Preserving Queries within Relational Databases

By:
Somaia Alaofe
2139515
Alao0003@flinders.edu.au

College of Science and Engineering

Supervisor: Dr Denise de Vries

March 2018

“Submitted to the School of Computer Science, Engineering, and Mathematics in the Faculty of Science and Engineering in partial fulfilment of the requirements for the degree of master of computer science at Flinders University – Adelaide Australia.”

DECLARATION

"I certify that this work does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text."

Signature	Date
Somaia Alaofe	09/03/2018

ABSTRACT

Most scientific research is stored in digital formats; however, with technology obsolescence, important resources for future research can be lost. Relational databases (RDBs), including component tables, relationships, and queries, are an important digital format for preserving research results. Therefore, preserving databases for a long time, regardless of any software or hardware issues, is critical. Previous studies have developed database preservation but have focused on preserving data, relationships, and keys rather than reports, pages, and queries within a database.

Therefore, this study proposes a strategy for long-term digital preservation of queries within RDBs because queries are a major component of providing accurate, relevant data that lead to better insights, better decision making, and improved business. This research surveyed literature on RDB preservation and identified a gap in query preservation. There are a range of challenges in place regarding the variation of manipulation languages between RDBs, and four database management systems were surveyed to determine how queries can be manipulated on each platform.

The proposed solution analyses relational algebra trees to preserve queries and researches eXtensible Markup Language (XML) to present relational algebra trees in XML format. Therefore, this research suggests an optimal approach to query preservation in various RDB platforms. A preservation methodology for queries is proposed based on converting all data manipulation language representation among various RDBs into one format: an XML file. This XML file does not preserve the data manipulation language itself but rather the relational algebra trees of queries. Thus, the logic of queries will be preserved in both human and computer readable formats.

ACKNOWLEDGMENTS

I would like to take this opportunity to first and foremost thank God for being my strength and guide in the writing of this thesis.

Also, I would like to thank my husband Badr and my parents for their continuous support and encouragement throughout my research. This achievement would not have been possible without them.

I would like to express my sincere gratitude to my advisor Dr Denis De Vries for her continuous support of my thesis, as well as her patience, motivation, enthusiasm and immense knowledge. Also, I thank Dr Carl Mooney who has supported my thesis through the database sample that he has provided me with.

Finally, I would like to thank my wonderful children, Leen and Hassan, for always making me smile and for understanding those times when I had to work on this thesis instead of spending time with them. I hope I have been a good mother and that I have not lost too much during the tenure of my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE: INTRODUCTION

The recent information boom has caused an information revolution, creating a heritage of data in various formats, such as media, links, and images, and a cognitive need to preserve these data in structured digital collections (Woods, 2010). This requires a type of program known as a database management system (DBMS) to manage storage, access, and retrieval of data (Freitas and Ramalho, 2010). According to data utilisation and retrieval statistics, only 15% of data are accessed daily by users, while remaining data are archived. Long-term preservation has been developed for digital dataset management as a result of evolving technologies for which long-term preservation ensures long-term access to data (Schaefer et al., 2016). Thus, the data lifecycle can be categorised into the following stages (Lindley, 2013).

Active state: Data are actively generated and modified in a system.

Archive state: Data are easily accessed and stored in such a way to ease processing without alteration (i.e., data cannot be edited, only retrieved and processed).

Long-term archiving: Only some parts of a dataset are preserved for a specified, long-term period.

1.1 Background

Data availability over an indefinite time period requires long-term preservation, which includes tools, strategies, and processes that protect the availability and accessibility of data in a readable state. Here, indefinite refers to a time period that covers technology and data obsolescence due to changes in technology that interrupt management of digital objects. Because less is known about digital objects compared to physical objects, digital objects require continual maintenance and refined support systems (National Library of Scotland, 2014).

Long-term preservation of digital data also requires novel security, data integrity measures, and satisfying long-term preservation. Security mitigates software failure and unauthorised data access due to software evolution (Seoane, 2014). Data integrity threats include storage media degradation or data format and software architecture annihilation caused by data formats only being supported by specific software or operating systems (Kremser et al., 2012). It is difficult to ensure satisfactory long-term preservation because success depends on future testing and requires extensive research efforts and innovations (Gorsel et al., 2014).

A database describes information and related data based on categories and is conveniently and efficiently managed, maintained, and accessed using a DBMS, which defines storage structures, manipulates and interrupts strategies, and ensures data security (Date, 2014). A DBMS can be accessed during the data creation phase by restarting applications or using independent software for information access. A relational database management system (RDBMS) provides an extra condition relevant to creating and defining relationships between tables, namely tabular structures. The distinguishing feature between a DBMS and a RDBMS is that the former does not support tabular structures and does not impose relational creation (Sabău, 2007). A relationship in this context is defined as an association between entities and is a mathematically formal term that represents logical relationships among entities or things; therefore, a relational model is a description of data tables and their relationships (Date, 2014).

1.2 Research Problem

Long-term preservation is a topic that first emerged at the beginning of 2000 and that changed the focus of research and scholarly concerns. Cumulative research efforts defined various aspects of long-term preservation, such as strategies, utilisation, limitations, costs, legalities, preserving data and metadata, and preservation as policy (Backus et al., 2016). Research to develop an efficient method for preserving accurate representations of entire databases, which include data, metadata, structures, relations, and data keys, is ongoing. No research has addressed preservation of queries as an optimisation for data storage, data genre preservation, access capabilities, or (most importantly) execution plans. Therefore, this research addresses queries as a database optimisation option for long-term data preservation.

1.3 Research Objective

The primary objective of this research is to formulate a proposal to preserve database queries as a long-term preservation model. Secondary objectives include reviewing the properties of four common relational databases (RDBs), converting query languages in relational algebra structures, which will provide the optimum execution plan providing the lowest execution plans' cost chosen from set of alternative plans for same query, and proposing XML as a preservation strategy for queries.

1.4 Research Questions

The research question is: what is the proposed model for long-term preservation of queries? Additional questions include: what are the properties implemented in four common RDBs; what is the structure of database queries; and how can XML be used as a preservation strategy for queries?

1.5 Research Significance

This research considers a form of query preservation that simplifies data retrieval, handles query creation and complexity among various databases, and takes into consideration the fact that few database users explore database documentation or tutorials for help. Such preservation increases database utilisation and the consistency of database access, regardless of an individual database’s query structure or format. This will benefit the referential integrity of databases, independent of the degree to which it supports Structured Query Language (SQL), and will create opportunities for further research of query preservation strategies, benefits, and rules.

1.6 Research Methodology

This research utilised both descriptive and experimental methodologies that divide the research into two phase: descriptive and experimental. The descriptive methodology is based on a survey of RDBs, while the experimental methodology is based on a proposed model for preserving queries in XML. The research sample in the first phase includes four common RDBs to which the preservation property was applied: MS Access, MySQL, MS SQL Server, and Oracle. The experimental phase included four common RDBMS: MSs Access Database, MS-SQL, MYSQL, and Oracle. The proposed model will preserve queries in an XML file as a preservation strategy. Then, the transformation of queries into relational algebra will be addressed to define the XML file's structure. Finally, the procedure for retrieving XML queries will be discussed.

1.7 Thesis Structure

This thesis is arranged into five main chapters. The introduction introduces the research background, research problem, main objectives, methodology, and significant research. The second chapter provides a literature review discussing relevant research. The third chapter describes the research methodology, and the fourth chapter presents a case study implementing the methodology and the results. The final chapter provides a brief conclusion of the research with recommendations for future research.

Chapter One: Introduction

This chapter introduces the research motivation and problem, providing a brief description of the research objectives, questions, methodology, and research significance.

Chapter Two: Literature Review

This chapter presents the theoretical background of this research, including definitions of research variables, theories, models, perspectives, studies, and scholarly achievements. The literature review focuses on RDBs and long-term preservation theory to show the theoretical framework for long-term preservation and RDBs prior to the research analysis.

