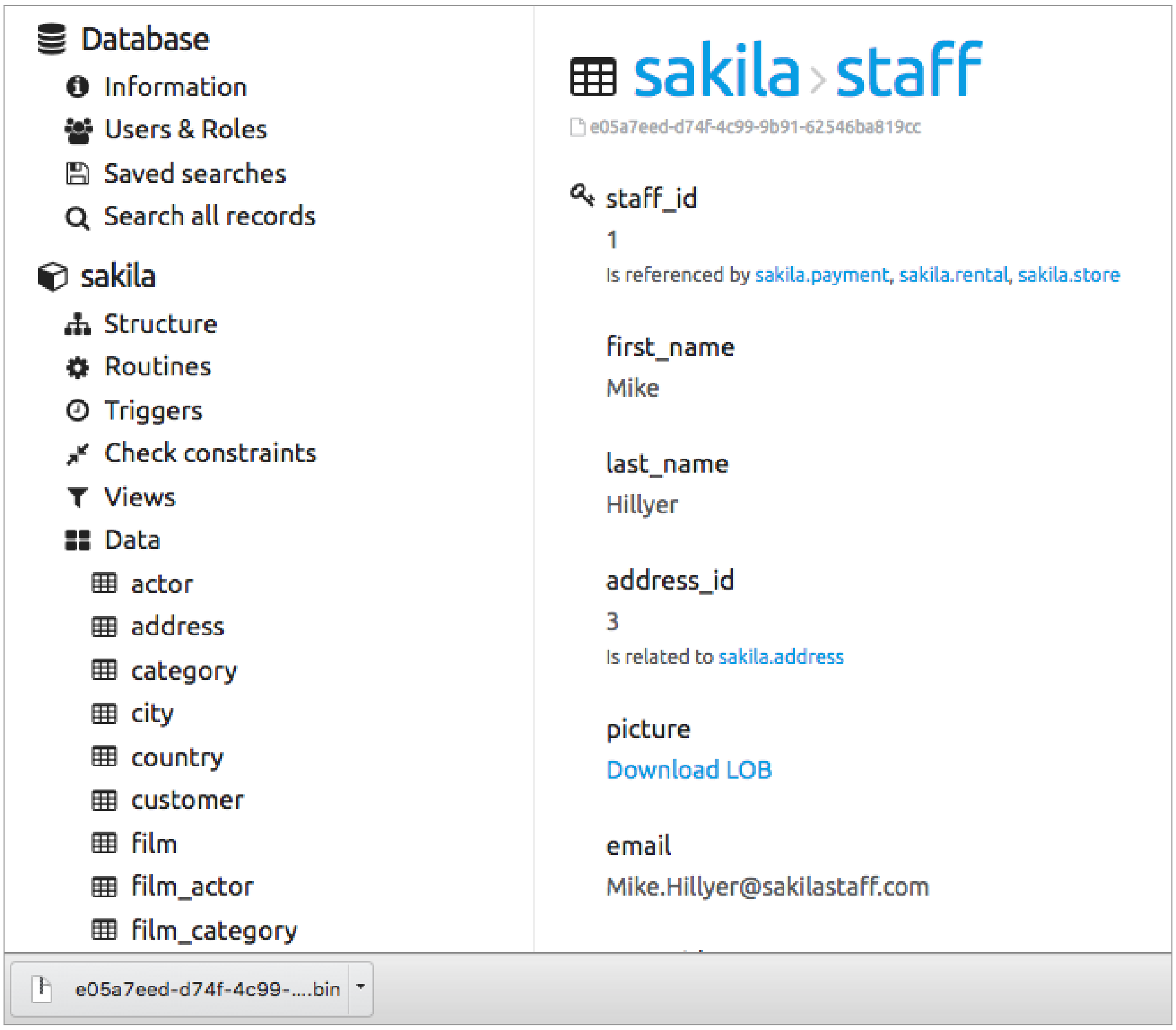
The inventory table contains one row for each copy of a given film in a given store. The inventory table refers to the film and store tables using foreign keys and is referred to by the rental table.

✓ inventory_id	42	108		
✓ film_id	4	4		
✓ store_id	42	108		
✓ last_update	2008-04-01	16:30:00	2016-06-20	18:10:30

inventory_id	film_id	store_id	last_update
16	4	1	2006-02-15 05:09:17
17	4	1	2006-02-15 05:09:17
18	4	1	2006-02-15 05:09:17
19	4	1	2006-02-15 05:09:17
20	4	2	2006-02-15 05:09:17
21	4	2	2006-02-15 05:09:17
22	4	2	2006-02-15 05:09:17

1-7 of 7

Figure 26.: Navigating to related records



sakila > staff

e05a7eed-d74f-4c99-9b91-62546ba819cc

staff_id
1
Is referenced by sakila.payment, sakila.rental, sakila.store

first_name
Mike

last_name
Hillyer

address_id
3
Is related to sakila.address

picture
Download LOB

email
Mike.Hillyer@sakilastaff.com

Figure 27.: Downloading a LOB

Requirement 14: Display a diagram that can help identifying the most important tables in a database schema.

Some databases that are submitted to an archive already include a data dictionary and an entity-relationship diagram, but it may not be explicitly stated in those documents which tables are most the important in a database. If those documents are not present, the DBVTK should have a way to simplify the identification of the most important tables.

The diagram dynamically generated by the application draws each table as a circle, and connects circles with arrows, representing the foreign key relations. To simplify identifying the most important tables, two highlighting methods are used: colour variation and size variation. Tables containing more rows and columns are represented by larger circles, and tables with more relations are coloured darker. As an example, if the diagram contains a very big and dark circle, it probably corresponds to the main table in the database.

