Database Preservation Toolkit

A relational database conversion and normalization tool

Bruno Ferreira
KEEP SOLUTIONS
Rua Rosalvo de Almeida 5
4710 Braga, Portugal
bferreira@keep.pt

Luís Faria
KEEP SOLUTIONS
Rua Rosalvo de Almeida 5
4710 Braga, Portugal
lfaria@keep.pt

José Carlos Ramalho
University of Minho
4710 Braga, Portugal
jcr@di.uminho.pt

Miguel Ferreira
KEEP SOLUTIONS
Rua Rosalvo de Almeida 5
4710 Braga, Portugal
mferreira@keep.pt

ABSTRACT

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

To complement the conversion software, the Database Visualization Toolkit is introduced as a software that allows 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 and promptly present search and filter results on millions of records.

This paper describes the features of both tools and the methods used to pilot them in the context of the European Archival Records and Knowledge Preservation project on several European national archives.

Keywords

Preservation; Archive; Relational Database; Migration; Access; SIARD

1. INTRODUCTION

Databases are one of the main technologies that support information assets of organizations. They are designed to store, organize and explore digital information, becoming such a fundamental part of information systems that most would not be able to function without them [5]. Very often, the information they contain is irreplaceable or prohibitively expensive to reacquire, making the preservation of databases a serious concern.

The Database Management System (DBMS) is the software that manages and controls access to databases, which can be described as a collection of related data. These two intrinsically related technologies function together to perform tasks such as information storage and retrieval, data transformation and validation, privilege management and even the enforcement of important business constraints. The most popular databases are based on the relational model[1] proposed by Codd. [5, 4]

The migration of the relational database information into a format well suited for long-term preservation is one of the most accepted strategies to preserve relational databases. This strategy consists in exporting the information of the relational database, including descriptive, structural and behavioural information, and content, to a format suitable for long-term preservation. Such format should be able to maintain all significant properties of the original database, whilst being widely supported by the community and hopefully based on international open standards [7]. Few formats fit this criteria, being the SIARD format one of the main contenders.

The Software Independent Archiving of Relational Databases (SIARD) format was developed by the Swiss Federal Archives and was especially designed to be used as a format to preserve relational databases. Its second version, SIARD 2, retains the (most commonly agreed upon) database significant properties and is based on international open standards, including Unicode (ISO 10646), XML (ISO 19503), SQL:2008 (ISO 9075), URI (RFC 1738), and the ZIP file format. [6, 8, 9, 11, 14, 15]

The manual creation of SIARD files is impractical, therefore an automatic conversion system was developed – the Database Preservation Toolkit (DBPTK). This software can be used to create SIARD files from relational databases in various DBMSes, providing an unified method to convert databases to a database agnostic format that is able to retain the significant properties of the source database. The software uses XML Schema Definition capabilities present in the SIARD format to validate the archived data and can also be used to convert the preserved database back to a DBMS.

The digital preservation process is not complete if the

[1]According to the ranking at http://db-engines.com/en/ranking (accessed on Apr. 2016), where 7 of the top 10 DBMS use the relational model

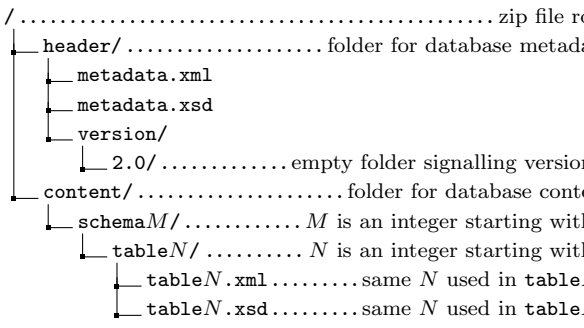

Figure 1: Basic SIARD 2 directory structure.

archived information cannot be accessed. To access and explore digitally preserved databases, the Database Visualization Toolkit (DBVTK) is being developed. This software can load databases in the SIARD format and display their descriptive, structural and behavioural information and content. The viewer also provides the functionality to search and filter the database content as well as export search results.