Chapter Three: Methodology

This chapter provides details about the research methodology. The appropriate research method for each topic is described with both descriptive and experimental methods. This chapter also defines research samples, procedures, and a proposed solution model.

Chapter Four: Case Study

The case study chapter presents the proposed model and its implementation and organisation, descriptions associated with research questions and objectives, and the suitability of the proposed model for solving the research problem.

Chapter Five: Conclusion

This chapter discusses the results extracted from the deployment of the proposed model in database structures and integration between results and the theoretical background. It also discusses plans for developing the research findings.

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

The digitisation of data has created issues pertaining to data storage and, more importantly, preservation, in terms of both authenticity and data integrity over long periods of time and across different types of database systems, including those currently in use and those that might be used for future data retrieval. The collection of digital data to populate a database is limited by factors such as the cost of operation, availability of data for time-limited cases, and the availability of resources. These factors prevent the instant creation of databases with relevant data and create a need for a means of preserving existing databases for future use. Other types of data, such as geological and scientific data, require continual collection and preservation over long periods of time without integrity loss (Gordon and Chaczko, 2015). Thus, long-term preservation of digital data involves a series of well-managed activities undertaken to ensure the long-term accessibility of databases (Ashraf and Kumar, 2016). In this context, the preservation of databases is not only concerned with data but also with the structure, description, and other components of database, such as queries and reports. This paper specifically proposes a methodology for preserving RDB queries.

2.2 RDBs

2.2.1 Digital Data

Digital data are discrete and discontinuous representations of information or objects rather than continuous representations of information in a wave or other continuous functions. Although digital information represents discrete information, such as numbers and letters, this information can also be continuous, such as in audio and video data. Digital data are characterised by a synchronisation scheme that presents the beginning and end of a sequence of ordered symbols and requires a formal language to communicate digital information between a sender and a receiver. Because data are represented as a sequence of symbols, digital data can be copied indefinitely and without error (Van Tassel, 2013). This property has led to new services, such as persistent identifiers (e.g., ISBNs) that may not provide direct access to digital media or identifier–resolver networks that allow data to be transferred remotely.

Representing analogue values in a digital form entails selecting a sequence of symbols to represent the analogue values, which often introduces quantisation errors caused by differences between stored values and original analogue values, such as when decimals from analogue values are rounded off (i.e., the granularity property of digital data; Kumar, 2014). Finally, digital data are easily compressed for transmission and decompressed at their destination, allowing for faster transfer of information (James et al., 2013). These properties allow for easy storage of data in computer systems that can be accessed when necessary. Although storage methods may differ, each represents a different form of data, different levels of compression, and different use cases (Miller, 2014).

2.2.2 Databases

Databases are an organised collection of digital data that specify structure, queries, tables, and relevant views of objects. This collection is organised to represent the real aspects of these objects; for example, a school may be categorised based on classes and the number of students each class can accommodate. This makes data retrieval simple by using structured queries to locate desired information (Bourgeois, 2014). Databases are created and maintained by a DBMS that facilitates interaction between users, other applications, and the databases themselves and allows for the collection and analysis of data. Each DBMS provides mechanisms for delivering features, such as data availability, administrative controls, security, and performance (Lake and Crowther, 2013).

2.2.3 RDBs

Relational refers to the mathematical representation of a logical relational aspect of a database in which data are presented in tables isomorphic to mathematical relations (MAHMOOD et al., 2010). RDBs store data in models that represent data as tuples (i.e., rows) grouped by relationships and reset into a solid mathematical foundation. An RDB provides a declarative means of specifying data and available queries, allowing the user to query for the information they want, while the DBMS handles the structure in which the data are stored, the procedures for retrieval, and user queries. As database technology evolves, relational data and RDBs have become more technically complex and grown in size, with archives, such as the National Climatic Data Centre (NCDC) database, storing more than 18 petabytes worth of data (NCDC, 2017).

Such large databases create new challenges for long-term preservation of data. Most RDBMSs are sophisticated and provide features, such as highly linked tables, that would otherwise be impossible to handle outside the system and that create adverse consequences when links are broken (Saikia et al., 2015). The data stored in such databases include specifically defined domains and data types, from basic data types, such as integers, dates, and strings, to more complex user-defined types, such as character encoding for multi-lingual support. Additionally, RDBMSs implement varying levels of features, such as the ability to assert predefined conditions, check constraints, generate customised views, automate triggers initiated by user activity, store procedures, use basic and user-defined functions in addition to foreign keys to ensure the integrity of referential data, and create roles to determine which actions specific users are allowed to perform (Dignös et al., 2016).

2.2.4 Queries

A query is a programming language that translates and reflects corresponding natural language questions or informational demands (Stajano, 1998). Commonly, a query is defined as a statement written in a specific programming language to extract information or data previously preserved in readable formats (Plew and Stephens, 2002). A database query is an expression written in a query language designed specifically to define data to be retrieved from the database. These languages have similar structures and formats to programming languages. Queries are used in several scenarios, the most popular being a request from an end user who is directly looking for information related to a database’s content or structure (Upadhyaya et al., 2013). A query action asks a database to perform additional tasks or operations to change or manipulate data, such as updates or deletions; other actions are determined by genres in the DBMS or the query language (Beaumont, 2011).

The most used query statement is the SELECT statement, which is an accessing query that can be converted into an action query (Cox, 2009). SELECT query composition includes both SELECT and FROM clauses (i.e., preserved keywords) and can include various other clauses, such as WHERE and ORDER BY, each of which has distinguishing functions. This research uses the simple and formal syntax composition of the SELECT statement (Plew and Stephens, 2002).

```
 SELECT [ * | ALL | DISTINCT COLUMN1, COLUMN2 ]

FROM TABLE1 [ , TABLE2 ];
```

Query statements cannot be summarised in one section, and this brief description of SELECT statements is given due to their wide usage in database management programming languages. The list of columns following the SELECT statement is defined as the output column of query execution and specifies the required data. The FROM statement is followed by table names from which data are extracted, defined as the table source of data. The SELECT statement in conjunction with various clauses adds conditions to data retrieval; in this case, the WHERE clause was used to define execution with a condition using the following syntax formula.

```
select [ all | * | distinct column1, column2 ]

from table1 [ , table2 ]

where [ condition1 | expression1 ]

[ and condition2 | expression2 ]
```

Query optimisation involves the execution of a query plan because a query can have many possible execution plans deployed by different DBMSs. Each plan has a different cost of execution; thus, optimisation of selected queries requires choosing from a set of alternative plans for same the query, which is addressed in the current research (Kaur, 2013). The cost difference between alternative execution plans can be significant, making the necessity of optimisation inevitable (Wu et al., 2013).

2.3 Long-Term Preservation

2.3.1 Preservation Context

Long-term preservation of databases requires the ability to collect data from different database systems and to ensure data accessibility for long periods of time. This means that databases should be maintained independently of their specific management systems, which can be short-lived (Roland and Bawden, 2012). While most major RDBMSs implement standard SQL support, it is almost impossible to port database layouts and SQL codes between management systems without modification and functionality loss (Jark et al., 2014). This makes the process of data preservation complex because it must handle intricately different database layouts and present data in a uniform and independently accessible manner. While RDBMSs implement core SQL standards, they have a large number of custom, product-specific, and non-standard functionalities that result in significant incompatibilities between databases. Modern RDBMSs also implement application-level data storage by moving physical data storage from file levels managed by the operating system to internal storage managed independently (Hellerstein et al., 2007).

