Database Preservation: The DBPreserve Approach

Arif Ur Rahman
and Muhammad Muzammal
Department of Computer Science
Bahria University, Islamabad
Pakistan

Gabriel David
and Cristina Ribeiro
Departamento de Engenharia Informática
Faculdade de Engenharia, Universidade do Porto
Rua Roberto Frias, Porto, Portugal

***Abstract*—In many institutions relational databases are used as a tool for managing information related to day to day activities. Institutions may be required to keep the information stored in relational databases accessible because of many reasons including legal requirements and institutional policies. However, the evolution in technology and change in users with the passage of time put the information stored in relational databases in danger. In the long term the information may become inaccessible when the operating system, database management system or the application software is not available any more or the contextual information not stored in the database may be lost thus affecting the authenticity and understandability of the information.**

This paper presents an approach for preserving relational databases for the long-term. The proposal involves migrating a relational database to a dimensional model which is simple to understand and easy to write queries against. Practical transformation rules are developed by carrying out multiple case studies. One of the case studies is presented as a running example in the paper. Systematic implementation of the rules ensures no loss of information in the process except for the unwanted details. The database preserved using the approach is converted to an open format but may be reloaded to a database management system in the long-term.

Keywords*—*Database Preservation, Transformation Rules

I. Introduction

There are many advantages of working digitally. Digitally stored data is easily accessible, manageable and helps in providing faster and better services to users than their paper based counterparts. However, with the benefits also come some trade-offs. For example the constant change in software and hardware technologies affect the data. This change may turn data unreadable as the old hardware, operating system and application software used for creating, storing and managing may not be supported by the latest technology. However, data may be relevant for the purpose of evidence of activities, institutional memory, scientific significance or historical significance and therefore preserving them may be required. Sometimes keeping organizational data is not a choice but it is mandated by law or by organizational policy. An example is the duration for which a university should keep the grades obtained by students in the courses they studied. This information is no longer required in day-to-day matters after a student completes his degree but it may be required as an evidence in the long-term.

Keeping in view the significance of data, digital preservation may be required for ensuring the constant availability of data over time. Selecting a preservation strategy, tools and formats for preserving data is a complicated task. Typically, decisions depend on the aims for given settings and institutional needs [5]. However, study shows that the selection of open formats is better for preservation than proprietary formats whenever possible [18]. Data stored in open formats may be easily accessible in the long term. New software may be developed for accessing data stored in open formats in case the existing software becomes obsolete. Moreover, widely used formats should be used for preservation as they raise the prospect that they will continue to be used for a long time [28]. In addition to this, formats for which a variety of writing and rendering tools are available should be given preference [20]. Metadata inherent to data and about the whole environment around them should be collected. It includes all the technical metadata like the software used for creation and preservation of data and non-technical metadata like the people involved in the creation of data and the reasons behind the creation. Another good practice may be to minimize the dependence of digital objects on users and software from the operational environment and preservation environment.

Different types of digital objects are created and managed by organizations for their operation including text documents, images, graphics, audio, videos, databases, websites and software. Different preservation approaches are required to preserve different types of digital objects as their nature and structure are different [4], [25]. For example emulation may be used as a preservation strategy if it is required to provide access to obsolete software but if the requirement is to provide access to data then migration may be used. The focus of this paper is on relational database preservation.

Software engineering good practices enforce the three layers approach to information systems design. The interface layer is accessible to users and get access to the data layer through the business rules layer. The data layer is implemented in a relational database. Relational databases are complex digital objects with a well-defined data structure. They are based on the formal foundations of relational modeling proposed by [9]. They organize collections of data items into formally-described tables. They are designed using the rules of normalization which is a step-by-step reversible process of replacing a given collection of relations by successive collections in which the relations have a progressively simpler and more regular structure [6], [16]. Normalization involves splitting large data tables into smaller and smaller tables, until reaching a point in which all functional dependencies among columns are

dependencies on the primary key of the corresponding table. Thus, preserving the uniqueness of the primary key ensures the uniqueness of the representation of the fact subject to the dependency. This helps databases to be consistent and efficient in capturing facts. However, because of normalization the model of a database may become too fragmented and difficult to understand.

The interface and business rules layers are implemented in code (triggers, functions, stored procedures and application). It is a fact that preserving code is an even harder issue than preserving data. Code is normally platform-dependent hence the requirement is to preserve the whole engine required to run it. Otherwise, there is a danger of losing the derived information which the code is able to generate. The preservation of code may refer to the generic software preservation problem which has been addressed in several ways e.g. emulation and technology preservation, but none of them makes it platform-independent [11], [19], [23]. The interface part dealing with presentation and interaction aspects is less relevant from an information preservation perspective. Part of the business rules deal with access control, compliance with organizational policy of the new transactions, and other operational aspects which are not very relevant as well. However, part of the business rules contain functions able to compute complex derived data which may be absent in the actual database. An example of this is a ranking function for scholarship granting.

Data stored in database systems are vulnerable to loss because it may become inaccessible and unreadable when the software needed to interpret them or the hardware on which that software runs becomes obsolete or are lost. Data may also become difficult to understand if the contextual information needed to interpret them is not known or is lost. Furthermore, if the structure of the information is too complex it may become a hard and time consuming task to query the information stored in a database. It is also possible to have wrong or missing information in operational systems which may lead to wrong results when queries will be made in the long term.

Different database systems require different solutions for solving the issues of complex structure, information embedded in code and missing or wrong information. For example a database system may have reached the end of its active life and is not used in day to day activities (retired databases). A database may retire in situations like an organization develops a new human resource management system using the latest technologies and the old one is not used anymore. Similarly, databases may retire if the activity for which they were developed comes to an end. Another kind of databases may be databases which are operational but a part of them is not used any more in day to day activities. For example once the fiscal year of an organization finishes, the data for that year may not be required in day to day activities.

The proposed solution works in cases where a retired database needs to be preserved or a snapshot of a database is taken and needs to be preserved. It follows the general framework of the Open Archival Information System reference model (OAIS) [7]. The producer sends a submission information package (SIP) containing the database and supporting information which may be helpful in the database preservation procedure. A digital preservation team preserves the database and produces the archival information package (AIP). The AIP is produced following the model migration approach which proposes to migrate a relational database to a dimensional model in the preservation procedure. Moreover, practical transformation rules are proposed which help in carrying out the model migration procedure. The AIP is stored for the long-term which may be accessed in the future using a simple to use and platform-independent tool. The proposed approach is a step further on the existing approaches for database preservation.