The highlighting methods adapt to the database information, mapping the provided database values (number of rows, columns and foreign key relations) to specific sizes and colours. To determine the colour, the number of foreign key relations for each table is obtained (both the relations to other tables and from other tables), and then the table with the most relations is coloured darkest, the table with the lowest number of relations is coloured lightest, and the other tables' colours are linearly distributed between the darkest and lightest colour according to their number of relations. To determine the size, the number of rows and columns for each table is obtained and a score is determined using the formula $N_{score} = N_{rows} * N_{columns}$, the score is then linearly mapped to a value between 0 and 2, modelled according to the inverse sine function: $size(x) = sin^{-1}(x-1) + \frac{\pi}{2}$ and then N_{size} is linearly mapped to the actual geometric size. The function plot is shown in figure 28 and shows the function domain, $[0, 2]$, and codomain, $[0, 3]$. The values are modelled through this function to increase the size difference of the biggest and smallest tables in comparison to other tables, making them easier to identify.

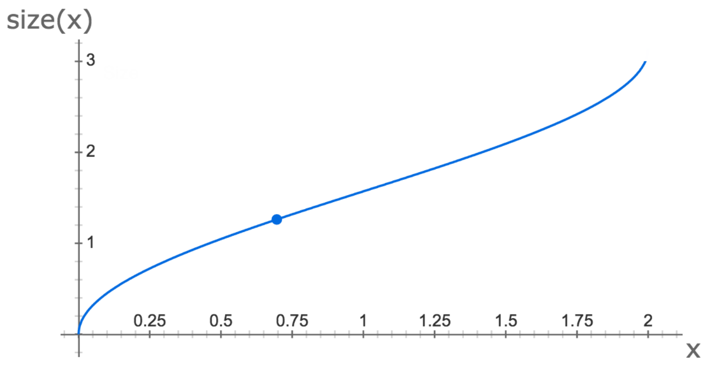


Figure 28.: The $size(x)$ function plot

Using example in figure 29, it is possible to identify that the tables "rental", "payment", "film", "staff" and "store" are the most important, and since the database has meaningful table names (i.e. they are not encoded) it is possible to guess that the database is about "a film rental store", which is indeed what the database is about.

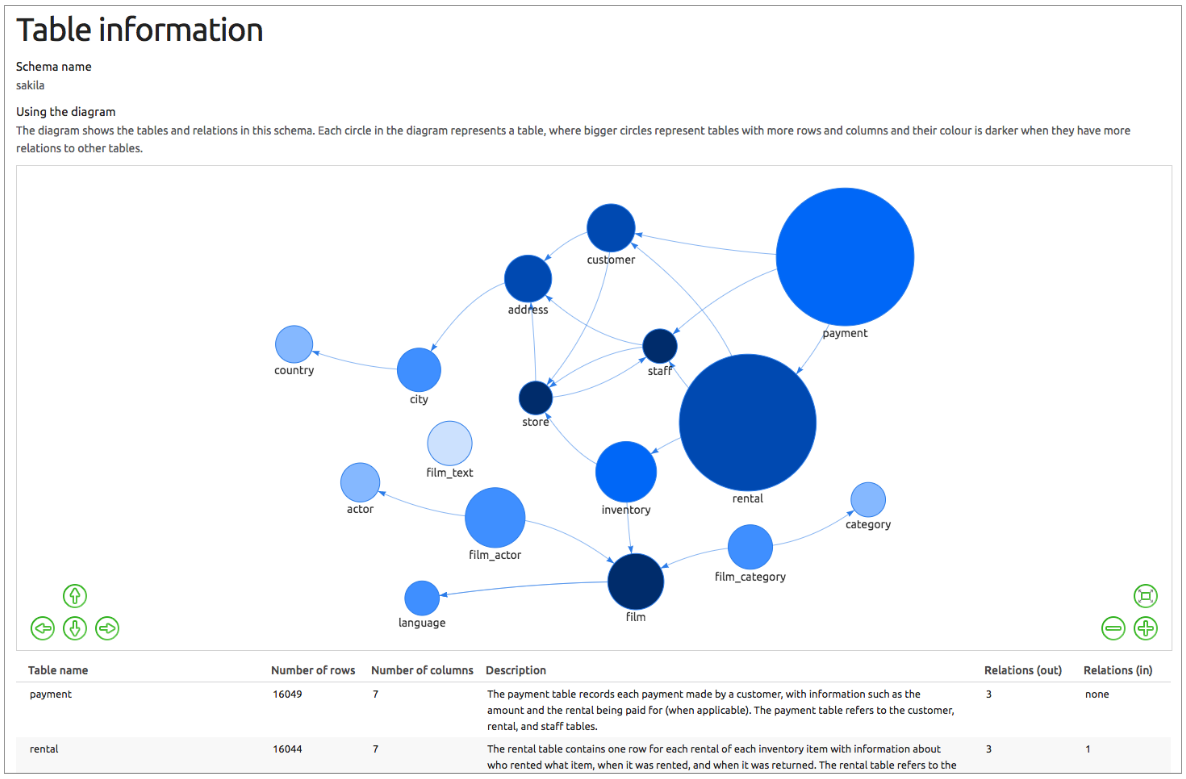


Table information

Schema name

sakila

Using the diagram

The diagram shows the tables and relations in this schema. Each circle in the diagram represents a table, where bigger circles represent tables with more rows and columns and their colour is darker when they have more relations to other tables.

Table name	Number of rows	Number of columns	Description	Relations (out)	Relations (in)
payment	16049	7	The payment table records each payment made by a customer, with information such as the amount and the rental being paid for (when applicable). The payment table refers to the customer, rental, and staff tables.	3	none
rental	16044	7	The rental table contains one row for each rental of each inventory item with information about who rented what item, when it was rented, and when it was returned. The rental table refers to the	3	1

Figure 29.: Web Interface: diagram that highlights important tables

This section concludes the description of the Database Visualization Toolkit artefact. The next chapter focuses on validating the tools using the E-ARK pilots.

5 VALIDATION

This chapter details the process used to validate the tools and the validation's outcome.

5.1 METHODOLOGY

The developed tools (the Database Preservation Toolkit and the Database Visualization Toolkit) have been developed in the context of the E-ARK project, and as such, are subject to the validation via piloting.

Several institutions are in the process of piloting these tools. The piloting of the tools consists in using the tools with real databases that they need to handle, possibly even databases that cause problems when handled by other tools. With this kind of "real-world" testing, the pilots help to improve the overall quality of the produced software.