2.3.2 Current Implementation Practices

A number of mechanisms are used to preserve highly and intricately linked databases in an accessible form for long periods of time despite technological changes (Locuratolo and Palomäki, 2015). Database preservation requires preservation of all data characteristics, including behaviour, context, appearance, and content. One common means of converting data for storage is denormalisation of databases to produce plain text files that contain tabulated data stored in tables with fixed lengths and delimited columns (IBM Knowledge Centre, 2017). These files are accompanied by separate data dictionaries and metadata files that provide a context for understanding the information and its provenance (Date, 2013). These descriptive files can be stored physically or electronically with copies of the data made every few years to prevent degradation (Stefanova, 2013). However, this method of archiving cannot present multiple data structures and tables in a single table, leading to the loss of precision in the archived data and inefficient long-term preservation.

2.3.3 Repositories

Repositories or the grouping of separate but related databases is another database preservation strategy. Repositories can be maintained individually or institutionally and collect data from various databases in a single logical entity that allows users to interact with that data in a single database. One means of operating a repository is to maintain a single system that directly accesses data items stored in another database. The principle use of repositories is to provide a logical aggregation of data from multiple databases and to provide functions that would not otherwise be logical when using each database individually (Koopman and de Jager, 2016). Repositories function as a database preservation method when they make local copies of databases, although they do not fully describe the data they store and are only linked to the parts of the databases that provide relevant data. Commercial repositories, such as institutional repositories, provide libraries for publishing databases for long-term storage and are preferable over personally managed repositories (Perrina et al., 2016). While these libraries offer different functionality from digital archives, they are an incomplete form of database preservation because they store copied data over long periods of time (Perrin, Winkler and Yang, 2015).

2.3.4 Emerging Database Preservation Technologies

Modern RDBMS structure differs from the structure of databases resulting from RDBMSs, making it difficult to create a universal digital data management standard. Although current technology, such as the Software Independent Archiving of Relational Databases (SIARD), has advanced the extraction of database properties, little research has been done on query preservation to enhance long-term preservation of databases (Thomson, 2016). SIARD has attempted to extract data automatically or manually for unknown or custom RDBMSs and archive these data as separate .zip or .zipx or as XML files containing descriptive information of data structures (Swiss Federal Archives, 2017).

The use of XML to store metadata is not a new concept, and it uses the same schema used to capture data and table structures in the database mark-up language. XML was chosen for its ability to extend to linked resource description frameworks for further optimisation based on the type of data being stored (Hardesty, 2016). This model of database preservation integrates different databases into a single archive without the complexity of changing data or structures to fit a DBMS-specific structure. Such preservation is efficient and preserves data integrity and authenticity over time.

2.3.5 Challenges to Data Preservation

Apart from the need to incorporate different database structures and schemas defined by different RDBMSs, data preservation must also take into account factors that ensure successful data preservation. Among the primary requirements for preserving data is the maintenance of the integrity of stored data—data must represent the same information as it describes in its original state. Additionally, archives should be stored in an accessible environment to provide continual operability over long periods of time. Stored data should be independently accessible without requiring the use of a specific DBMS to read the information, which should be kept in secure storage both physically and virtually for protection from unauthorised access and manipulation (Elhai, Levine and Hall, 2017). To provide this capability, researchers often make compromises between the safety and stability of archives and must choose between outsourcing to stable data-preservation platforms and storage facilities or commercial cloud-service providers, both of which risk data exposure to third parties (Stancic, Rajh and Brzica, 2015). Choosing the latter may also expose archives to targeted hacking activities.

2.4 Relational algebra

This study represents a model of preserving queries in a relational algebra format; thus, this section provides a brief background for the theory of relational algebra.

2.4.1 Algebra Definition

Algebra takes a defined set of variables and modifies them using a set of defined applicable operators. An operator and one or more variables make an algebraic expression. The rules can be defined in an expression but not every rule can be applied to all operators (Trissl, 2012). Algebraic operators are applied to an expression and can be defined as single or multiple relation operators. Single relation operators are listed below (Franchitti, 2014).

- Selection operator (σ): Extracts relation tuples (rows)
- Projection operator (π): Extracts relation attributes (columns)
- Renaming operator (ρ): Changes relation names

Multiple relation operators include the following (Franchitti, 2014).

- Set operators ($\cup$, $\cap$): Perform union and intersection functions
- Cartesian product ($\times$): Extract all tuples from two relations
- Join operators ($\bowtie$): Select joined tuples from two relations based on a condition
- Minus (-): Extracts all tuples excluded in the relational
- Semi-join ($\ltimes$): Extracts the tuples from one of the joined tables.
- DIVISION ($\div$):gives a relation that includes all tuples in one relation in combination of other relation
- Left/right outer (⟗): all tuple in left(first)/ right(seconed) relation are kept in the result of R ⟗ S

Rules related to writing expressions contain numerous relations and operators in different orders without affecting execution results, which is similar to mathematical operators. For example, the selection and join operators in Equation 1 can be rewritten as in Equation 2 (Franchitti, 2014).

$$\sigma c(R \bowtie S) = \sigma c(R) \bowtie S \text{ iff } a \in A \ldots\ldots\ldots (1)$$

$$\sigma c\ (R \bowtie S) = R \bowtie \sigma c(S) \text{ iff } a \in B\ldots\ldots\ldots (2)$$

The extracted list from R joined S and satisfied the condition (C), which executes the joined group and selection operators based on the join property. The selection of all tuples in R satisfied the first condition and executed the join with S to ensure the the tuples in R satisfied the condition and joined one in the S and the vice versa.

2.5 Limits of Existing Practices

One of the limitations of current practices of database preservation is the inefficiency of the resulting archives, in terms of their ability to accurately depict the databases they seek to preserve (e.g., exported archived element limitations, data access and performance, and data exchange formats; Lindley, 2013). As requirements, long-term data preservation should not alter the structure, content, appearance, or meaning of data, and current data-preservation methods fail to meet these requirements (IBM, 2010). The reason of this problem is due to the existence of a variety of RDBMSs that, despite being SQL-compliant, create databases that are customized or optimized for use with only one DBMS. This makes the process of porting databases from different RDBMSs much more difficult to achieve and even more difficult to automate (Park and Brenza, 2015).

Although these archiving methods demonstrate the necessity of preserving database queries in various forms as part of the preserved databases, they have not explored the possibility of using queries as a method of enhancing the preservation process. Given the varying capabilities of RDBMSs, the extraction and building of queries differs for each RDBMS, and the creation and performance of queries in these systems are supported by different functions. The same problem arises for accessing archived data from RDBMSs that are different from those in the original ported databases, which requires modifications for compatibility. Although current methods have been successful for preserving data in some form or another, preserving whole databases without altering structure and content remains a challenge. Current methods have advanced query transformation; however, they have not attempted to integrate query use to enhance long-term preservation of databases.

2.6. Optimisation

Optimising a query occurs during the process of extraction from an RDBMS by converting a query from an SQL format to a relational algebra expression for easy transfer and cost savings in data retrieval. This also optimises compatibility between different RDBMSs and preserves complete query structures of databases while maintaining usability across different platforms over time. This is achieved through a process that isolates queries to fit the profiles of different RDBMSs (IBM, 2013).

2.7 Research Gaps

Although there has been extensive research conducted about data preservation methods, extraction of metadata, and data storage and accessibility over long periods of time, research on preserving whole databases, including different components, is still ongoing. The current research proposes the use of relational algebra expressions to represent all queries, stored in a format that simplifies data-retrieval from RDBs and maintains optimal execution and preservation of data, regardless of the specific RDBMS used to create them. This process transforms the query from being a non-preserved part of a database to being part of the preservation process by not only improving the process of data retrieval but also by optimising the same data for presentation across various RDBMSs. This research also explores optimisation of the execution strategy and the processing time for queries through the reduction of redundant procedures and transformations. It does this by separating database-specific and dynamic features that are common versus those that need to be changed for each database. Thus, queries are not only used as basic data-retrieval tools but also as a way to improve databases by preserving referential integrity, authenticity, and accessibility independent of the software used to access or create them.