II. Related Work

Significant research has already been conducted for preserving relational databases for the long-term. The conclusions discard approaches like building technology museums for preserving specimens of machines, system software and applications, in all their main versions, so that the backups of every significant system could be used whenever required. Emulation is another approach which suggests simulating the old hardware or software in new machines [10], [13]. However it is not a permanent solution as technology changes very fast and writing new emulators is required whenever a change in technology occurs. More promising research suggests the conversion of a database into an open and neutral format with a significant amount of semantics associated, hence making it independent of the details of the actual DBMS. Such an approach is the software-independent archival of relational database approach presented in the sequel.

A. SIARD

The Swiss Federal Archives (SFA) proposed the Software-Independent Archival of Relational Databases (SIARD) for preserving relational databases for the future [12]. A software package known as SIARD Suite was developed which facilitates the conversion of a relational database to an open format known as the SIARD format [26]. It can convert a database originally in Oracle, Microsoft SQL Server, Microsoft Access and MySQL to the format. A database archived using the SIARD Suite needs to be loaded back to a DBMS to be able to query it. The SIARD format is based on open standards such as XML, SQL:1999, ZIP and UNICODE. It is able to record metadata at different levels including database, schema, table, column, trigger, routine, user, role and privilege. Most of the metadata is automatically retrieved from the data dictionary. However parts of metadata like the description of database and database objects, life span, owner and the people performing the preservation procedure need to be manually entered.

The DBPreserve Suite (presented in the sequel) is used to extend the metadata in a SIARD archive.

B. DBPreserve Suite

The DBPreserve Suite is a tool developed in Java for generating dimensional modeling metadata [2]. The metadata includes the names of stars and the relevant fact tables and dimensions. Moreover, it includes the metadata about the levels and hierarchies based on the levels in dimensions. The tool automatically extracts most of the metadata from the data dictionary of the DBMS. However, some information including the descriptions of stars, dimensions and dimension levels has to be manually added. The tool adds a new file with the dimensional modeling metadata in a SIARD Archive.

III. Model Migration Approach

A preservation procedure must be initiated when a database is approaching the end of its active life. The end of a database active life may be determined by the end of the reason for its creation but very often the case is of a replacement of the corresponding information system by a new one. Even in the latter case, though migration of the previous data may occur, it may be just partial. So, in all these cases a global appraisal of the database must be performed and consider the regulations in force, the probability of data loss, the data value in terms of evidence for the recorded facts, of scientific relevance and of organizational memory. This archival analysis must be accompanied by a technical and economic analysis on the feasibility of the preservation procedure.

Figure 1 presents the model migration approach for database preservation which involves the migration of a database to a dimensional model in the preservation procedure. The two issues to be considered in the migration procedure are the complexity of a relational model and the embedding in code of important knowledge from the application domain. The requirement of normalization does not remain in force once the active life of a database finishes and it becomes closed or when a snapshot is taken and a frozen database is to be preserved. The data will no longer be updated but it may be queried for purposes like verifying evidence of activities. This change of usage and a change in consumers bring a change of requirements. The need is now to preserve a database in a form that is easier for consumers to understand and write queries against. It means that it should be simple and it should give quick access. This can be achieved by migrating the database from a relational model to a dimensional model as suggested by Figure 1. The how-to issue will be dealt latter.

A. Dimensional Modeling

Dimensional modeling is a logical design technique that seeks to present data in a standard framework which is intuitive, allows for high-performance access and is resilient to change [1], [3], [27]. It is compatible with relational modeling and there is a straight forward mapping between them [8], [21]. It organizes data in tables of two natures, namely dimensions and fact tables. Dimensions store the detailed data about objects or entities. Furthermore, they can have levels and hierarchies which enable users to view data at various levels of detail. Fact tables store references to the set of relevant dimensions involved in a business process as well as the values representing real world facts [14], [15], [17]. A representation where a fact table is surrounded by dimensions involved in a business process is known as a star. Dimensions can be shared by different stars. A bus-matrix is constructed in the process of developing a dimensional modeling for a source relational model. It identifies business processes and the dimensions involved in them. Business processes are listed as matrix rows and dimensions are listed as matrix columns. The cells of the matrix are marked to represent which process includes which dimensions. Data transformations may be needed in the process of developing a dimensional model from a relational model as changes may occur in the structure, representation or content of data [22].

B. Migration Procedure

Figure 1 presents the database preservation procedure using the model migration approach. It can be seen that extraction, transformation and loading (ETL) is needed to migrate a database to its dimensional model. ETL is the process of taking out data from one or more operational systems (extract), modify them to fit into the dimensional model formatting needs (transform) and finally insert them into the dimensional model (load) [24]. In the migration process the relevant functions in the database are executed and the derived information are explicitly stored in new tables or in new columns in existing tables. This removes the dependency of a database on the underlying DBMS or computation environment and improves accessibility. The step of migrating the database to XML format is included to make it completely platform-independent. In this manner the main preservation concern of preserving the database content and the information implicit in business rules is accomplished.

The next section presents the transformation rules for carrying out the model migration procedure.

IV. Transformation Rules

To make the process of model migration efficient and traceable a set of transformation rules are proposed. These rules have three main purposes. The first is to design each component of the dimensional model from the systematic analysis of the relational model. The second is to perform a form of extraction, transformation and loading process on the data, with the care of keeping track of the original records. A final verification of the result against the original is important. The third purpose is to add enough contextual, content, technical, and provenance metadata to ensure understandability and authenticity of the database. The overall goal of these rules is to make sure that as much as possible of the information in a relational database is transferred to a dimensional model without any changes or loss. Thus, the rules help in carrying out the process in a way which ensures that the integrity of the database is not lost. Moreover, the rules ensure that it is still possible to query the information for the same purposes as it was possible with the source systems. Each step taken following the rules is documented. This helps to ensure the authenticity of the preserved database. The rules were obtained from carrying out case studies. One of them is used as a running example for presenting the transformation rules. The case study involved a database system which was used to manage all the information required by the institution about teachers as well other staff members. Oracle was used as the DBMS and an application developed in Visual Basic was used for interaction with the database.