The pilots have a few goals defined, e.g. migrate a database with a million of records to SIARD 2 using DBPTK, and try to use the tools to achieve that goal. If they run into problems, they should report them, wait for them to be fixed and try to complete the pilot again with the latest software version.

The problems were reported on the projects' GitHub issue tracker[1] and were given higher priority when they were impeding the successful conclusion of a pilot. Most of the pilots tested client's complex databases, that could not be distributed or accessible outside the archives, therefore the logging produced by the application had to be very detailed while obfuscating as much of the database metadata as possible. Using the logging and gathering more information about the environment and the database, the problem could be identified and fixed, and a new (experimental) version could be developed and tested by the pilot. If the new version succeeded in completing the task, the corrections would be added to the software and be released in a new official software version.

1 The issue tracker for DBPTK is available at `https://github.com/keeps/db-preservation-toolkit/issues` and the issue tracker for DBVTK is available at `https://github.com/keeps/db-visualization-toolkit/issues`.

5.2 RESULTS

The following sections describe the official pilots and their results. Alföldi and Réthy (2016)

Danish National Archives

The Danish National Archives (DNA) pilot attempted to use DBPTK to make four successful data extractions from live databases into the SIARD 2 format:

- Database from the Health System of the Danish Serum Institute, containing information from reported infectious diseases at the national level, and infectious diseases for all Danish citizens. This is a Microsoft SQL Server database containing 90 tables and around 500,000 records;
- Database from a private sector business, Kultunaut Aps., containing information about cultural events at the national level, from the smallest local gatherings to international exhibitions and events in Denmark. This is a MySQL database containing five tables and 33,000,000 records;
- Database from the administrative system of the Danish National Archives, containing information about all the incoming scientific research data, and public deliveries of research data. This is a Microsoft SQL Server database containing 289 tables and 1.3 million LOBs;
- Database from the administrative and health records system, from the Ministry of Higher Education and Science, containing information about social, psychological, and psychiatric counselling to students. This is a Microsoft SQL Server database containing 180 tables and 100,000 LOBs.

The pilot report confirmed that all the above pilot scenarios have been completed successfully. The following testimony was given by the Danish National Archives regarding the DBPTK pilot.

> As part of the E-ARK project we tested and used the open source Database Preservation Toolkit (DBPTK) developed by Bruno Ferreira from KEEP SOLUTIONS. The aim was to export relational databases from four real life relational database management systems and transform these exported databases from their proprietary format into the system independent archiving formats SIARDDK and SIARD 2.0. The SIARD 2.0 format was developed by the E-ARK project and the Swiss Federal Archives.

The development of the DBPTK was performed in a professional way using an issue tracker system that gave us the possibility to easily report feature requests and bugs in a prioritized way. Issues were dealt with rapidly and professionally, referring to the SQL Standard as the basis. Changes in the source code between versions were shown, and to the extent that we could understand the code it seemed well organized and modularised. Having successfully tested the DBPTK (in an iterative way with new code releases) on a few small sample databases we contacted the four producers (the owners of the databases) and made agreements on when we could arrive at their site and try to export the databases. We connected to the their RDBMS using a laptop and an external drive. The most challenging task was on the producer side, namely to ensure that we had been provided with a user having sufficient rights on the database system to export the database we desired. The DBPTK was sufficiently fast enough that we could perform the export within normal working hours. In a few cases we not only exported to SIARD at the producer site but also received a backup of the database in its proprietary RDBMS format, giving us the possibility to redo the export at the premises of the Danish National Archives.

All four databases were successfully exported to both SIARD 2.0 and SIARDDK format. Since we have our own validation tool for SIARDDK (defined in the Danish Executive Order on Submission Information Packages – no.1007), we can say that the exports comply with SIARDDK. We have also manually validated against the SIARD 2.0 format (defined as the Swiss eCH-0165 SIARD Format Specification 2.0) and found no errors. Nevertheless, we still find that in the future the users of SIARD 2.0 should develop an open source validation tool that systematically validates the SIARD 2.0 format.

The results from the pilot show that the E-ARK project open source Database Preservation Toolkit (DBPTK) made by project partner KEEP SOLUTIONS can export live databases in proprietary formats to system independent archiving formats SIARD 2.0 and SIARDDK (the latter provided by E-ARK project partner Magenta).

The coverage, efficiency and stability, in fact the overall impact of the open source DBPTK is such that we at the Danish National Archives has decided to stop further development of our own prototype for database export and solely rely on the open source DBPTK. This open source tool efficiently exports databases and can therefore reduce the cost and time spent for producers of SIPs as well for the archives due to fewer errors being submitted. We encourage all parties interested in digital archiving to support its future use and development and we hope to see it along an open source validator of SIARD.

— Digital Archivist Phillip Tømmerholt and Senior Adviser Anders Bo Nielsen, Danish National Archives

The DNA also piloted the DBVTK, by performing user testing and creating a report with the conclusions. The user testing asked four users (with different levels of knowledge regarding databases, from completely unaccustomed to a database administrator) to perform some tasks on the tool and "think aloud" as they perform those tasks. The person conducting the test would register their thoughts while using the tool, and whether they were able to complete the tasks. The final report on this activity provided a lot of useful feedback and different opinions from different points of view. The following testimony was given by Ann-Kristin Egeland, who planned and conducted the user test.