CHAPTER THREE: METHODOLOGY

3.1 Introduction

This chapter presents the proposed method and procedures to satisfy the research objective and answer the research questions put forth in the introduction. The main objective of this research is to propose an appropriate prototype for long-term preservation of queries within RDBs, ensuring consistency and functionality in the databases that are aligned with organisational needs. This chapter covers both research methods and research designs, presenting research tools and a comprehensive overview of the research procedures.

3.2 Research Procedures

The first phase of the research involved collecting characteristics for each RDB, while the second phase involved tabulating and categorising these characteristics, focusing on the important and unique features of each database. Based on this table, the research defined the main characteristics, functions, and compatibility needs that must be preserved through the proposed approach.

The experimental phase consisted of two steps. First, all common features among the implementation and database structures of RDBMSs were identified to generate a list of common features, functions, compatibilities, and incompatibilities for each system. These were organised into subcategories for common, cross-platform features that required no modification (e.g., core implementation of SQL and descriptions of table columns) and placed in the first level of the schematic representation of the database. Second, features available but implemented differently in all databases were identified to create an intermediary model to translate databases into a common relational algebra form to preserve queries. This allowed the field to accommodate all data from the various RDBMSs. Finally, the research defined a translation model from basic SQL query structures in relational algebra, including customised views, functions, and triggers to provide query building and preservation stored in XML syntax form as follow:

```
<?XML version= “1.0” encoding=“ISO-8859-1”?>
<root describes=”common_features”>
         …
         Features, structures, functions and other implementations common to all RDBMSs.
         …
  <translated describes=”common_but_different_features”>
          …
         Cross-compatible features, functions and other implementations.
         …
    <custom describes=”custom_features”>
         <for dbms=”dbmsA”>
         Features unique to dbmsA
         </for>
         <for dbma=”dbmaB”>
         Features unique to dbmsB
         </for>
    </custom>
  </translated>
<root>
```

This XML file will then provided complex, consistent mapping of a relational algebra tree for queries, creating an expressive XML schema as a preservation strategy.

3.3 RDB Characteristics

The aim of this survey was to determine the main differences among database and query languages deployed to verify that XML could be used an intermediate language for all databases.

3.3.1 MS Access

MS Access is a Microsoft product that is widely available as an RDB tool and that combines a graphical user interface with software-development tools. MS Access is a desktop database, which means all data must be preserved on an individual computer. The tools preserves, systematises, and administers data retrieved in various formats and from various reports and navigates database objects, such as tables, queries, forms, relations, and reports (Lemons 2016). MS Access consists of two main elements: a Jet/ACE database engine and a Rapid Application Development tool, both of which facilitate form- and report-building restrained to the database.

MS Access stores all database objects in one file format (‘mdb’) that stores tables, relations, queries, forms, and reports in a single-user session file stored on the user’s devices. In a multi-user environment, the 'mdb' file is located on a file server so that all users can share the file. MS Access supports both user and administrator environments; in a user environment, one can insert and retrieve data, while in the administrator environment, one can use additional functions, such as design, modify, create and define relations, create new form, and define index (Lauesen 2011). The variety of access objects allows users to easily manage access and analyse stored data. The following is a brief description of access objects (MSDN Library, 2017).

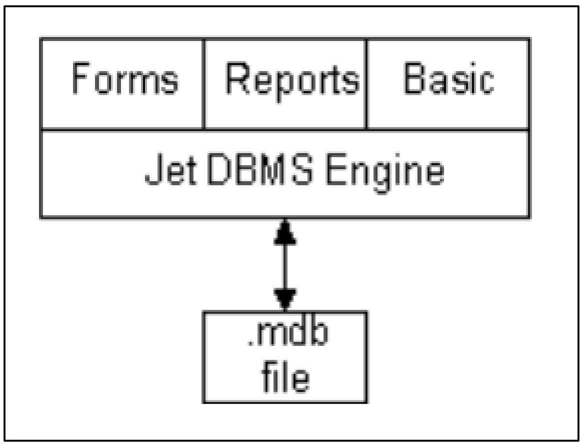


Figure 1. MS Access architecture (Oracle, 2013)

Table: A combination of columns and rows. Each column has a unique name and a header called a 'field' that is assigned to a specific data type. Each row represents data information, called a record for particular instances.

Query: A basic data-retrieval process used to explore, edit, and analyse stored data. Queries can be created using a wizard or through query design functions. Queries can be created in numerous forms, including the following (Roman, 1999).

1. SELECT queries retrieve data from one or more tables based on criteria such as category, range, and Boolean operators.
2. PARAMTER queries are prompt queries based on the value of defined parameters in the database.
3. CROSSTAB queries are synonyms of the pivot table wizard. A CROSSTAB query is a wizard query displaying value summarisations, such as sums, counts, and averages.
4. ACTION queries influence multiple records in a single query. An ACTION query is associated with one of four actions: delete, update, append, or table.
5. SQL queries are made using SQL statements, such as union (which combines multiple fields into one), pass through (which sends orders to the open database connectivity [ODBC] data sources to retrieve records), and edit data or run.
6. Data definition queries utilise the data definition language (DDL) to create, delete, or edit tables and create indexes.
7. Sub-queries are a combination of two select queries or one select query and one action query.

Form: A customised tool for inserting, exploring, and printing data from a database in a simple manner to ensure reliability.
Report: A customised tool for extracting and analysing data and for creating an easy-to-read summary of the database.

MS Access supports several data types, including text, memo, numbers (e.g., integer, long number, single, double, and decimal), date/time, and Boolean. It also supports datatype changes in previously stored records (Lauesen, 2011).

The Jet database engine controls channels to external data sources and ODBC. The advanced version of Jet contains an access database engine (ACE) and creative security features not implemented in earlier versions. During the query process, the ACE returns a record set based on query criteria as either snapshots or dynasets. Snapshots are an image of data representing status at query time; while editing data in snapshot images is not allowed, other operations are, such as query, forms, and reports. Dynasets are a live view of data, and key values are extracted and stored in the memory by the Jet engine, which uses key values to index and retrieve data from external databases instantaneously upon request (Microsoft Office, 2010). MS Access supports indexes by preserving them in tables and defining them as primary keys that are created automatically or defined by a user (Tutorials Point, 2017).

MS Access partially supports the referential integrity of SQL commands and supports the creation of various relationships between tables to provide an entity relationship model (i.e., a graphical tool that simplifies the relation set between tables). The main contribution of referential integrity is to check matches between foreign keys and primary keys when defining relationships. The entity relationship interface demonstrates referential integrity in cascades and no-action referential actions. A cascade action is when the user removes the corresponding foreign key for a deleted record, while a no-action action is an unauthorised deletion process for records that are dependent on foreign keys (Blaha, 2005; Lauesen, 2011). Three types of relationship creation are supported: 1) 'one to many,' in which a single field with a primary key is related to another record in the table; 2) 'one to one,' in which a single record in a table is related to a single record in a different table; and 3) 'many to many,' in which multiple records are related without a unique value or a relationship is composed of two one-to-many relations (McFadyen, 2015).

3.3.2 MS SQL Server

SQL Server is a cloud-ready Microsoft information platform that guarantees security and scalability of data across devices or in private and public clouds. It is a dependent platform that supports the structured english query language (SEQUEL). MS SQL Server uses a client–server architecture, in which a workstation is equipped with operators, such as the SQL server management studio and SQL server configuration manager, and the server station is equipped with a centralised server, such as the SQL server, SQL server agent, Microsoft SQL Server integration services, SQL server analysis services, SQL server reporting services, or SQL browser (Tutorials Point, 2016a).