A few years ago the old database system was replaced by a new state-of-the-art information system. The old system became frozen as the data stored in it was not updated anymore. However, the data is important and it may be required in the future as an evidence of past activities and institutional memory. Therefore, the database needs to be preserved for the long-term. A part of the model of the database is presented in Figure 2. It includes different tables including a table for the basic information of the personnel, their contracts and the contract categories. It also includes a table for the salary

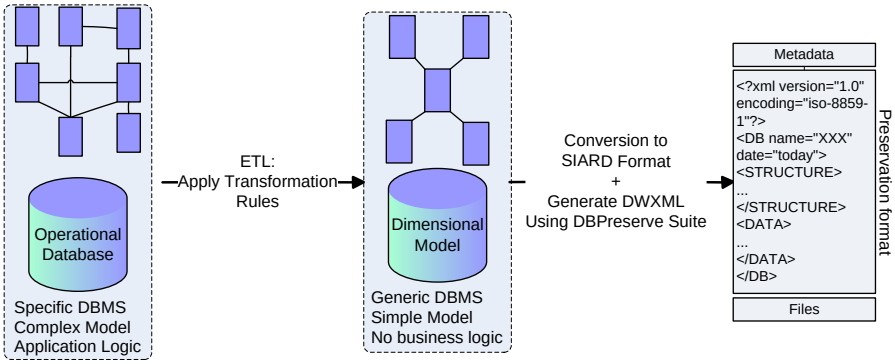

Fig. 1: Model Migration Approach for Database Preservation

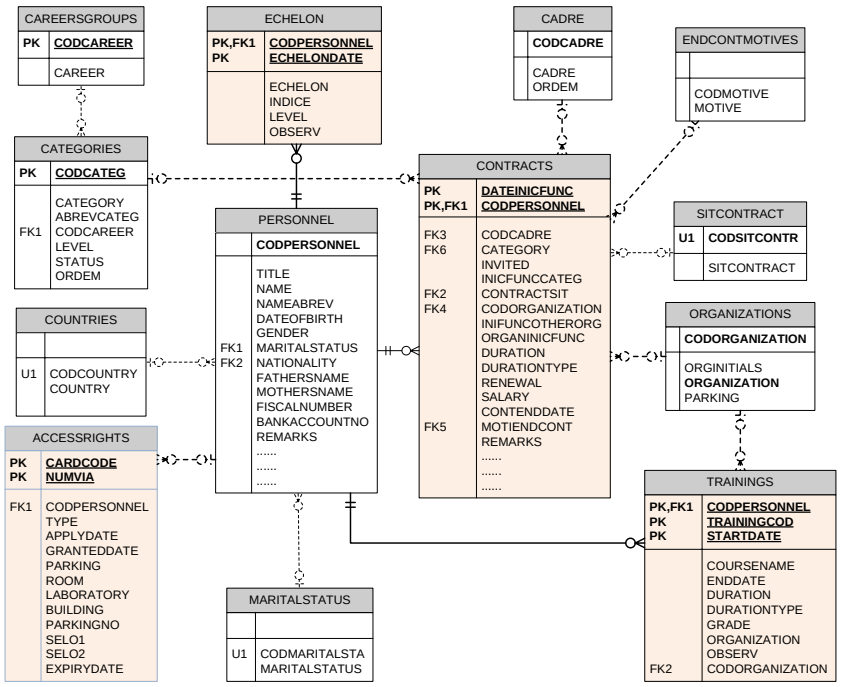

Fig. 2: A Part of the Model of the Human Resources Department Database

echelon and the motives of the end of contracts. The rules are grouped based on the phases in which they are implemented.

The rules are grouped based on the phases in which they are implemented.

1. **Preparation and Cleanup**

In this phase the source database tables are analyzed which includes writing a description of each one of them and identifying empty and temporary tables. The following rules are implanted in this phase.

i. Describe tables and columns: *A description of each table is prepared with name, meaning, source, number of rows, number of columns, date of creation, last update, number of LOB columns and constraints. In addition to this, information about each table column is collected including name, meaning, source, data type, number of distinct values, minimum and maximum value and number of nulls.*

An initial analysis of the case study database was done. It was found that there were 129 tables in the database, collectively having 999 columns. Following the transformation rules the descriptions of the tables were written. Statistics about each table were recorded including the number of rows, number of columns and number of distinct values in a column and maximum and minimum value in a column. The description of Contracts database table is shown at table level and at column level in Table I and Table II respectively. These descriptions are stored separately in two database tables in the staging area. Parts of the descriptive information were automatically retrieved from the data dictionary. However some information needed to be manually entered including the source and meaning.

ii. Remove empty tables and columns: *Empty tables may exist in a database because of various reasons e.g. an anticipated but unimplemented functionality. Such tables may be ignored after a thorough analysis. The analysis may include a check on the meaning of its columns and its relationships with other tables. Similarly, there may be columns in tables which are*

TABLE I: Table Description at Table Level

Table Name	CONTRACTS
Meaning	This table contains data about the contracts of employees. Information like the start and end date, salary, type, duration, renewable or not about the contract are included in the table.
Source	Data is added to the table each time a new employee is hired or his contract is renewed. There is a form for adding a new employee in the application software.
Number of Rows	6220
Number of Columns	36
Date of Creation	
Date of Last Update	
Number of LOB Columns	0
Comments	This table contains data about all the contracts of an employee. The `personnel` table also contains data about the current contract of an employee which seems repetition and may be ignored.

TABLE II: Table Description at Column Level

Column Name	Meaning	Source	Data Type	Number of Nulls	Distinct Values	Minimum Value	Maximum Value
codpessoal	Reference to personnel table	New employee addition form	char	0	2224	AA	ZMAV
organization	The organization where the employee works	....	char	1256	10	0032	6
codcadre	Cadre Code	....	char	4	13	1	9
category	Contract Category	....	char	1657	260	0	930
invited	If the employee is invited from another institution	....	char	5666	9	0	V
inicfunccateg	Start date in a category	....	date	5924	175	1969.08.22	2008.12.31
constractsit	....	....	char	6023	12	A	XXX
organinicfunc	....	....	date	5673	27	0	99016
duration	....	....	number	268	40	0	51
durationtype	....	....	char	3219	2	A	M
renewal	....	....	char	4247	5		S
motiendcont	....	....	char	629	20	0	22
remarks	....	....	char	4036	1998		ú

empty for all the rows. Such columns may also be ignored after an analysis.

The example database tables were analyzed. It was found that there were 26 empty tables and 67 columns with all nulls in the non-empty tables. After this step the empty tables and empty columns were ignored.

iii. Remove temporary tables: *Sometimes snapshots of tables are taken on a specific date and kept in a database. For example in the process of adding a new functionality in a database system, a snapshot of the existing tables involved may be taken as a backup in case of any problems. If documentation on the database exists, namely its data model, a check against it may help on the identification of temporary tables. Tables having similar names and having the same columns are candidates to be analyzed. If such a copy exists, it may be discarded and only the table which has the latest information is kept. The table which has the latest information may have a greater number of rows. Moreover, database logs may also be queried to see which copy of the table was updated last. The analysis may also include comparison of results of queries on both copies of a table.*

Temporary tables of two types were found in the example database. These tables were either storing data about a middle step in a process like recording data about all the applicants for a position before selection or they were snapshots of tables taken on a specific date. For example the Personnel table contains data about all the employees in the higher education institution. However, the Docdee table had data about teachers working in a specific department. This table should actually be a View as it presented a part of the Personnel table. Similarly, the Contests, Conteststea and Contestsntea tables were storing data about the contests for a specific position having 1, 16 and 1 row respectively. No reference to these tables was found in other tables. The low number of rows in the tables shows that they were added because of a newly added functionality in the system. However, the functionality was not fully implemented and it was not used. After a confirmation from the users of the system the temporary tables and the empty tables were ignored.