2. SIARD 2

The second version of the SIARD format emerged from lessons learnt by creators and users of database preservation formats. The SIARD format was originally 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). The SIARD-DK is a variation of the SIARD format created by the Danish National Archives to fit their specific needs. The Database Markup Language (DBML) format, created at University of Minho, was used by the Repository of Authentic Digital Objects (RODA)[2] software to preserve databases at the Portuguese National Archives[3]. The Archival Data Description Markup Language (ADDML) is the format used by the Norwegian National Archives to describe collections of data files. [3, 12, 13, 1]

The SIARD 2 format, in its most basic form, consists of a ZIP file that contains a hierarchy of folders and files of XML and XSD (XML Schema) format, illustrated in figure 1. The XML files inside the SIARD file hold database metadata information and contents.

The `metadata.xml` file contains database description information, such as the database name, description and archival date, the archivist name and contact, the institution or person responsible for the data; database structural information, including schemas, tables and data types; and behavioural information like keys, views and triggers. Such information is useful not only to document the database but also to allow the reliable export of its structure on a different DBMS.

The `tableN.xml` files correspond to each of the database tables and hold the content of the rows and cells from that table. All XML files are accompanied by a corresponding XML Schema file, that can be used to validate the structure and contents of the XML files.

[2]Current version of RODA is available at http://www.roda-community.org/

[3]Direcção-Geral do Livro, dos Arquivos e das Bibliotecas

The SIARD format includes advanced features such as the support for Large Objects (LOBs)[4], and ZIP compression using the *deflate* method [15]. SIARD 2 brings many improvements over the original SIARD and other database preservation formats, mainly the support for SQL:2008 standard and data types, including arrays and user defined types; the strict validation rules present in XML Schema files to enforce valid XML structure and contents; and allowing LOBs to be saved in the `tableN.xml` file, saved as files in a folder inside the SIARD, or saved as files in a location outside the SIARD file. Furthermore, the SIARD 2 specification allows LOB files to be saved in multiple locations or storage devices outside the SIARD file, increasing support for databases which contain large amounts of LOBs.

3. DATABASE PRESERVATION TOOLKIT

The DBPTK is an open-source project[5] that can be executed in multiple operating systems and run in the command-line. It allows the conversion between database formats, including connection to live Relational Database Management Systems, for preservation purposes. The toolkit allows extraction of information from live or backed-up databases into preservation formats such as SIARD 2. Also, it can import back into a live DBMS, to provide the full DBMS functionality, such as SQL[6] querying, on the preserved database.

This tool was part of the RODA project and has since been released as a project on its own due to the increasing interest on this particular feature. It is currently being developed in the context of the European Archival Records and Knowledge Preservation (E-ARK) project together with the second version of the SIARD preservation format – SIARD 2.

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.

Currently supported DBMSes include Oracle, MySQL, PostgreSQL, Microsoft SQL Server and Microsoft Access. All of these support import and export, except Microsoft Access where only import is available, i.e. conversion *from* Microsoft Access is possible, but conversion *to* Microsoft Access is not. 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, given the correct configurations and dependencies, may enable con-

[4]Large Objects, usually stored in binary format in databases. Can also be referred to as BLOBs or CLOBs if they contain binary or character data, respectively.

[5]Software and documentation available at http://www.database-preservation.com

[6]SQL: Structured Query Language

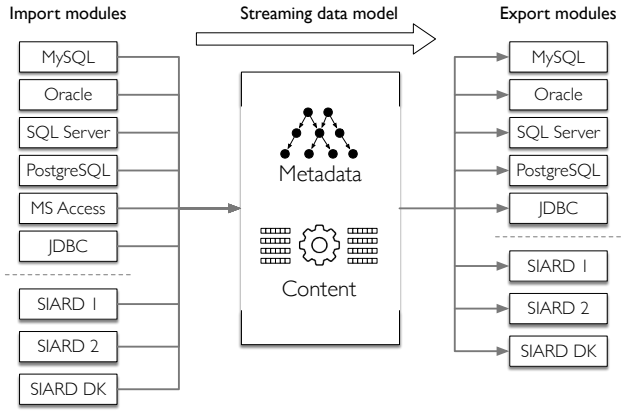

Figure 2: Application architecture overview

version to or from any 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.

The SIARD modules are an essential part of DBPTK, being used to allow the conversion of databases to and from this database preservation format. The SIARD export modules allow filtering out some database tables, as well as exporting contents from views as if they were database tables.

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

Figure 2 depicts an overview of the information flow in the application, 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 without loss, 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 DBMSes that do not check the invariants and constraints imposed by behaviour like primary and foreign keys, or views which have DBMS-specific and untranslatable queries, not supported in the target DBMS.