MS SQL Server consists of three main elements: 1) SQLOS, which enforces basic SQL server services; 2) the relational engine, which enforces RDB elements; and 3) the protocol layer, which demonstrates SQL server functionality (Acronics, 2008). SQLOS executes primary functions related to the operating system, such as thread scheduling, memory management, synchronisation primitives and locking, and deadlock detection. MS SQL Server has a special standalone memory-and-thread management system that executes tasks (Delaney et al., 2009). The relational engine associates stored relational data by utilising SQLOS features in the SQLOS API and supports and defines data types authorized by the software. It consists of a storage engine that manages data preserved on defined devices. In addition to controlling data access, the engine also executes log-based transactions to ensure analysis console for intrusion databases (ACID) compliance when changing data. The relational engine also includes a query processor that handles data retrieval through the SQL query form, using a query parser, query optimiser, and query executor to translate a query into a series of operations scheduled for execution by SQLOS (Acronics, 2008). The protocol layer provides a user interface for the SQL server. All operations performed by the user go through this layer to the tabular data stream defined by Microsoft; these are then transmitted to the lower physical layer to reach the SQL server (Delaney et al., 2009).

MS SQL Server supports numerous datatypes, including text and characters (char, varchar, nchar, and nvarchar), numbers (tinyint, smallint, integer, and decimal), currency (smallmoney and money), date/time (smalldatetime and datetime), Boolean (bit), and a special type of image binary (Mistry and Misner, 2012). MS SQL cooperates with the Business Intelligence Studio to offer report services, and it has a unique architecture for data files, as described below (Tutorials Point, 2016a).

File groups: These data files are constructed from a combination of files for allocation and management purposes. The file group is defined as one of two types: primary and user-defined. The primary file group consists of primary data files with miscellaneous files that are not classified or related to any group. The user-defined file group is a group of files assigned to utilise keywords for database creation or statement editing. Initially, the SQL Server creates one primary file group as the default file group (Delaney et al., 2009).

Files: SQL Server supports three types of files: the primary data file, secondary data file, and log file. The primary data file is the initial file used for database creation with a common extension of ‘mdf’. The secondary data file is not a necessary file in the database; thus, some databases do not have secondary data files and others might have multiple files with the common extension ‘ndf’. Log files save the essential log information for recovering the database and have a common extension of ‘ldf’. All file locations are stored in both the master database and the primary data file. However, metadata are stored only in the master data files (Tutorials Point, 2016a).

Extents: These are the spaces allocated to tables and indexes. An extent consists of eight adjacent pages. There are two types of extents: uniform, consisting of a single object, and mixed type, consisting of up to eight objects (Delaney et al., 2009).

Pages: These are the elementary unit of MS SQL Server data storage. Each page has a header used to save system information. There are nine types of pages: a data page consisting of data rows without text, ntext, and image row data; an index page that handles index entries; a text/image page that handles text, ntext, and image data; a GAM page that handles allocated extent information; an SGAM page that handles system-level allocated extent information; page free spaces that handle information about available page spaces; an index allocation map page that handles extent information utilised by the table or index; a bulk-changed map that handles modified extents by bulk operation; and a differential-changed map that handles extent information changed since the last backup of database statements (Tutorials Point, 2016a).

Query execution and procedures preserved in a procedure cache reduce the number of queries generated. MS SQL Server supports all SQL query languages, including data query language (DQL), data-manipulating language (DML), data-definition language (DDL), and data-control language (DCL). This allows for common commands, such as select, query, creation of table, check, like, insert to, update, and delete (Halvorsen, 2016). For referential integrity, MS SQL Server utilises a foreign key as the reference to a primary key or a special combination of columns. Even though MS SQL Server does not support referential actions as set null or set default, it fully supports cascade actions and no-action actions, which are different from the SQL standard (Blaha, 2005).

3.3.3 MySQL

MySQL is an open-source RDBMS that supports SQL and multi-user access and implements various communication protocols for authentication, querying, and server administration. MySQL also has inner API support for C, C++, and Eiffel and can be connected to other databases, such as *Object* linking and embedding, database (OLE DB) and ODBC, in Microsoft environments. MySQL supports various storage engines, although the InnoDB storage engine is recommended due to its support of various relation creations (Shirish, 2010).

The MySQL structure consists of tables grouped in columns and rows, in which columns contain data associated with a column-defined datatype. MySQL data are saved in separate tables, and the user can define the relation between fields as one-to-one, one-to-many, unique, required, or optional, with pointers between tables. MySQL also supports temporary tables that save temporary data during sessions that are deleted when a session is terminated. As a client–server platform, it preserves database metadata, which the user can fetch using a select query (MySQL, 2017). MySQL supports the following data types: text (char, varchar, tiny text, text, medium text, long text, and set), numeric (tinyint, smallint, mediumint, int, bigint, float, and real), currency (decimal), date/time (date, time, and datetime), Boolean (tinyint and enum), and special types (tinyblob, blob, mediumblob, and long blob; Tutorials Point 2016b).

The InnoDB engine ensures ACID compliant features in all transactions and save and query processes that satisfy data integrity. It is also includes in-row level locking in multi-user sessions, and user data are stored in cluster indexes to reduce input/output (I/O) operations. All queries are based on a primary key (Shirish, 2010). Queries defined in MySQL are categorised using the SQL query languages, including DQL, DML, DDL, and DCL. It supports commands, such as insert for data insertion, select for retrieving data from the database, update for modifying stored data, and delete for record deleting (Tutorials Point, 2016b). MySQL also supports indexes, which can be defined as a primary key, unique, index, and fulltext. Thus, MySQL enhances referential integrity and fully supports the SQL standard because it supports all referential actions (cascade, set default, no action, and set null) for deleting and updating. Database indexes are created for foreign keys and reference keys, where the referenced column can be the primary key or unique (Blaha, 2005).

3.3.4 Oracle

Oracle has two disconnected structures: logical and physical. The disconnect ensures that the logical structures can be accessed without affecting the administration of physical structures that include data files consisting of data from databases. Redo files contain redo records describing all changes made to data, which are utilised for recovery and control of files. These data files describe a database's meta-information, such as the name of the database and location. The logical structures contain data blocks, extents, and segments to facilitate storage space control in tablespace (Oracle, 2017).

Oracle defines several objects in the database. Tables are the fundamental unit, containing data entered by users. Indexes are optional objects that improve retrieval performance, especially for large databases, and also indicate the location of information. Views are customised data in tables and can also consist of stored queries. Clusters are groups of tables sharing one location based on column similarity between tables. The database instance is a memory architecture set for managing database; it is intentionally saved separately from the database file and contains a system global area set and a background process set (Oracle, 2017).

Oracle supports communication over several networking protocols, such as TCP/IP, HTTP, FTP, and WebDAV, and transaction implementation in which each SQL statement performs a logical transaction (Oracle, 2013). It also supports PL/SQL programming used to write programs and triggers in Oracle and allows users to save codes (PL/SQL procedures or functions) aligned with the database. There are stored procedures in pre-compiled forms on the server as well as a general form that can be used by user (Kytes, 2005).

Oracle utilises SQL query statements in SQL query languages, including DQL, DML, and DDL, and can handle commands such as update, insert, delete, select, merge, great, alter, drop, rename, grant, revoke, commit, and rollback (Singh and Pottle, 2009). Oracle supports the following datatypes: text (char, varchar2, nchar, long, and clob), numeric (byte, smallInt, integer, number, float, and real), currency (money), date/time (date), Boolean (bit), and special characters (blob, raw, and long raw). It also includes a data dictionary that contains read-only metadata tables. Indexing is supported by defining a candidate key that is assigned to one or more columns. The primary key is a candidate key that has special characteristics. The index for tablespaces uses a primary key and a foreign key, which is a group of columns that have a primary key value (Loney, 2009). In Oracle, the primary key or the column group can be referenced by a foreign key. Oracle also supports delete actions that allow the following referential actions: cascade, no action, and set null. It does not support update actions or set defaults; however, no actions are implied based on the referential integrity weakness (Blaha, 2005).

3.4 Survey Results

The following table shows the characteristics of the four RDBs described in this section.