> I have used the Database Visualization Toolkit (DBVTK) and it helps me in the process of searching and exporting information from the databases preserved by the archive. I already work with a similar tool, but the DBVTK improves my workflow by allowing searching across all tables, automating the creation of advanced search forms for a single table and displaying the table- and column descriptions in the result of a search. A tool like DBVTK is of great use in the digital preservation field of the archive, because few end users have the skills to write SQL queries into the databases them self. And end users will always be better at finding what they are searching for than the IT archivist.
>
> Beside the improvements mentioned above, the layout of the DBVTK is also more intuitive and user friendly than what I am used to. However, the tool is basically useful for searching in simple database and after single information. When searching in complex databases with many tables and relations it is quite confusing for an end user to navigate through the tables and almost impossible to get an overview of all the information in the database linked to one subject or person. Here a traditional database management system is more suitable because precise SQL queries can be defined across many tables. However, the traditional database management system lacks the metadata from the archival information package and this way slow down the process with identifying the content of tables and columns. Also export of tables of "one to many relations", e.g. all the actors starring in one film, is not possible in the DBVTK, but a typical use case.
>
> All in all, the automation of search forms for simple databases definitely improve the accessing workflow a lot combined with the idea of defining a DIP in a traditional database management system designed for the specific user need and creating it with the Database Preservation Toolkit (DBPTK). This way the preparation of the DIP for the user is the same as today regarding the analysis

> of the complex database, but the time spend on defining the search forms are saved and the search functionality for the end user is improved.
> — Ann-Kristin Egeland, System Developer at the Danish National Archives

The DBVTK version piloted by the DNA did not yet include the database diagram (corresponding to requirement 14), making it difficult to understand the database and find specific information.

National Archives of Hungary

The National Archives of Hungary (NAH) pilot used both DBPTK and DBVTK to migrate databases and access them:

- Use DBPTK to convert several databases from Oracle DBMS to SIARD 2;
  - A database that is not normalized[2] and contains more than 300,000 cases of the Hungarian Prosecution Office;
  - At least two more databases with different characteristics and containing both restricted and open content.
- Use DBPTK to convert the databases from SIARD 2 back to Oracle DBMS;
- Add the databases to the DBVTK and make it available for consumers to access and provide feedback;

The pilot is ongoing, but preliminary feedback confirmed that some data has been successfully converted using the DBPTK, and provided some suggestions to improve the DBVTK. The following testimony was given by the National Archives of Hungary regarding the DBPTK and DBVTK pilot.

> According to a 2012 survey on electronic records, 92 percent of public data are held in relational databases. During NAH's first e-archiving project, digital preservation was in its infancy. Today, with the raise of Big Data and Internet of Everything these challenges are of a different magnitude. In this age the only way for public archives to fulfil their mandate in protecting democracy, strengthening personal and community identity and preserving the evidences the roots of our culture is to automate and standardize their processes along with stakeholders of different sectors.

2 A non-normalized database does not conform with the normal forms. The normal forms are several guidelines to reduce data redundancy and improve data integrity. (Codd, 1970, 1971)

> First, what we wanted to achieve in the National Archives of Hungary was to able to export data from a live oracle database (RDBMS) into SIARD 2.0 format, and also to ingest SIARD 2.0 packages into a live oracle database (RDBMS). We tried out and used DBPTK because of the E-ARK Project, and we are planning to use it in the future. KEEP SOLUTIONS were able to keep up a very reliable fast support when the tool was under development, and they still are! By now they developed a trustworthy software, capable of exporting and importing from/to different kinds of popular database management systems, in SIARD 2.0. What helped their work was the log file what DBPTK always makes during a process. This kind of logging method, that logs every little detail, always helps the developers to isolate problems.
>
> Using DBVTK, people are able to see a whole database in SIARD 2.0. With other tools what we used since 2014, we were able to open a bit smaller, SIARD 1.0 file at a time, because the program had to work on our local computer. It means, every time we wanted to see into an exported SIARD 1.0 database, we had to wait to the computer to finish with the preparing. In contrast, DBVTK can visualize lots of databases in SIARD 2.0 on our local machine, or on a remote server. With this tool, we would open databases more confidently, that holds terabytes or petabytes or even more data. And because it is running as a service, after we closed the web interface, we just open it again, so we can continue browsing those huge databases. What we really liked in the previous tools and DBVTK can't do, is the ability, to edit and save a SIARD file. This is what we miss from DBVTK.
>
> What makes SIARD so important is in it's name: Software Independent Archival of Relational Databases – for longer-term preservation. With DBPTK and DBVTK this can be possible more easily and more productively. Comparing to other products in the field I would like to emphasize the detailed documentation and user-friendliness of the GUI's which going to enable their broad proliferation.
>
> — Zoltán Szatucsek, Zoltán Lux and József Mezei, National Archives of Hungary

National Archives of Estonia

The National Archives of Estonia (NAE) pilot aimed to use the DBPTK to migrate a database with a complex structure and around 200.000 records to the SIARD 2 format; the pilot also aimed to test the functionality provided by the DBVTK. The following testimony was given by the National Archives of Estonia regarding the DBPTK and DBVTK pilot.

The NAE has since 2002 been addressing the issue of archiving and preserving relational databases of the Estonian government sector. In general, the process of exporting database content into a reasonable open preservation format and reusing archived databases has been rather manual, erroneous, time consuming and costly due to the lack of good quality and scalable tools.

In 2014 we got aware of an initial version of the Database Preservation Toolkit (DBPTK) and since then we have been testing and evaluating it for the purpose of replacing our current tools and methods.

DBPTK has been tested a lot from our side. Our main test cases were the archiving of databases built on MYSQL and POSTGRE platforms. In addition, we have also tested the cross-migration of some databases (for example from MSSQL to MYSQL). Even while some bugs where encountered during the testing process, we would like to highlight the good level of communication and prompt availability of bug fixes from Bruno Ferreira.

The DBPTK in general is very flexible and its command line interface is easy to use for its intended users (database administrators). After the initial bugs were fixed the DBPTK is also sufficiently stable and scalable to be used for the archiving of crucial Estonian government databases. Based on our testing we are certain that we can start using DBPTK throughout the Estonian public sector in 2017 and that the application of the tool will provide us with a crucial increase of automation and throughput, therefore streamlining the whole process, making it less costly and more effective. In short – the DBPTK is one of the most useful and practical tools in digital archiving.

We hope that the work on the DBPTK will continue, also that a GUI will be added which would allow the DBPTK to be also used by less technical users (as an example archivists) as well as provide some additional functionality like saving and reusing database connection profiles.