2. **Identify Keys**

In this phase the primary keys and foreign keys identified. The rules presented below are followed for their identification.

i. Identify primary keys: *It may be required to merge and split tables in the migration process. Typically, it is required to know the unique identifier in a table to carry out such operations. Moreover, if a table is a candidate to become a single dimension, it requires a primary key for linking with a fact table. Though it may not be always required, it is helpful to identify primary keys of tables before proceeding to any operations on the tables. There are different scenarios which may happen.*

(a) The primary key is implemented through database constraints and it can be identified just by looking at the constraints of a table.

(b) The primary key is not implemented through database constraints and there is a single candidate key. In this case there is no other choice, therefore the candidate key is declared as the primary key.

(c) There is no primary key implemented through database constraints and there are multiple candidate keys. In this situation, actual foreign keys in the other tables referencing candidate keys in the current table are searched for. The

primary key is chosen among the candidate keys involved in the external references. Otherwise, any candidate key may be considered as the primary key.
(d) There is no primary key and no candidate key in a table. In this situation a primary key column may be added and filled with numeric values.
The primary keys of the tables in the example database were identified. It was found that the primary keys conceptually existed but they were physically not implemented. For example there was a column in the Personnel table with the initials of the names of the employees. The number of distinct values in the column was equal to the number of rows in the table so it was an obvious candidate key. Furthermore, it turned out that this column was used in many other tables as a reference to this table. Therefore, this column was recorded as the primary key of the table. In addition to this, a check was performed on the Personnel table to see if it contains repeated records for persons with the same name. It was found that there were ten people with the same name. They were considered different records as the primary key values for these rows were different.
ii. Identify foreign keys: *All the tables are analyzed and any foreign keys are recorded. In some cases foreign keys are declared through database constraints. To identify the other, the fields of all the tables are analyzed and their meaning and contents studied to find out if they are references to other tables. If a field references any other table, it should be checked if it contains any orphan child records (OCR). There are three ways to resolve the situation if OCRs are found.*
(a) The OCR may be unwanted data e.g. inserted as a result of poor data validation applied prior to record insertion. In such situations the records may be ignored and deleted. However, any deletion is double checked in order not to damage the integrity of the database.
(b) The second way is to insert a parent record for each set of OCRs who needs the same parent record in the parent table. This way the problem is solved but it may create problems as the insertion of a new record may mislead users in the future. Users may consider the new entry a part of the data. For example the creation of an extra department may mislead users in the future to consider it a functional department. Therefore, when using this option it is ensured that it is not confusing for future users.
(c) The third way is to insert a single parent record for all OCRs meaning precisely unknown parent record. However, the issue with this choice is that the keys in the column need to be changed to the newly inserted record. This way the actual value inserted in the life of the database is lost.
In the next step the foreign keys in the tables were identified. It was found that relationships among the tables existed conceptually but in most cases they were not physically implemented. This had resulted in many data quality problems like orphan child records and illegal values in many tables. The foreign keys for the Contracts table are presented in Table III. For example in Contracts table the Codorganization column is a reference to the Organization table. It included values like xtCodand for 32 and 44 rows respectively. These are illegal values and are not in the domain. After an analysis of the application it was concluded that the application software had no validation in place before inserting a new record. Therefore, the system accepted whatever values were entered by users. To resolve the situation, records were inserted in the Organization table with the values to avoid losing the records in the loading process. A similar situation arouse in the Department column of the Personnel table. The Department column was a reference to the Department table. In the reference there were some records with a value iCC The value was correct but because of case difference OCRs were detected. The actual value in the parent table was ICC The value was updated to the same as in the parent table. However in the same table there were some employees with no department. Therefore a record was inserted in the parent table with as 'Unknown Department' with a code as D All these actions taken for removing the anomalies from the data were documented and kept for future reference. The PL/SQL code for these actions was also recorded in textual format. Table IV presents the preservation actions log. This log is also included in the AIP.

TABLE III: Foreign Keys in the `Personnel` table

Column	Referenced Table
codcadre	cadre
category	category
regime	regime
coddegree	degree
codorganization	organization
codstatus	status
codprogram	program
contracttype	contracttype
department	department
section	section
codteachertype	teachertype

3. **Normalization** In this phase the source database tables are analyzed to see if they are properly normalized or not. The rules presented below are followed in this phase.
i. Expand abbreviations: *Codes may have been used in the data to represent the status of something e.g. A - Approved, N - Not approved, C - Cancelled. Such codes may be replaced with meaningful words making them clearer and easily distinguished among them. Alternatively, the encoding may be recorded as metadata in the corresponding column of the preserved database. Information on the encoding may come from the application interface or the code generating it, from the database documentation or from domain experts.*
In some tables abbreviations were used, for example in the Contracts table 'Y' for year, 'M' for month and 'D' for day were used in the Durationtype column. The abbreviations were replaced with the full word.
ii. Normalize tables: *Each table is checked against the rules of relational modeling. Sometimes there are tables in a database that are not properly normalized. If a table is not properly normalized it may need to be split into multiple tables which may then become sources for different dimensions in the resulting dimensional model. In the normalization procedure, the extraction of columns into a new table goes along with the definition of its primary key and leaves a corresponding foreign key in the original table.*
In the example database, there were tables which needed to be merged. For example the Personnel table was storing basic data about employees as well as some data about the current contracts. It was needed to verify if the information about the current contract of an employee is also stored in the Contracts table. Queries were made on both tables and the results were compared. It came out that Contracts table stored information about all the contracts including any past contracts and the current one. Another table Personnelinfo was storing

TABLE IV: Preservation Actions Log

SNo	Action Description	Rule	Code
1	Assigning foreign key to `personnelinfo` (Column: `maritalstatus`) referencing `maritalstatus` table.	2	`alter table personnelinfo add constraint fk_persoO_data_maritstat foreign key (maritalstatus) references maritalstatus (codmaritalstatus);`
2	Some persons are without department in the `personnel` table (`department` is null), so I inserted a row in `department` table	2	`insert into department (sigla, nomedep) values('UD','Unknown Department');`
3	Rows in the personnel table with department = null, updated with department = UD (190 rows)	2	`update personnel set department='UD' where department is null;`

detailed data about personnel like names of parents, date of birth, marital status, address, country of origin, bank account information and so on. A merge was performed but in the Personnelinfo table there were records for which there was no matching record in the Personnel table. In order not to lose information, the records which did not fulfil the joining condition were manually inserted in the resulting merged table. Furthermore, the columns which were storing information about the current contract of an employee in the Personnel table were ignored as this information was recorded in the Contracts table.