Table 1. Datatype characteristics of four RDBs

Data type	Text	Numeric	Currency	Date	Boolean	Special
Description	alphanumeric data	Numerical data type variation in length	money fields	datetime, date	two value data yes/no check box	represent graphics, sound, hyperlink
MS Access	text memo	integer, long number, single, double, decimal	currency	datetime	yes/no	OLE object hyperlink
MS SQL Server	char, vchar, nchar, nvarchar	tinyint, smallint, integer, decimal	Smallmoney, money	smalldatetime datetime	bit	image, binary
MySQL	char, varchar, tinytext, text, meduimtext, set, long text	tinyint, smallint, mediumint, int, bigint, float, real	decimal	date, time, datetime	tinyint, enum	tinyblob, blob, mediumblob, longblob
Oracle	char, nchar, varchar2, long, clob	byte, smallint, integer, number, float, real	money	date	bit	blob, raw, long raw

Table 2 summarises the types of data saved in the four databases. As shown, the SQL is not stored in any of the databases except MS Access, which has some SQL queries saved in the wizard while most are saved in the front end or in the documentation database, which are rarely accessed or lost over time. The common element among these databases is recognition of basic query languages, but data saved varies among databases, as shown in Table 2. All databases save data, although MYSQL does not save indexes, and MS Access has a distinguished query wizard (SQL) and SQL server that provides query execution.

Table 2. Data saved in the four databases

MS Access	SQL Server	My SQL	Oracle
data, indexes, forms, reports, query wizard (SQL)	data, indexes, extents, log file for recovery, location saved in master data file, query execution	data, database, metadata	data, indexes, stored queries, PLSQL procedures or functions, metadata in redo log

Table 3 summarises the SQL supported languages (i.e., DQL, DML, DDL, and DCL) in the four databases. SQL is not fully supported in some databases; SQL Server and Oracle support all SQL, while MS Access and MySQL support some but not all.

Table 3. SQL language support in the four databases

MS Access	SQL Server	My SQL	Oracle
DQL , DDL	All SQL (DQL, DDL, DML, DCL)	Select (DQL), DDL (insert, update, delete)	All SQL (DQL, DDL, DML, DCL)

Table 4 summarises the four RDBs' referential integrity, which is fully defined in the supported SQL; thus some databases partially support the SQL standard, while some fully support it.

Table 4. Referential integrity of the four databases

MS Access	SQL Server	My SQL	Oracle
Based on partial SQL support, referential integrity is lacking, but the E/R graphical is able to cover this shortage as well as cascade and no action in the referential integrity action.	Does not fully support SQL standards, and support varies for set null and set default actions. It only supports cascade and no actions. Also, the no-action syntax is varied from the SQL standard.	Fully supports SQL standard. Supports all actions in delete and updates. Also requires an index defined in the foreign key and the referenced key.	Supports delete action but not the set default or update actions. The no-action syntax differs from the SQL standard as well as from the SQL server database.

3.5 Proposed Solution

Based on Tables 2 and 3, SQL is either not fully supported by the databases or not fully stored in the databases. One solution is to access the SQL code of the databases and scan for the DML of the SQL to represent the code in a common language (i.e., relational algebra). This means converting all queries based on SQL into relational algebra, which is proposed because it is understood by both users and database software. Thus, a conversion tree is proposed to be saved and stored as one main database structure that supports preservation regardless of which SQL version a database uses because main queries are built based on the relational algebra tree stored in the database as either a tuple or an XML file. Relational algebra has a solid mathematical representation that considers the basics of any new development or improvement theorem. The main feature of relational algebra is that it can be substituted with the most efficient plan of execution.

3.5.1 Relational Algebra

Relational algebra is an algebraic query language correlating with a relational model. The algebraic query is a combined set of algebraic operators corresponding to a structured relation expression model (i.e., tree). The input relation is represented as a leaf node and the relational algebra as an internal node where the executions are involved. There are many relational algebra operators; however, only some operators are described below.

- Select (σ): Selects a set of records
- Project (π): Deletes unwanted columns
- Rename (ρ): Renames a relation or attribute
- Set difference (-): Explores records in one table but not in another
- Union ($\cup$): Clarifies all records in two tables
- Cartesian product ($\times$): Combines two relations
- Join ($\bowtie$): Combines similar tuples from two relations into longer tuples

Each set of SQL operators corresponds to relational algebra operators. For example, the SELECT query in SQL is a basic retrieval query defined in all databases considered in the research. The syntax for the SELECT command is as follows.

SELECT x, y, z

FROM R_1, …., R_2

WHERE where-condition

This is converted to the following in relational algebra:

$\pi_{x, y, z.}\ \sigma_{\text{where-condition}} (R_1 \times \ldots\ldots \times R_2)$.

Below is a more complicated SELECT statement with a join operation.

SELECT x, y, z

FROM R_1 as R

INNER JOIN w as w_1

ON w.x = ci.z

AND i.y = C.z

INNER JOIN tt As T

ON T.y = i.y

This is converted to the following,

$\pi_{x,\ y,\ z.}\ \sigma_{\text{where-condition}}\ ((R_1 \bowtie_{w.x = ci.z} w) \bowtie_{T.y = i.y} tt))$.

Thus, the query tree is given as below.

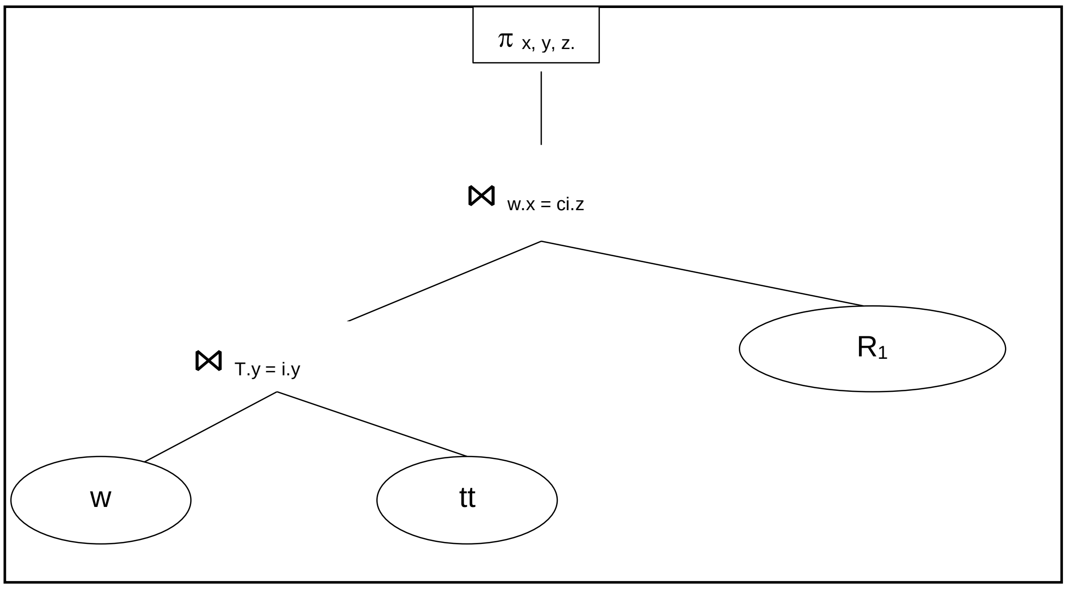


Figure 2. Relational algebra tree for previous expression

3.5.2 XML as a Preservation Strategy

XML converts a range of file formats to one normalised representation (XML, 2002), and it has been described as a general purpose data manipulation language for both structures and content (Hunter et al., 2004). XML is an open standard defined by the Wide Web Consortium and is a straightforward means to preserve data in accordance with document type definition standards as well as an appropriate format for digital object preservation, regardless of the objects original format. Thus, XML is optimal for normalisation and provides the most cost-effective and simple format that mitigates data loss or corruption risks (Ball, 2006; Hunter and Choudhury, 2003). Future programmers and software will always be able to work with XML, and it is easy to convert XML to any new format because it uses well-structured data tools with metadata. This research adopted the XML format for the following reasons.