In addition to the DBPTK we have also tested the Database Visualisation Toolkit (DBVTK). While not expected to be as mature as the DBPTK it delivers already good possibilities for searching and browsing a preserved database. As an end-user oriented tool it is worth to mention that it also has a pretty and understandable layout, the search works fine and all data is accessible.

We would recommend continuing to work on additional functionality (as an example to provide a visualisation of the data model/tables and their relations), to simplify ways for uploading preserved databases into the DBVTK, and to continue testing the tool for scalability and stability.

> After these updates the National Archives of Estonia would love to implement the tool as the default access tool for archived relational databases.
> — Kuldar Aas and Lauri Rätsep, National Archives of Estonia

National Archives of the Republic of Slovenia

The National Archives of the Republic of Slovenia provided a lot of feedback on the migration of Microsoft Access databases to SIARD 2, and suggested improvements to the DBVTK. The following testimony was given by the National Archives of the Republic of Slovenia regarding the DBPTK and DBVTK.

> As members of the E-ARK project we have been involved in specification and test activities for the tools DBPTK and DBVTK. In both areas, i.e. database preservation and access, we have had previous experiences using our own in-house tools. Therefore, it was relatively easy to start work with the tools that will become the outcome of the E-ARK project.
>
> The DBPTK brings a tool that is based on the new SIARD-2 format. I appreciate the ease of use in both areas of functionality: extracting data from the source database and creation of a database in the archive environment. A complicated technical task is done automatically and this saves us a lot of effort. Even more important is minimizing the human errors that are always a risk and demand a detailed testing. During the project, good on-line documentation and quick communication with the development team helped us resolve the issues that came up during testing. As expected from the test process, we were supposed to cover as much scenarios as possible to test the functionality and quality of the tool. Luckily, the real life scenarios had real-life errors in the data tables and some fine-tuning of the tools was needed in order to enable ingest of such data as well. This is important because the archivists are not supposed to do the corrections of the data, and the creator of the data is not may not be willing to do it just before the delivering to the archive. The typical reason is that usually the applications are not in use any more, original administrators are not around and the current keeper simply wants to deliver what has been waiting on the disk. The DBPTK tool can now be delivered to the creator and they can create the package without the assistance of the archive. Of course they need to follow the archival instructions about the process and needed deliverables. Current implementation fulfils these needs and also allows integration with our environment and other tools. The wide international use of the tool minimizes the risks of insufficient support in the future, and that is critical for us. However, we still have archival material in dBase format that is not supported by DBPTK.

We will decide whether to first migrate to MS Access and then use DBPTK, or use our in-house format that is based on CSV files, and the procedure to export from dBase is already established. I am sure that future versions of the DBPTK will increase the number of supported databases. It is also not sufficient to rely only on typical database scenarios as some real life applications can implement more complex or sometimes even badly designed solutions that will be bringing new challenges to the team in order to create them "easy and understandable" for the archival use in the far future. For this reason, the process of creating the package needs to be tested on as many as possible real life databases to enable further improvements of the tool.

The second tool is DBVTK. It is supposed to fulfil the gap between the ease of access to the data in the original application and a raw database in the archives. Firstly, the DBVTK provides technical explanation of the entities based on SIARD-2 metadata. Besides the views, the queries and reporting descriptions are not part of this SIARD-2 metadata and thus cannot be automated. Therefore, the descriptions of the archival use have to be prepared separately and this is then technically enabled by the functionality of DBVTK. Current and future researchers can examine and research the database data using the Search functionality. Nevertheless, with current test version we miss some functionality for a typical archival use and where less skills are required: simple creation of customized SQL queries and formatting of their output. This would enable us to simulate the search and reporting functionality of original applications. Current full text search approach is just one use scenario but the search results are too unclear to provide a basis for an official statement of the archives about some certain inquiry. For example, a person loses a school certificate and comes to the archives. There, the archivist executes a prepared query in the school database and gets the data about this person that are enabling him/her to issue an official statement about the status in the database. In this scenario the archivist expects a clear and unambiguous output. In this area, in our archives we have been using the dbDIPview tool as a generic application for accessing the databases where the typical queries/reports are prepared in advance as XML files. This enables the archivists and some other target users to get immediate answers to their queries. We are aware that a long-term solution needs to be widely accepted and in use by other archives. Late start of the DBVTK activities on the E-ARK project did not allow to implement and test all possibilities. The typical scenario during the archival ingest is to create the "simulation" of the most important reports of the original application thus allowing the archivist to review and verify the ingested material and in the same process also provide a

basis for the future access. Of course, the researchers will find many other ways to use the data.
— Boris Domajnko, National Archives of the Republic of Slovenia

National Archives and Records Administration

The National Archives and Records Administration (NARA, United States National Archives) have been doing some preliminary testing with the DBPTK and the DBVTK, and are interested in piloting these tools in the E-ARK project.

NARA plans to extract databases from an Oracle system and examine the applicability of data-warehouse concepts in an archival environment, in order to maintain both the original structure and intellectual interpretability of ingested data. The access to the archived databases will be provided by the DBVTK.

The databases include the "Bureau of Export Administration's Export Control Automated Support System" database, containing information spanning from 1976 to 1997; and the "Environmental Protection Agency's Program Tracking, Advisories, Water Quality Standards, and Nutrients (PRAWN)" database, containing information spanning from 1995 to 2016. NARA also plans to archive a SIARD database in their current archival system, the Electronic Records Archives, and then access it using the DBVTK and by loading it into a live DBMS.

Unofficial pilots

There were other institutions piloting the tools in a slightly less formal way. Some institutions that piloted the tools also tested them with other databases that were not initially planned to be in the pilot, further improving the quality of the project's artefacts. Although most of the feedback was obtained from the pilots, there were individuals and other institutions that tried the tools and contributed to the project by reporting problems and suggesting improvements.

6 CONCLUSION AND FUTURE WORK

This chapter presents the conclusions that can be extracted from this work, the main contributions it provides to the digital preservation field and a description of future work.

Databases are commonly used to store critical information that would be very hard to recover if it was somehow lost or destroyed. Some of that information must remain accessible for multiple decades, e.g. for legal purposes.