Figure 4 introduces an overview of a database structure as a hierarchy. As most database structures fit this structure entirely or partially, it is used by all modules. 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 5. 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.

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, serving as a base for a few flavour-specific 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 3.

Most DBMS implementation specific 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 specially for each DBMS. An example of such case is the `YEAR` MySQL data type[7], depicted in figure 3, which the import module first perceives as representing a date, but is in fact a 4 digit numeric type (corresponding to the SQL:2008 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 also vary with the SQL implementation, requiring, for those cases, a specialized interpretation and conversion to an internal standard representation.

Due to the prioritization of 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 broad enough for 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

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

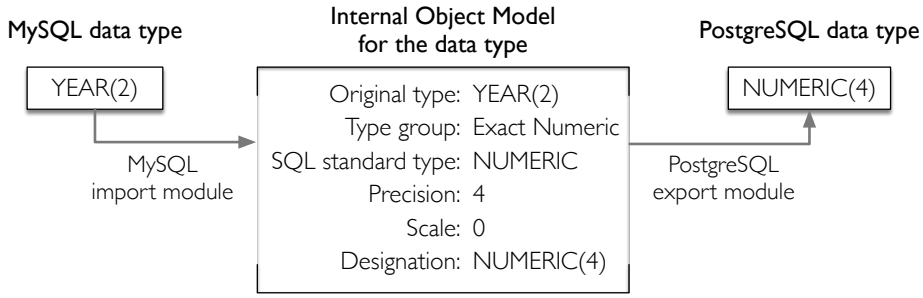

Figure 3: Conversion of MySQL YEAR data type to PostgreSQL

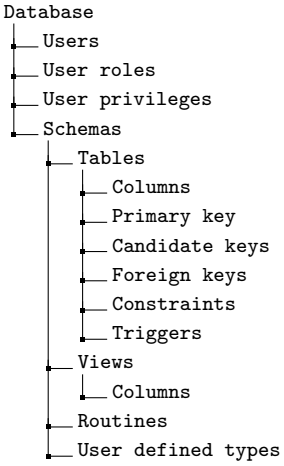

Figure 4: Database structure as an hierarchy.

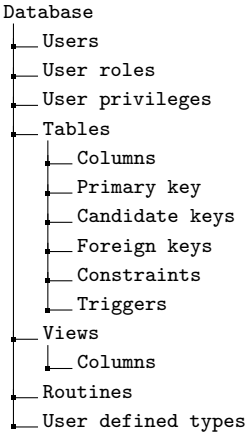

Figure 5: Schema-less database structure as an hierarchy.

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 information loss.

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.

During the conversion, data types and value changes are registered in a report file for manual analysis. This file may also be used as additional documentation for the generated SIARD file.

Some optimizations are also carried out by specific modules. One of those optimizations is implemented in all DBMS export modules and 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. Also it allows to migrate the database even if constraints fail.

When converting the database contents, a flexible internal model must be used to represent different kinds of information and avoid data losses. The import module should select the most adequate model to be used for each record during a conversion.

Some values obtained from the database may not be in a standard format and must be converted to the standard format. A common example of such values are the `DATE`, `DATETIME`, `TIMESTAMP` and other date or time data types, because the format used internally by DBPTK is the ISO standard for representation of dates and times (ISO 8601)[10] and some dates 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.

3.1 Evaluating the Conversion

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 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.

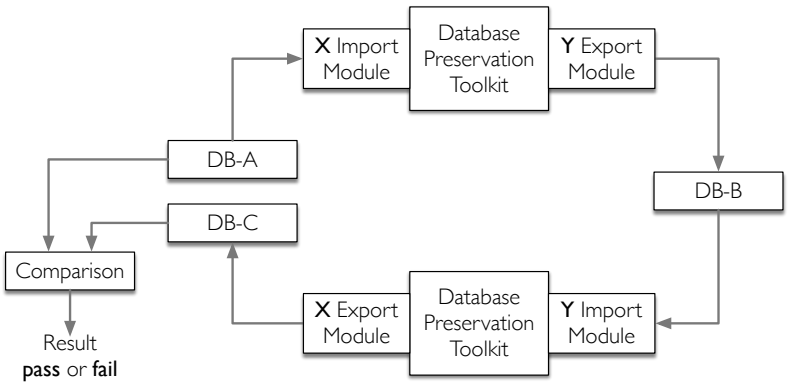

Figure 6: The *roundtrip test*

The *roundtrip test* is described below and illustrated in figure 6.