4. **Define Stars**

In this phase the tables are clustered, the dimensions are identified and then the bus matrix is defined. The rules presented below are followed in this phase.

i. Cluster tables: *All the tables are clustered considering their foreign keys and under the broader context of the relevant processes in the organization. The organizational processes about which the system stores data are found out. A discussion with the users of the system may be helpful. Moreover, looking at the application program and understanding the way it works is also important.*

For each organizational process there is a set of tables in the model which stores related data. For all the processes the set of participating tables are listed. In addition to this, in each set of tables there is normally a central table which is identified. This table normally has larger number of rows than the tables it references. Usually, it has no or a small number of incoming references but it references other tables. The central tables have the real world facts recorded in the organizational processes. They may be candidates to be loaded into fact tables and become the centers of stars. In the end all the tables must be included in one or more stars, otherwise new stars are defined to encompass the remaining tables. To help in the clustering task, the following guidelines are provided, to be applied to the set of all tables after the normalization steps.

- *An isolated table is a cluster and it is its central table.*
- *When a table has no incoming references but it references at least one table, it becomes the central table of a cluster and the tables it references are recursively included in the cluster.*
- *When a table, though having incoming references, is referencing at least two other tables and has meaning, in the context of an organizational process, as a fact recording table at some level of granularity, it becomes the central table of a cluster and the tables it references are recursively included in the cluster.*

The union of clusters must contain all the tables. A table may belong to more than one cluster. Sub-clusters may be contained in larger clusters.

The following organizational processes were identified in the part of the model of human resources department database presented in Figure 2.

- Contracts: This process is about the contract of an employee. An employee is hired by an organization, the contract is according to a cadre and it may be for a certain period with an explicit termination motive.
- Echelon: This process manages data about the salary echelon of an employee. An employee is assigned an echelon which is further divided into different levels.
- Access Rights: This process manages the access rights of an employee. For example access to different rooms, buildings and car parking of the institution.
- Trainings: This process manages the trainings which an employee takes. The trainings may take place in a specific organization.

Now keeping in view the organizational processes, the tables Contracts, Echelon, Trainings and AccessRights seem to be the central tables of clusters. Starting from the Contracts table, it can be seen that it references tables, namely Personnel, Categories, Cadre, Endcontmotives, Sitcontrat and Organizations. It can be seen that the Personnel table references tables, namely Countries and MaritalStatus. Furthermore, the Categories table references the CareerGroups table. The tables referenced by the Personnel table and the Categories table are included in the Contracts cluster.

The Echelon and AccessRights tables references the Personnel table. Therefore, they become the central tables of two different clusters. Both reference the Personnel table. Therefore the Personnel table and the tables referenced by it are included in the clusters. The Trainings table references two tables, namely Personnel and Organizations. Therefore, the Trainings table becomes the center of another cluster. The central table references the Organizations table, the Personnel table. Therefore, the Organizations table, the Personnel table and the tables referenced by it are included in the cluster. The Table V shows the result of this step.

Though the tables Personnel and Categories reference other tables, they are not considered central tables of a cluster as they are not fact recording tables about an organizational process. Furthermore, it can be seen that all the tables in the model belong at least to one cluster. Therefore, no table is left in the process.

ii. Identify dimensions: *Each cluster of tables is analyzed to identify dimensions in the future stars. Moreover, levels and hierarchies in each dimension are identified. A single dimension may be a target for multiple source tables. The identification is best guided by the knowledge of the organizational processes and of their main entities.*

The following analysis should be applied to all the clusters, starting with the smaller ones. In each cluster analyze each of the foreign keys out of the base table. If the referenced table is

TABLE V: Table Clustering

Clusters \ Tables	Categories	CareerGroups	Countries	**AccessRights**	Personnel	MaritalStatus	**Echelon**	**Contracts**	Cadre	**Trainings**	Organizations	SitContract	EndContMotives
Contracts	X	X	X		X	X		X	X		X	X	X
Echelon			X		X	X	X						
Access Rights			X	X	X	X							
Trainings			X		X	X					X		

not part of a dimension or a sub cluster, build a dimension starting with that table as the lowest level and the tables it references, on the next level, recursively. The hierarchy is defined by the sequence of references. If the referenced table is the base table of a sub-cluster, build a dimension with just the non-foreign key attributes. The foreign key attributes may be added to the base table of the main cluster if considered relevant to understand it.

In the example consider the Contracts cluster. The central table i.e. Conracts, has several keys. Following FK1 the Personnel table becomes the lowest level in a dimension. Following FK1 of the Personnel table, the maritalStatus table becomes the next level in the dimension. Similarly, following the FK2, the Country table becomes another level in the dimension. Following the FK2, FK3, FK4 and FK5 of the central table, the SitContract, Cadre, Organizations and EndContMotives tables respectively become single level dimensions. Following FK6, the Categories table become another dimension. The FK1 of the Categories table lead to the creation of another level in the dimension by adding the CareerGroups table. Now considering the AccessRights cluster, the AccessRights table is identified as the central table of the cluster. At first look it may seem that the columns, namely Room, Laboraory and Building may become a separate dimension which levels including Rooms and Buildings. However, an analysis of the data shows that these columns have only two distinct values i.e. '0' and '1'. Therefore, they are kept in the central table without creating a dimension. The FK1 of the central table is a reference to the Personnel table. The Personnel dimension is already identified in the previous cluster. Therefore, the same dimension is included in this star. Similarly the Echelon cluster has the central table Echelon which references the Personnel table using its FK1. The Personnel dimension is already identified so the same dimension can be shared by this star. The Trainings table is the central table of the Trainings cluster. The central table seems to have columns for the course an employee takes, namely Codcourse and Coursename. These columns are separated and a separate dimension is created, namely Courses. The central table references the Organizations table using FK2. Though a column for the name of the organization exists in the table, it is better to use the Organization dimension.

iii. Define the bus matrix: *Decide whether all the clusters will be converted into stars. For each cluster, the central table becomes the fact table in the corresponding star and the dimensions identified in the previous step are also included in the star. Once all the dimensions and fact tables are identified a bus-matrix is constructed. The bus-matrix makes it clear which dimensions are part of which star. In some cases, the connection between a certain fact table and a dimension may require a bridge table of some sort. The source tables in the original model for the bridge tables are also identified.*

It is necessary to check whether all the columns excluding the ones which are deliberately left-out because of a known reason, belong to at least one map in the end of this mapping process.