The digital preservation field attempts to preserve information so it can be experienced in the future, but there is no widely accepted way to preserve databases, due to the complexity of the digital object and the lack of consensus on database significant properties.

This work, inserted in the E-ARK project, attempted to define a methodology to preserve databases. Format migration was the chosen preservation strategy, for its proven effectivity. To execute this strategy a new format was created, the SIARD 2, built on top of the SIARD format and maintaining the most commonly agreed upon database significant properties. The common workflow includes producers migrating the databases to SIARD 2 format and submitting them to the archive, where they would be stored for some years until they are requested, and then providing access to them through an application that could render SIARD 2 files to consumers.

This work improved the SIARD 2 specification, that was in its final development phases, and developed the needed migration and access tools: The Database Preservation Toolkit and the Database Visualization Toolkit, respectively. The tools were developed according to the requirements and use-cases specified by the E-ARK project, and were submitted to some real-world testing via piloting.

The DBPTK was made flexible to support different database management systems and formats, including their specific features and optimizations; as well as to perform on low computing hardware requirements, even when converting databases containing millions of records.

The DBVTK provides access to databases preserved in the SIARD 2 format, with a fast and intuitive interface in which consumers can search and explore the preserved databases.

The pilots, composed of national archives and other memory institutions, tested both tools to make sure that they were usable in a real-world environment and that they were capable of handling complex databases that were submitted to the archives.

The validation provided by the E-ARK Pilots determined that all the requirements set forth by the E-ARK Project for these tools were achieved. Proven to be viable by the pilots and presented in conferences worldwide by members of the E-ARK Project, these tools already had an impact on the community and the digital preservation field, including in institutions that considered long-term database preservation to be impossible or not possible in the near future. Janet Delve, Co-ordinator of the E-ARK project, has contributed some thoughts on the tools and their impact in the database preservation field.

> As Co-ordinator of the E-ARK project, I am extremely impressed with the performance of the Database Preservation Toolkit. As used currently in the project, the toolkit is considerably advanced, compared to previous versions, and it can take many different makes of database (SQL Server, Oracle etc.), and convert them into an archival database format, or into a different make of database. It can also carry out the reverse process. This functionality is extremely useful for archives, business, government, the creative industries: in fact, anywhere where databases are used, and need to be kept for future use. We have found the tool to be reliable, and extremely versatile, and we are very pleased with its performance during the project. It has featured successfully in several of our pilots, and indeed we have two new pilots: The National Archives of Chile; and the National Archives and Records Administration of America, who are each carrying out pilots solely on the Digital Preservation Toolkit. This is very prestigious for the project, and brings great credit to all involved. It is particularly satisfying to note that the project archival partners themselves are very pleased indeed with the Database Preservation Toolkit.
>
> In E-ARK we have collaborated with the Swiss Federal Archives to produce the latest version (2.0) of the SIARD (Software Independent Archiving of Software) standard. This is the only such standard worldwide, and is much in use internationally. The Database Preservation Toolkit can convert many different makes of databases into SIARD 2.0 (and the toolkit is also backwards compatible with SIARD 1.0). In E-ARK we have created a Data Mining showcase, to display advanced querying techniques based on Big Data (Hadoop clusters, SOLR searches etc.) as well as Data Warehousing techniques/tools such as OLAP (Online Analytical Processing), Dimensional Modelling and Data Marts. As part of this activity, we have demonstrated a use case at the National Archives of Hungary, where a complex judicial database was loaded into the Database Preservation Toolkit, then converted into SIARD 2.0 format. A considerably simplified dimen-

> sional model was developed for a Data Mart, and the SIARD 2.0 format data was then loaded into it using the Oracle Data Integrator. The resulting OLAP cube was then queried to produce excellent discovery outputs for the archivists and judicial department, including graphs, charts, and tables with understandable headings instead of the obscure codes in the original database. So we have been able to demonstrate a clear case for gaining Business Intelligence when using the Database Preservation Toolkit.
> — Janet Delve, Co-ordinator of the E-ARK project and Professor of Digital Humanities at the University of Brighton

The national archives that are using the tool (the Danish National Archives, the National Archives of Hungary, the National Archives of Estonia, the National Archives of the Republic of Slovenia and the National Archives and Records Administration) have already archived several open and private client databases of different natures, such as medical, social, cultural, environmental and judicial using the Database Preservation Toolkit and the Database Visualization Toolkit. Further demonstrating that the tools are able to handle real-world databases of different natures and sizes.

FUTURE WORK

The developed tools, although functional, can still be improved. Some of those improvements are described in this section.

Graphical user interface

Even though the DBPTK needs to be executable in a command line environment, it would be simpler to use by non-technical personnel if it was accompanied by a graphical user interface, specially for demonstration purposes.

External modules

Supporting external import and export modules in DBPTK (i.e. as plugin modules) would be an improvement to the existing application, since new modules could be developed and simply added to a specific folder and DBPTK would automatically start using these modules to support more database systems and formats. This functionality would improve the portability and maintainability of the DBPTK and its modules.

Materialize views in SIARD

When migrating databases to the SIARD 2 format, there could be a way to retrieve and store the views' contents. From there, the DBPTK SIARD 2 export module could support exporting not only specific tables, but also views; and the DBVTK would be also able to display the contents of the views. This allows an archivist to manipulate information by following a work flow similar to this one:

1. Use the DBPTK to load the SIARD into a DBMS;
2. Create views that hide sensitive data or prepare data in some special way;
3. Create a new SIARD file containing only the views (containing not only the views' metadata but also their data);
4. Provide access to the new SIARD file.

This is an improvement over the current method, in which the archivist would have to do the materialization of the view in the DBMS, creating it as a table, and then migrating the table to SIARD.

Support SQL queries in DBVTK

The DBVTK uses Solr, which is a NoSQL database, as its data layer, which does not support SQL querying. But the pilots' feedback is that they are used to use SQL to search the database, and that it not being available in DBVTK seems strange.

Solr version 6 introduced support for SQL and the ability to search collections as if they were tables. Using this functionality was not a priority during the development of DBVTK, but it would be an improvement to the tool's capabilities.