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*. 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 may still consider the conversion to have succeeded when some database structure and behavioural information was lost or changed. This tolerance 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 3) and would be created as `NUMERIC(4)` when converting the database back to MySQL.

4. DATABASE VISUALIZATION TOOLKIT

The preservation of databases is only successful if there is a way to access the archived databases. To accomplish this, the DBVTK is being developed, allowing archivists and consumers to preview and explore preserved databases in an intuitive interface.

The DBVTK is a scalable web-service that is able to serve multiple archived databases. It displays database description information, structural information, behaviour information and content, providing the ability to search and filter records from a single table or a whole database. Advanced search functionality allows filtering records using multiple search criteria and advanced data searches, such as searching date and time ranges. The DBVTK is optimized to provide almost instantaneous responses to searches on millions of records. Search results can then be exported to formats such as PDF (Portable Document Format) and CSV (Comma Separated Values).

When searching tables containing primary and foreign keys, it is often useful to be able to follow these relations and see the related records from the other table. This functionality in the DBVTK is triggered by clicking a cell containing a primary or foreign key, which will show the records from the other table related to the key. The database structural and description information can also be used to understand these relations.

The DBVTK can integrate with an external authentication and authorization system, providing the means to identify the users and verify their permissions to access each database.

After logging in, users will be able to see the list of databases they can access. By clicking one of the databases the user is shown some database metadata, such as the database name, description or data owner; from there the user can begin searching and exploring the database.

The DBVTK is not backed by a relational DBMS due to scalability and performance issues, instead the Apache Solr[8] platform is being used to store preserved database records. 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.

In order to provide access to preserved databases, the DBVTK requires the database to be loaded into Solr. This is achieved using DBPTK with a Solr export module. This module grants DBPTK the ability to add a SIARD database to a Solr platform such as the one used by the DBVTK (see the top part of figure 7).

As consumers use web-browsers to access the DBVTK web interface and explore databases, the back-end server application retrieves the database records and sends them to the web interface, which shows the records to the consumer (see the bottom part of figure 7).

[8]Apache Solr is available at http://lucene.apache.org/solr/

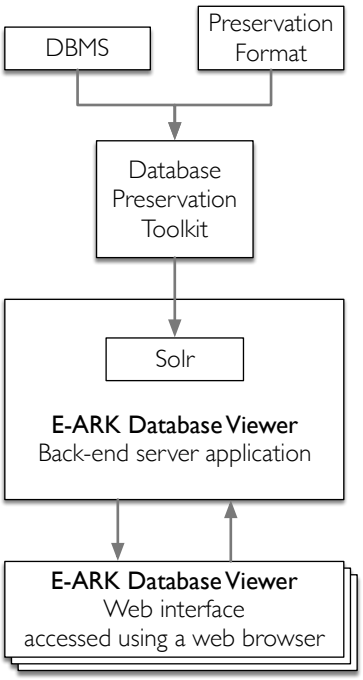

Figure 7: Application architecture overview

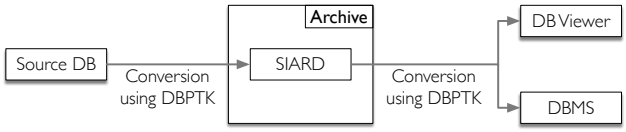

Figure 8: Usage scenario for an archive

5. USAGE SCENARIOS

The Database Preservation Toolkit and the Database Visualization Toolkit can be used in multiple cases to achieve different goals in an archive context.

1. Producer using DBPTK during the pre-ingest phase

During the pre-ingest phase, the producer can use DBPTK to convert a database to SIARD 2. After adding some documentation and other information, the producer can deliver the database to the archive. Such procedure is depicted in the left part of figure 8 and the following usage scenarios correspond to the right part of the same figure.

2. Consumer using DBVTK to explore a database

The archivist grants the consumer access to a database, and after logging in to the DBVTK web interface, the consumer is able to search and filter database records at will.

3. Consumer using DBVTK to explore a database prepared by an expert archivist (add views)

To serve a database with a specific view requested by a consumer, an archivist can use DBPTK to convert the SIARD database to a supported DBMS. The archivist can then use the DBMS to create the requested views, create a new SIARD using DBPTK. The new SIARD file can be exported to the DBVTK.