The bus matrix for the example database is shown in Table VI. It can be seen that four stars are built. The Personnel dimension is shared by all the stars where as the Organizations dimension is shared by two stars, namely Contracts FT and Trainings FT.

TABLE VI: Bus Matrix

FT \ Dim	*Personnel*	*SitContract*	*EndContMotives*	*Organizations*	*Categories*	*Cadre*	*Courses*
Contracts_FT	X	X	X	X	X	X	
AccessRights_FT	X						
Echelon_FT	X						
Trainings_FT	X			X			X

5. Implement Dimensions

In this phase the dimensions are implemented. In some cases it may be needed to use degenerate dimensions. The rules presented below are followed for implementing dimensions.

i. **Implement dimensions:** *The migrated model should be made of simple stars, to be easy to query. One technique to achieve this is to de-normalize the dimensions, including in each one of them simple or multiple hierarchies. This corresponds to merging tables in the original model, but keeping the primary key for each level as a level key. There may be situations in which some columns from one table may need to be moved to another table.*

While merging tables, situations may arise where some records in one table mismatch records in other tables. Such situations are carefully verified not to lose data and the records may be manually inserted in the resulting merged table if required.

The identified dimensions were implemented. This required joining and splitting tables. The Personnel dimension required joining three tables, namely Personnel, Countries and MaritalStatus. These tables were joined and level keys were added. Similarly, for the creation Categories dimension the Categories and CareerGroups were joined. The creation of Courses dimension required splitting the Trainings table. The columns related to the course taken, namely CodCourse and CourseName were added to the Courses dimension. The other dimensions did not require splitting or merging tables and they became single level dimensions. The resulting model

included four stars, namely Contracts, Echelon, Trainings and AccessRights. The Contracts star is shown in Figure 3.

ii. Include degenerate dimensions: *There may be columns storing descriptive information e.g. 'remarks'. Such a column may be added to a dimension if the values are solely dependent on its key. However, it may be added as a degenerate dimension value in the fact table if it depends on a combination of dimensions.*

In the Contracts star fact table there column remarks could become a dimension. However, the number of distinct values in the column was almost equal to the number of rows in the fact table; it was decided to keep it in the fact table.

6. **Implement Fact Tables**

In this phase the fact tables are implemented. They may require the use of a star schema or a snowflake schema. The rules presented below are followed in this phase.

i. Use snowflake schema: There may be situations where a set of tables in the operational system may need to be joined for constructing a dimension and one of them is a lookup table with more records than actually used by the lower level in the hierarchy. In such situations a snowflake schema is constructed which helps in not losing the higher level rows.

ii. Handle nulls: *Nulls are abnormal values in a column in the sense that, in some situations, they lead to unexpected results and they require specific operations. When they appear in a column that is a foreign key, the corresponding line is stripped from inner joins with the referenced table. One solution is to use outer joins. Another one is to add a line for the nknownor ot applicablevalue in the referenced table and update the null values in the foreign key column to become the corresponding key.*

There were some records in central tables of the clusters with a null in the foreign key columns. For example the Sitcontract column in the Contracts table (Figure 2) had some lines with nulls. Therefore, a row was inserted in the SitContract table with CODSITCONTR = 0 and SITCONTRACT = 'Unknown' and the records in the Contracts table were pointed to this record.

iii. Convert the records: *Before loading data to the target dimensional model the stars may be compared among them and it may be checked if two stars are candidates to be combined. They may be candidates to be combined if they share the same dimensions, record information at the same level of granularity and the timing and nature of events is the same. Data may be loaded to the target dimensional model once the check is performed.*

In the example database there were no similar stars. Therefore, the dimensions were loaded from the source tables.

7. **Code**

In this phase the application software or code in other forms including functions, stored procedures and triggers are dealt with. The rules presented below are followed in this phase.

i. Analyse application forms: *The application program may have forms for adding new data, displaying the data already in the system and generating reports. The forms are analyzed and the functionality is recorded. This may be used to verify that the resulting dimensional model is able to answer the queries used in the forms for retrieving information from the database. Furthermore, screen shots of the forms are taken and kept for future reference.*

Associating different menu items of the application with the database tables may be helpful in understanding the functionality of the form in the future.

Screen shots of the application software were analyzed. The forms in the application software for inserting new data did not have validation in place. This was one of the reasons of orphan child records in some tables.

ii. Make short descriptions of algorithms (code) *If there are functions, stored procedures or code in any form to derive information from the data stored in a database, they are executed and the results are explicitly stored. In addition to this, a description of each piece of code explaining it and the information it produces is written and kept in the preserved database.*

In the application software there was no such code which generated derived data not explicitly stored in the tables. However, there were triggers to show notifications about the last date of a contract on the contract end date. Description of the triggers were written and kept for future reference.

8. **Metadata and Verification**

Metadata is recorded in the migration procedure. In this phase the results are verified using some of the metadata collected in different phases. The rules presented below are followed for collection of metadata and verification of the resulting model and the data.

i. Record metadata: *There is a set of metadata elements that should be recorded along with the migration procedure, at several stages. These elements include reference, contextual, technical, and provenance metadata at several levels like global, organizational process level, star, dimension and column levels. Even the mappings which are the basis for the ETL process from the source database to the target dimensional model are kept for future reference. The mappings record the origin of data in the dimensional model and document the whole process. They may be used to facilitate any verification procedure.*

The recorded metadata was initially kept in the staging area in database tables. It was moved to the dimensional model schema once the migration process completed.

ii. Verify the result: *In the end of the process a complete verification of the number of rows in each table against the original database must be performed, in order guarantee that no relevant information has been lost. Notice that in rule 8 a completeness check is performed at the level of tables, in rule 9 at the level of columns, and in rule 11 at the level of records, so no data is lost in the migration procedure, except for the assumed irrelevant or empty tables and columns.*

In the end of the migration procedure, it was verified that all the data from the source system was transferred to the dimensional model. For this purpose the table clustering done following rule 8 was checked once again. Moreover, it was verified that the identified dimensions contained all the columns of all the tables. For this purpose the bus matrix constructed following rule 10 was reviewed. Furthermore, it was also verified that the stars contained the relevant dimensions.

The model migrated database becomes free from the operational environment. It has a simple model which is easy to understand. Moreover, the metadata gathered in the migration procedure help in assessing the authenticity of the database. After the model migration procedure the Archival Information package is prepared and kept for the long-term.