- XML is platform independent in that it does not depend on any specific software or application platform.
- XML can be converted to any accessible format by future users, allowing data to be represented in new formats readable in a human language. XML is simple, which allows for easy conversion to new formats, especially those developed from XML.
- XML provides a clear, detailed description of a digital object's structure.
- XML storage for query preservation upgrades any database redundancy problems by manipulating XML design documents, which are free of redundancy and based on relational schemas (Kolahi, 2008).
- XML provides an accurate digital reproduction, and its output is suitable for any data scheme based on an RDBMS.
- XML guarantees authenticity and integrity due to the XML signature property.

3.6 Discussion

Based on the survey of the four databases, all databases are primarily concerned with preserving data or digital material using unique record and field forms. All databases are similar in terms of primary defined properties, such as name, datatype, size, foreign key, and primary key. With respect to datatypes, all databases support text, numerals, Boolean, and date and time, but differ in terms of support for datatype length or category. For example, MS Access supports alphabetic text in one type (text memo), while the three RDBs have numerous categories for this datatype. However, each database supports a specially defined datatype different from the other databases. For example, MS Access supports the hyperlink datatype, while the other three databases support large binary files for audio or video to various degrees.

The main concerns of optimisation for data retrieval are refraction of long-term preservation ideas, retrieval based on query build, query preservation, and referential integrity. These concerns constrain data-manipulation functionality. The tables show that most referential integrity is associated with indexes and primary keys, which preserve data. The four databases all save indexes along with data files, but SQL saves extra records and files to facilitate retrieval and recovery of data.

Referential integrity is defined in several actions, such as set null, set default, no action, and cascade, that are defined in SQL languages (i.e., DQL, DDL, DML, DCL), although some databases do not fully recognise SQL (e.g., MS Access). Additionally, the partial implementation and recognition of

SQL languages constrain the functionality of databases to meet all data requirements and provide functions to support data storage for manipulating and retrieving purposes. Thus, the optimisation proposed in the query requires two main principles. First, data must be freed from SQL dependency by converting all SQL statements into relational algebraic functions, which are commonly and fully defined and supported by all RDBs. This will create opportunities to utilise database-functionalities. Second, the relational algebraic tree must be stored with data files.

CHAPTER FOUR: CASE STUDY

4.1 Introduction

This chapter presents an example of the proposed solution. The case study structure includes three steps.

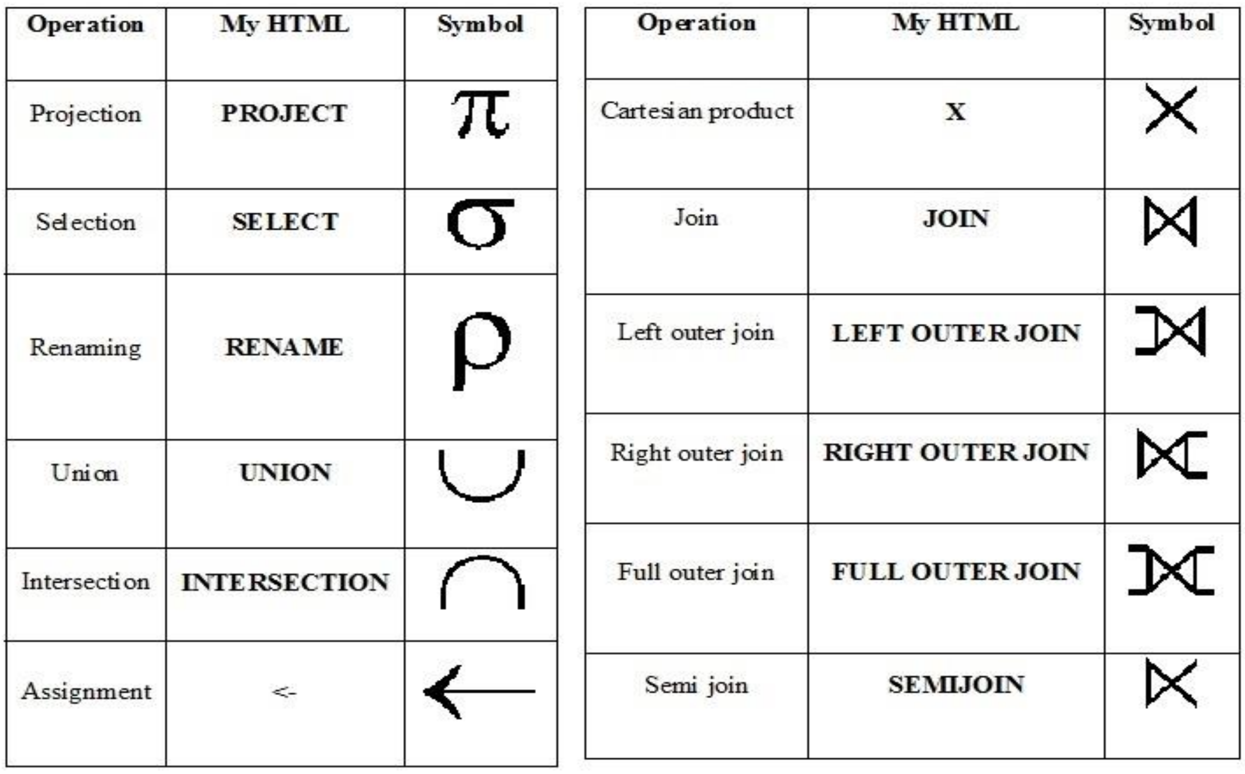
1. Extract the SQL query manually from the DBMS.
2. Convert each selected query into a corresponding relational algebra expression and tree.
3. Transform the algebraic relational tree into an XML tree structure and code for preservation.

To prove the proposed concept, these steps were applied in MS Access, MS SQL Server, My SQL, and Oracle.

4.2 Queries Translation

Because of time limitations, only basic queries were extracted manually to demonstrate the proposed procedures, and because of resource limitations, the work does not consider automation of the proposed operation, although the work illustrated creates a baseline. Another problem is that the differences among the databases in SQL standard compliance mean that one specific standard was not applied. The relational operations have their own symbols. For clarity, the term for the symbol is used in this study, but symbols for each term are given below.

TABLE 5: Operations of relational algebra



Operation	My HTML	Symbol
Projection	**PROJECT**	π
Selection	**SELECT**	σ
Renaming	**RENAME**	ρ
Union	**UNION**	$\cup$
Intersection	**INTERSECTION**	$\cap$
Assignment	<-	$\leftarrow$

Operation	My HTML	Symbol
Cartesian product	**X**	$\times$
Join	**JOIN**	$\bowtie$
Left outer join	**LEFT OUTER JOIN**	⟕
Right outer join	**RIGHT OUTER JOIN**	⟖
Full outer join	**FULL OUTER JOIN**	⟗
Semi join	**SEMIJOIN**	$\ltimes$

For further elaboration, the symbolic relational algebra operators are written in verbal form as in the following example

PROJECT$_x$ (SELECT$_{y<3}$ (5).)

This corresponds to the symbolic notation:

$$\boldsymbol{\pi_x(\sigma_{y<3}(5))}.$$

The main algorithm of conversion is described in the Appendix.

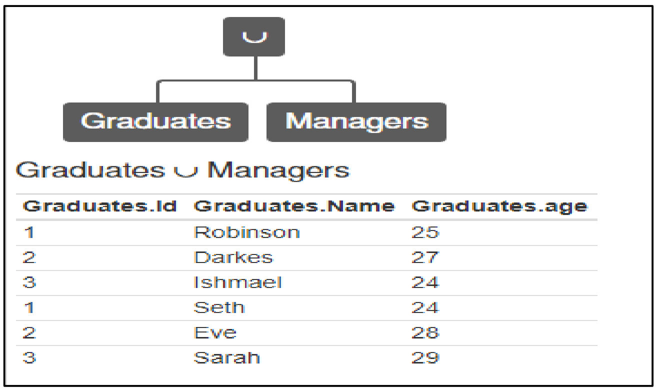
4.2.1 Ms Access

In Ms Access, SELECT query is the main query, and the relational algebra for SELECT query is represented as UNION, INTERSECTION, or MINUS as following. It is necessary to determine how a relation is preserved before conversion.