SIARD 2 metadata editor

Editing the SIARD 2 metadata, e.g. to add descriptive information, is a complex task that requires knowledge of the SIARD format specification and appropriate tools to extract, edit and repackage the SIARD.

Since it would be troublesome to try to include this functionality in the DBPTK or DBVTK, as it would veer from the tools' intended purpose, a new software could be developed for this end. This software could even be built on top of DBPTK, by using the SIARD import module to obtain information from the SIARD, providing a graphical user interface to change the metadata information (contained in the DBPTK internal data model), and then using the SIARD export module to create the modified SIARD.

BIBLIOGRAPHY

Alföldi, I., Fülöp, Z., and Szatucsek, Z. (2014). General pilot model and use case definition, e-ark deliverable 2.1, revision 1.3). Technical report, E-ARK.

Alföldi , I. and Réthy, I. (2016). Detailed pilots specification, e-ark deliverable 2.3, revision 2.1. Public deliverable, E-ARK.

Boutell, T. (1997). Png (portable network graphics) specification version 1.0. Technical report.

Brandl, S. and Keller-Marxer, P. (2007). Long-term archiving of relational databases with Chronos. *First International Workshop on Database Preservation (PresDB'07)*, (March).

Bruggisser, H., Büchler, G., Dubois, A., Kaiser, M., Kansy, L., Lischer, M., Röthlisberger-Jourdan, C., Thomas, H., and Voss, A. (2013). eCH-0165 SIARD Format Specification. Standard, Swiss Federal Archives, Berne, Switzerland.

Chikofsky, E. J. and Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. *IEEE software*, 7(1):13–17.

Clark, J., DeRose, S., et al. (1999). Xml path language (xpath) version 1.0.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM*, 13(6).

Codd, E. F. (1971). Further Normalization of the Data Base Relational Model. *ACM Transactions on Database Systems*.

Connolly, T. M. and Begg, C. E. (2004). *Database Systems: A Practical Approach to Design, Implementation and Management (4th Edition)*. Pearson Addison Wesley.

Consultative Committee for Space Data Systems (2002). Reference Model for an Open Archival Information System (OAIS). *forSpace Data Systems*, (January):1–148.

CSP Software GmbH & Co. KG (2015). Declaration of Incorporation, CHRONOS Version 4.9. Technical report.

Danish National Archives (2010). Executive order on submission information packages. Executive order, Danish National Archives.

Dappert, A. and Farquhar, A. (2009). Significance is in the eye of the stakeholder. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5714 LNCS:297–308.

E-ARK Project (2014). About the e-ark project. url: `http://www.eark-project.com/about`, (visited on 05/09/2016).

European Comission (2008). *Model requirements for the management of electronic records (MoReq2)*. Office for Official Publications of the European Communities.

Faria, L. (2015). *Automated Watch for Digital Preservation*. Doctoral dissertation, University of Minho.

Faria, L., Nielsen, A. B., Röthlisberger-Jourdan, C., Thomas, H., and Voss, A. (2016). eCH-0165 SIARD Format Specification 2.0 (draft). Standard, Swiss Federal Archives, Berne, Switzerland.

Ferreira, B., Faria, L., Ramalho, J. C., and Ferreira, M. (2016). Database Preservation Toolkit - A relational database conversion and normalization tool. *Proceedings of the 13th International Conference on Digital Preservation: iPRES 2016*.

Ferreira, M. (2006). *Introdução à Preservação Digital: Conceitos, estratégias e actuais consensos*.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext transfer protocol–http/1.1. Technical report.

Fitzgerald, N. (2013). Using data archiving tools to preserve archival records in business systems - a case study.

Freitas, R. A. P. and Ramalho, J. C. (2010). Significant properties in the preservation of relational databases.

Geiger, K. (1995). Inside ODBC. Redmond.

Gordon, G. and Nadig, H. (1977). Hexadecimal signatures identify troublespots in microprocessor systems. *Electronics*, 50(5):89–96.

Grace, S., Knight, G., and Montague, L. (2009). Investigating the Significant Properties of Electronic Content over Time: Final Report. Technical report, King's College London.

Hamilton, G., Cattell, R., Fisher, M., et al. (1997). *JDBC Database Access with Java*, volume 7. Addison Wesley.

Hockx-Yu, H. and Knight, G. (2008). What to Preserve?: Significant Properties of Digital Objects. *International Journal of Digital Curation*, 3(1):141–153.

Hodge, G. and Frangakis, E. (2004). Digital Preservation and Permanent Access to Scientific Information: the State of the Practice. Technical Report April, International Council for Scientific and Technical Information & CENDI.

ISO Standard 14721:2012(E) (2012). ISO 14721:2012, Space data and information transfer systems – Open archival information system (OAIS) – Reference model. Standard, International Organization for Standardization.

ISO Standard 8601:2004(E) (2004). ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times. Standard, International Organization for Standardization.

ISO Standard 9075:1999(E) (1999). ISO 9075:1999, Information technology – Database languages – SQL. Standard, International Organization for Standardization.

ISO Standard 9075:2008(E) (2008). ISO 9075:2008, Information technology – Database languages – SQL. Standard, International Organization for Standardization.

Jacinto, M., Librelotto, G., Ramalho, J. C., and Henriques, P. R. (2002). Bidirectional conversion between XML documents and relational databases. In *The 7th International Conference on Computer Supported Cooperative Work in Design*. COPPE/UFRJ.

Lagoze, C., Payette, S., Shin, E., and Wilper, C. (2005). Fedora: An architecture for complex objects and their relationships. *Journal of Digital Libraries - Special Issue on Complex Objects*, abs/cs/0501012.

Lavoie, B. (2014). Information System (OAIS) Reference Model: Introductory Guide. Technical report, Digital Preservation Coalition.

Lee, K.-h., Slattery, O., Lu, R., Tang, X., and Mccrary, V. (2002). The State of the Art and Practice in Digital Preservation. *Journal Of Research Of The National Institute Of Standards And Technology*, 107(1):93–106.