The complete number of the transformation rules is high but

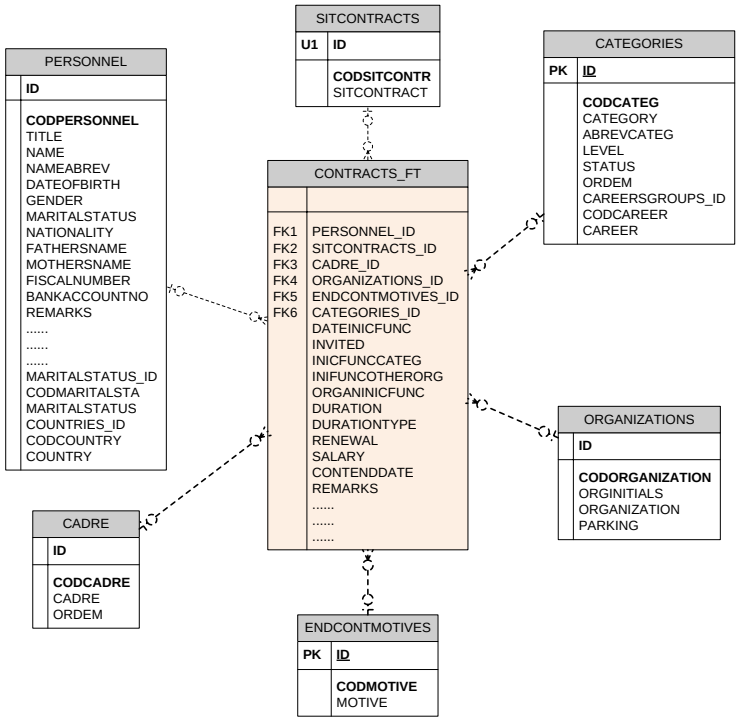

Fig. 3: Contracts Star

in many processes not all of them will be actually requiring effort. The implementation of some of the rules is immediate if the original database to be preserved is well designed and with no junk. For example, the identification of keys (Group II) will be very quick if they are implemented through database constraints and normalization (Group III) may not be required. The best opportunity to preserve a database is during its deactivation but before the system is dismantled. If the system is still able to run, several steps of the procedure become easier. If the DBMS is no longer available, or the platform to run the application, or the know-how on its use, it may become hard to restore the backups and establish the environment for the migration to take place.

V. Archival Information Package

The AIP contains all the information considered relevant for future use. It contains the following items.

- The model migrated database is converted to SIARD format for making it completely platform-independent and then it is included in the AIP. It includes the data values with a simple model as well as some contextual metadata. SIARD Suite allows for the addition of table descriptions, column descriptions, the name and contact details of the contact person, the owner of the database and so on.
- The dimensional modeling metadata is generated using the DBPreserve Suite. The DBPreserve Suite automatically adds the metadata to the SIARD Archive.
- The preservation log containing the record of all the steps taken in the database preservation procedure. It includes the steps taken according to the transformation rules. The log is stored in a text file.
- Screen shots of the application forms used to add new data to the database are included in the AIP. A text file explaining the application forms is included in the AIP.
- The schema of the original database and the model migrated database is also included in the AIP.

The structure of the AIP is based on the structure of the SIARD archive. It is shown in Figure 4.

VI. Database Dissemination

A database preserved using the proposed approach can be accessed in two ways. Firstly, the database is loaded to a DBMS using the SIARD Suite. This will require users to have the knowledge of a query language e.g. SQL. Furthermore, it will also require a DBMS installed on a machine. The other choice is the use of the DBPreserve Archive Browser (DAB). The DAB is a browser developed using Java. The use of DAB may avoid the need to load back a preserved database to a DBMS in many use cases. A user needs to select a table from one combo box thus making it the current focus (base table). The names of all the tables referenced by the base table are displayed in another combo box. Users may then open one of the tables to browse its contents. The browser provides different browsing functionalities including the following.

- **Table Filtering:** A user may filter a table by searching a string in the entire table. Moreover, the search may be made limited to a single column. The filtering is

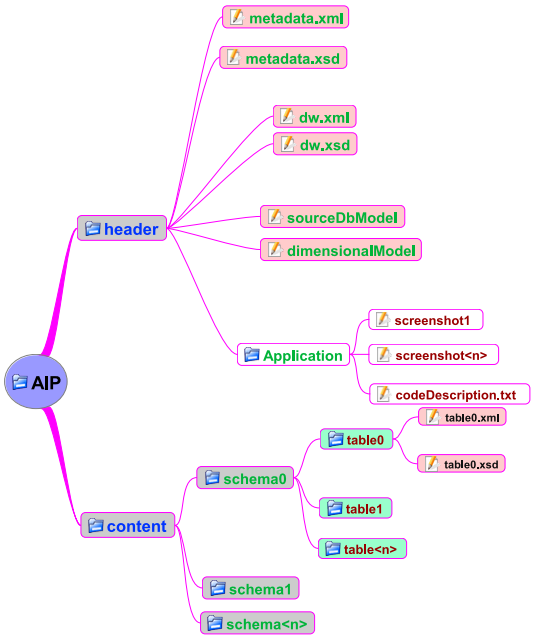

Fig. 4: Archival Information Package

done regardless of the case (upper case and lower case) of the string.

- **Following Keys:** The browser also allows navigating from one table to another table using keys. If a user clicks on a value in a foreign key column, the corresponding row in the parent table is displayed. Similarly, clicking a value in the primary key column leads to all the corresponding values in the selected base table
- **Table Joins:** The browser also allows joining the base table with the tables it references (if any).

VII. Evaluation

Evaluation of digital preservation procedures and the result is a complicated task. The evaluation of a database preservation procedure is even more difficult as there are no studies available identifying the parameters for the evaluation. It is hard to anticipate the tools available in the future and technical expertise of the people who will be using the preserved database. Moreover, the usage of a preserved database may be different from the anticipated usage during the preservation procedure. However, some points related to any type of digital objects and procedures can be verified including the use of open formats, addition of metadata and independence from the operational environment.

The proposed approach uses the SIARD format as the archival format for a database. SIARD format is an open format used by many libraries and archives around the world for archiving databases. It allows adding some technical and non-technical metadata. However, the format does not allow to add dimensional modeling metadata. Therefore, the DWXML was developed which is also openly available. The DWXML is added to the archive using the DBPreserve Suite. Moreover, all the steps taken for the migration of a database are documented and kept for future reference. This is helpful in controlling the loss of information and ensuring the authenticity of the database.

The structure of the database is simplified and can be easily understood by users in the long term. The simplicity of the structure also eases the development of new software for dissemination of a database preserved using the proposed approach. The DAB is an example which may be helpful in many use cases for querying a database without the need of a DBMS.

The database is not dependent on the operational environment anymore. The application software used in the active life of a database is not required in the long term to query the database. In conclusion the simplification of database structure, usage of open formats, the collection of metadata, minimizing the dependence on tools from the operational environment and preservation environment is a step further in the current state-of-the-art solutions for database preservation.