After being given access to the database, the consumer can access it using the DBVTK web interface to explore and search the records from the views.

4. Consumer using DBVTK to explore a database prepared by an expert archivist (serve only specific views)

An alternative to the previous method can be used when the archivist only wants to make part of the database information available to the consumer. By providing some options when creating the new SIARD file on DBPTK, the archivist may create a SIARD file containing a subset of the tables and views present in the source database.

Even after obtaining access to the new database, the consumer will only be able to access information present in the tables and views that were exported to SIARD. This particularly useful to restrict or completely block access to sensitive database information.

5. Researcher performing complex queries and analysis

A researcher may initially act as a consumer, requesting access and exploring databases until a suitable database is found. At that point, the researcher could obtain the suitable database in SIARD format, use DBPTK to convert it to a supported DBMS, and finally use the DBMS functionality to research the data. This allows a researcher to use Data Mining and OLAP techniques to research archived databases.

6. E-ARK PILOTS

The DBPTK is being piloted in the context of the E-ARK project by the Danish National Archives, the National Archives of Estonia and the National Archives of Hungary. [2]

The Danish National Archives pilot goal is to make four successful data extractions from live authentic databases into the SIARD 2.0 format:

- Extract records from Microsoft SQL Server database bigger than 100 GB (with a minimum success rate of 90%);
- Extract records from a large database containing documents;
- Extract records from Ms SQL database containing 50-60 tables and about 90.000 records (with a minimum success rate of 90%);
- Extract records from Microsoft SQL Server database containing about 5 million records.

One of the National Archives of Hungary goals is to convert an Oracle database to and from a preservation format, and accessing it using the DBVTK. This database is not normalized and contains more than 300.000 cases of the Hungarian Prosecution Office. The archives also aim to migrate two or more databases with different characteristics and containing both restricted and open content.

The National Archives of Estonia pilot aims to migrate a database with a complex structure and around 200.000 records.

The pilots demonstrate the potential benefits of these tools and how they can be used for easy and efficient access to archived records.

7. CONCLUSIONS AND FUTURE WORK

The Database Preservation Toolkit aims to support the migration of databases to the SIARD database preservation format and back to a DBMS. The SIARD format retains the database significant properties and its data can be validated using XML Schema Definition. Furthermore, by prioritizing the conversion, DBPTK ensures that no database content information is lost, and attempts to map the database structural and behavioural information to standard SQL, whilst keeping the original information as documentation. The software was made flexible to support different Database Management Systems and formats, including their specific features and optimizations. Moreover, DBPTK performs on low computing hardware requirements, even when converting databases containing millions of records.

The Database Visualization Toolkit aims to provide access to preserved databases, and achieves this by providing a fast and intuitive interface in which consumers can search and explore the preserved databases.

Both tools will be validated by the E-ARK pilots, by the European national Archives, ensuring that they are qualified to be used with real-world databases in a real archive environment.

Future work includes continuing the development of both tools, using means like the *roundtrip* tests and feedback from the E-ARK pilots to ensure top software quality and stability.

8. ACKNOWLEDGMENTS

This work is supported by the European Commission under FP7 CIP PSP grant agreement number 620998 – E-ARK.

9. REFERENCES

[1] ADDML - Archival Data Description Markup Language 8.2. Standard, Norwegian National Archives, Mar. 2011.

[2] I. Alföldi and I. Réthy. D2.3 - detailed pilots specification. Public deliverable, E-ARK, Mar. 2016.

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

[4] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.

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

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

[7] H. Heslop, D. Simon, and A. Wilson. *An Approach to the Preservation of Digital Records*. National Archives of Australia, Canberra, 2002.

[8] ISO 10646:2012, Information technology – Universal Coded Character Set (UCS). Standard, International Organization for Standardization, June 2012.

[9] ISO 19503:2016, Information technology – XML Metadata Interchange (XMI). Standard, International Organization for Standardization, Jan. 2016.

[10] ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times. Standard, International Organization for Standardization, 2004.

[11] ISO 9075:2008, Information technology – Database languages – SQL. Standard, International Organization for Standardization, 2008.

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

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

[14] RFC 1738, Uniform Resource Locators (URL). Standard, Internet Engineering Task Force (IETF), 1994.

[15] .ZIP File Format Specification, version 6.3.3. Standard, PKWARE Inc., Sept. 2012.