Masinter, L., Berners-Lee, T., and Fielding, R. T. (2005). Uniform resource identifier (uri): Generic syntax.

Michael, M., Moreira, J. E., Shiloach, D., and Wisniewski, R. W. (2007). Scale-up x scale-out: A case study using nutch/lucene. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–8. IEEE.

Mockapetris, P. V. (1987). Domain names-concepts and facilities.

Norwegian National Archives (2011). ADDML - Archival Data Description Markup Language 8.2. Standard, Norwegian National Archives.

Pohlmann, K. C. (1989). *The compact disc: a handbook of theory and use*, volume 5. AR Editions, Inc.

Pokorny, J. (2013). Nosql databases: A step to database scalability in web environment. *International Journal of Web Information Systems*, 9(1):69–82.

Ramalho, J. C., Ferreira, M., Castro, R., Faria, L., Barbedo, F., and Corujo, L. (2007a). XML e Preservação Digital.

Ramalho, J. C., Ferreira, M., Faria, L., and Castro, R. (2007b). Relational database preservation through xml modelling. *International Workshop on Markup of Overlapping Structures (Extreme Markup 2007)*.

Swiss Federal Archives (2008). *SIARD Format Description*.

The Commission on Preservation and Access and The Research Libraries Group (1996). Preserving Digital Information: Report of the Task Force on Archiving of Digital Information. *Archives and Museum Informatics*, 10(2):148–153.

The Unicode Consortium, C. (1996). *The Unicode Standard, Version 2.0*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Thibodeau, K. (2002). Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years. In *The State of Digital Preservation: An International Perspective*. Council on Library and Information Resources.

Wilson, A. (2007). Significant Properties. pages 1–10.

Woodyard, D. (2000). Digital preservation: The australian experience. In *3rd Digital Library Conference: Positioning the Fountain of Knowledge*, Malaysia. State Library of Sarawak.

Ylonen, T. and Lonvick, C. (2006). The secure shell (ssh) protocol architecture.

ZIP 6.3.3 (2012). .ZIP File Format Specification, version 6.3.3. Standard, PKWARE Inc.

A

SIARD 2 FORMAT DETAILS

A.1 DATABASE METADATA ELEMENTS

The following list describes the metadata elements supported by SIARD 2.

DBNAME
The database name used in the original DBMS;

DESCRIPTION
A textual description of the database. This description would ideally include the meaning and content of the database as a whole;

ARCHIVER, ARCHIVER CONTACT
Name and contact information of the person who carried out the migration of the database into SIARD format;

DATA OWNER
The institution or person that, at the time of archiving, has the right to grant usage rights for the data and is responsible for compliance with legal obligations such as data protection guidelines;

DATA ORIGIN TIMESPAN
The originating period of the data in the database;

PRODUCER APPLICATION
The software that converted the database into SIARD format;

ARCHIVAL DATE
Date on which the database was converted to SIARD;

CLIENT MACHINE
The identification of the computer used to create the SIARD file, e.g. the DNS[1] name.

1 The Domain Name Service is used to consistently refer to resources over a network (Mockapetris, 1987).

DATABASE PRODUCT
Information (name and version) of the DBMS from which the data was migrated;

CONNECTION
The connection used to access the database, in textual format. This commonly includes information such as the DBMS name and access port, as well as the user name and password used to connect. The password may be filtered out for security reasons;

DATABASE USER
The name of the database user that carried out the database migration.

SCHEMAS
List of schemas in the database. The schema element will be detailed in this section.

USERS
List of database users. Each user element contains a name and a description, that should include details about the user's significance and function.

ROLES
List of user roles. Each role element has a name, a description, and the name of its administrator (which can be a user or another role).

PRIVILEGES
List of user and role privileges. The privilege element includes information about the type of privilege granted and the object it affects, the grantor and grantee, the grant option and the privilege description.

LOB FOLDER
A technical element related to the placement of LOBs. Other elements contain more information related to LOBs, which are discussed in section 4.1.4.

MESSAGE DIGEST
A digest message generated by running a digest algorithm over the database contents in the SIARD file. By running the digest algorithm on a copy of the SIARD file and comparing the result to the original digest message, one can confirm that the file contents are intact;

A.2 TABLE METADATA ELEMENTS

The following list describes the characteristics of the table element in SIARD 2.

NAME
The table name;

FOLDER

The "`table`*N*" folder in figure 9, inside which are the table contents as an XML file;

DESCRIPTION

Textual description of the purpose and contents of the table;

COLUMNS

List of columns in the table. Each column has information about the column types (the original and SQL ISO Standard 9075:2008 (E) equivalent types, or the user defined type name), its default value and whether it can have `NULL` values or not, and the column's description. The column also contains additional information needed to handle UDTs having fields of LOB types.

PRIMARY KEY

The primary key metadata, consisting of a name, a description and the list of columns that compose the key.

FOREIGN KEYS

List of foreign keys in the table. Each foreign key has a name, description, a match type, a delete action and an update action[2]. This element also contains information needed to preserve the foreign key relation, namely the referenced schema, the referenced table and a list of reference elements; with each reference element mapping a column to its counterpart in the referenced table.

CANDIDATE KEYS

List of candidate keys for the table. Each candidate key has the same information as a primary key: a name, a description and the list of columns that compose the key.

CHECK CONSTRAINTS

List of constraints in the table. Each constraint element includes a name, a description and the original condition expressed in SQL.

TRIGGERS

List of triggers in the table. A trigger has a name, a description, the action time ("`BEFORE`", "`AFTER`" or "`INSTEAD OF`" an event) specifying when the trigger will execute after being triggered, the trigger event (specifying when the trigger is triggered), and the original trigger action expressed in SQL.

ROWS

The number of records in the table.

2 The match type specifies how to match values when navigating in foreign key relations. The delete and update actions are executed whenever a row is deleted or updated, and are used as a less verbose way to define some commonly used actions (the alternative would be to write a trigger implementing the same behaviour). All of these elements are detailed in the SQL ISO Standard 9075:2008 (E).