VIII. Conclusion

Database preservation is a complex process that needs to be carried out in a step by step and traceable manner. The loss of a single piece of information in the process may make the authenticity of a preserved database questionable. The proposed transformation rules for database preservation are implemented in a systematic manner which guarantee that no loss of information occurs in the procedure except for the irrelevant and unwanted details. The resulting preserved database has a different model from the original one. However, the model of the preserved database is simple to understand and easy to write queries against for people who did not develop it or used it in its active life.

The implementation of the transformation rules shares some intuitions and techniques with traditional data warehouse systems. However the ultimate goal of preserving a database is very different from the usual goal of building a decision-support system. This has a main consequence in the nature of the fact tables, which often lack clear measures or the measures included are just secondary elements.

The model migrated database is converted to the SIARD format which makes it completely platform independent. The proposal includes the use of a platform-independent and easy to use tool for dissemination of a preserved database. The tool does not require users to have knowledge of a query language thus avoiding the need of an expert for disseminating a preserved database. The use of the tool also avoids the need of installing and configuring a DBMS for loading a preserved database to it.

References

[1] R. Agrawal, A. Gupta, and S. Sarawagi, "Modeling multidimensional databases," in *Proceedings of the Thirteenth International Conference on Data Engineering*, ser. ICDE '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 232–243. [Online]. Available: http://portal.acm.org/citation.cfm?id=645482.653299

[2] C. Aldeias, G. David, and C. Ribeiro, "DWXML - A Preservation Format for Data Warehouses," in *XML: Aplicações e Tecnologias Associadas*, Vila do Conde, Portugal, June 2011.

[3] C. Ballard, D. M. Farrell, A. Gupta, C. Mazuela, and S. Vohnik, *Dimensional Modeling: In a Business Intelligence Environment*. Vervante, 2006.

[4] C. Becker and A. Rauber, "Decision criteria in digital preservation: What to measure and how," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 6, pp. 1009–1028, June 2011. [Online]. Available: http://dx.doi.org/10.1002/asi.21527

[5] D. Burda and F. Teuteberg, "Sustaining accessibility of information through digital preservation: A literature review," *J. Inf. Sci.*, vol. 39, no. 4, pp. 442–458, Aug. 2013. [Online]. Available: http://dx.doi.org/10.1177/0165551513480107

[6] E. F. Codd, "Normalized data base structure: a brief tutorial," in *Proceedings of the 1971 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control*, ser. SIGFIDET '71. New York, NY, USA: ACM, 1971, pp. 1–17. [Online]. Available: http://doi.acm.org/10.1145/1734714.1734716

[7] *Reference Model for an Open Archival Information System (OAIS)*, Consultative Committee for Space Data Systems (CCSDS) Std. ISO 14 721:2003, Rev. Magenta Book, Recommended Practice, Issue 2, June 2012. [Online]. Available: http://public.ccsds.org/publications/archive/650x0b1.pdf

[8] Daniel L. Moody and Mark A. R. Kortink, "From ER models to dimensional models part II: Advanced design issues," *Journal of Business Intelligence*, 2003.

[9] Edgar Frank Codd, "A relational model of data for large shared data banks," *Communications of the ACM - Special 25th Anniversary Issue*, vol. 13, no. 6, pp. 377–387, June 1970. [Online]. Available: http://doi.acm.org/10.1145/362384.362685

[10] S. Granger, "Emulation as a digital preservation strategy," Tech. Rep., 2000.

[11] M. Guttenbrunner and A. Rauber, "A measurement framework for evaluating emulators for digital preservation," *ACM Trans. Inf. Syst.*, vol. 30, no. 2, pp. 14:1–14:28, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2180868.2180876

[12] S. Heuscher, S. Järmann, P. Keller-Marxer, and F. Möhle, "Providing authentic long-term archival access to complex relational data," in *Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data*. European Space Agency, 2004.

[13] J. V. D. Hoeven and H. V. Wijngaarden, "Modular emulation as a long-term preservation strategy for digital objects," in *5th International Web Archiving Workshop*, 2005.

[14] C. Imhoff, N. Galemmo, and J. G. Geiger, *Mastering Data Warehouse Design: Relational and Dimensional Techniques*, R. Elliott, Ed. Joe Wikert, 2003.

[15] W. H. Inmon, *Building the data warehouse*, 4th ed. Wellesley, MA, USA: QED Information Sciences, Inc., 2005.

[16] W. Kent, "A simple guide to five normal forms in relational database theory," *Communications of the ACM*, vol. 26, no. 2, pp. 120–125, February 1983. [Online]. Available: http://doi.acm.org/10.1145/358024.358054

[17] R. Kimball, L. Reeves, W. Thornthwaite, M. Ross, and W. Thornwaite, *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

[18] R. Latham, "Digital preservation formats," Tasmanian Archive and Heritage Office, Tech. Rep., 2012.

[19] B. Matthews, B. McIlwrath, D. Giaretta, and E. Conway, "The significant properties of software - a study," Joint Information Systems Committee (JISC), Tech. Rep., December 2008.

[20] E. P. Mclellan, "Selecting formats for digital preservation: Lessons learned during the archivematica project," *Information Standards Quarterly*, vol. 22, no. 2, pp. 30–33, 2010.

[21] D. L. Moody and M. A. R. Kortink, "From ER models to dimensional models: Bridging the gap between OLTP and OLAP design, part I," *Journal of Business Intelligence*, 2003.

[22] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Engineering Bulletin*, vol. 23, no. 4, pp. 3–13, 2000.

[23] J. Rothenberg, "Avoiding technological quicksand: Finding a viable technical foundation for digital preservation," Council on Library and Information Resources, Washington DC, Tech. Rep., January 1999.

[24] A. Simitsis, P. Vassiliadis, and T. Sellis, "State-space optimization of ETL workflows," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 17, no. 10, pp. 1404 – 1419, OCTOBER 2005.

[25] S. Strodl, C. Becker, R. Neumayer, and A. Rauber, "How to choose a digital preservation strategy: evaluating a preservation planning procedure," in *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, ser. JCDL '07. New York, NY, USA: ACM, 2007, pp. 29–38. [Online]. Available: http://doi.acm.org/10.1145/1255175.1255181

[26] H. Thomas, *SIARD Suite Manual*, Swiss Federal Archives, SFA, Archivstrasse 24, 3003 Bern, Switzerland, May 2009.

[27] R. Torlone, "Multidimensional databases," M. Rafanelli, Ed. Hershey, PA, USA: IGI Publishing, 2003, ch. Conceptual multidimensional models, pp. 69–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=887433.887438

[28] C. Webb, *Guidelines for the Preservation of Digital Heritage*, Information Society Division, United Nations Educational, Scientific and Cultural Organization Std., March 2003.