- Union relation:

```
SELECT * FROM Graduates
Union
select * from Managers
```

The UNION relation is presented in the following relation as a relational algebra (RA) tree.

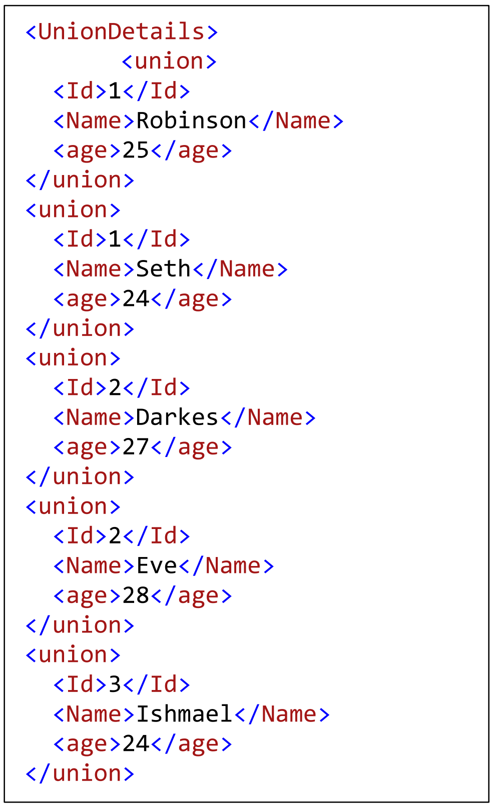


Graduates ∪ Managers

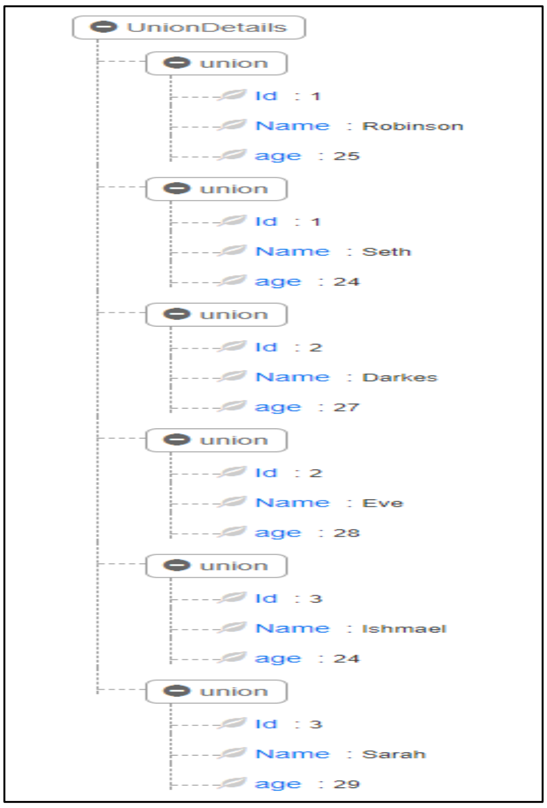
Graduates.Id	Graduates.Name	Graduates.age
1	Robinson	25
2	Darkes	27
3	Ishmael	24
1	Seth	24
2	Eve	28
3	Sarah	29

Figure 3. RA tree for select query represented by union

The XML code and XML tree are illustrated in the following:



```
<UnionDetails>
       <union>
  <Id>1</Id>
  <Name>Robinson</Name>
  <age>25</age>
</union>
<union>
  <Id>1</Id>
  <Name>Seth</Name>
  <age>24</age>
</union>
<union>
  <Id>2</Id>
  <Name>Darkes</Name>
  <age>27</age>
</union>
<union>
  <Id>2</Id>
  <Name>Eve</Name>
  <age>28</age>
</union>
<union>
  <Id>3</Id>
  <Name>Ishmael</Name>
  <age>24</age>
</union>
```

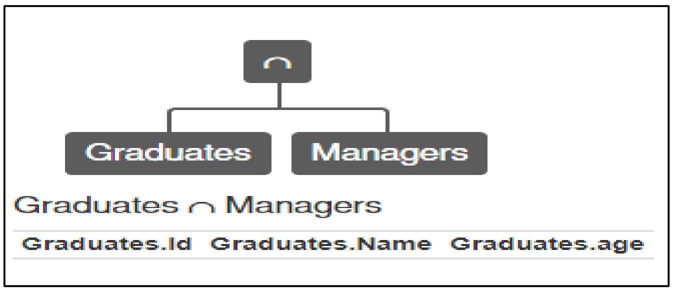


```
UnionDetails
  union
    Id : 1
    Name : Robinson
    age : 25
  union
    Id : 1
    Name : Seth
    age : 24
  union
    Id : 2
    Name : Darkes
    age : 27
  union
    Id : 2
    Name : Eve
    age : 28
  union
    Id : 3
    Name : Ishmael
    age : 24
  union
    Id : 3
    Name : Sarah
    age : 29
```

- INTERSECTION and MINUS is given as:

```
SELECT * FROM Graduates
Intersect
select * from Managers
```

It corresponds to the following RA representation.



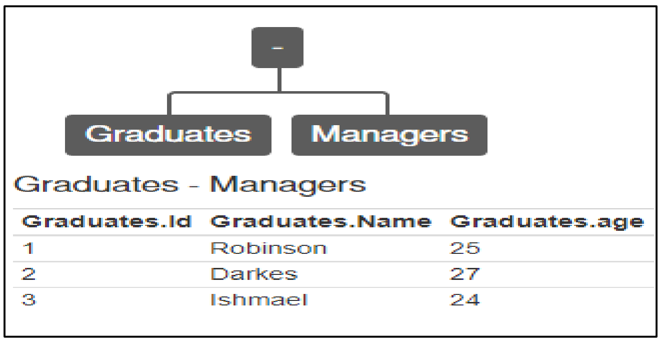
Graduates ∩ Managers

Graduates.Id	Graduates.Name	Graduates.age

Figure 4. RA tree for select query represented by union

However, the MINUS considers alternative operations for INTERSECTION, which is represented as follows.

```
SELECT * FROM Graduates
minus
select * from Managers
```



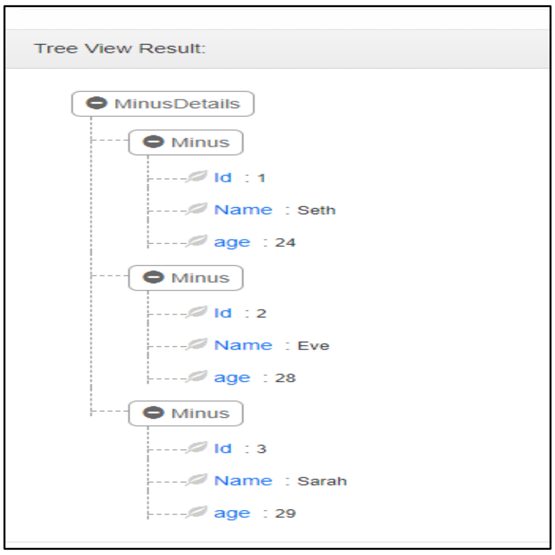
Graduates - Managers

Graduates.Id	Graduates.Name	Graduates.age
1	Robinson	25
2	Darkes	27
3	Ishmael	24

Figure 4. RA tree for select query represented by minus

Both INTERSECTION and MINUS are represented by the same XML statement and tree.

```
<MinusDetails>
       <Minus>
       <Id>1</Id>
       <Name>Seth</Name>
       <age>24</age>
</Minus>
<Minus>
       <Id>2</Id>
       <Name>Eve</Name>
       <age>28</age>
</Minus>
<Minus>
       <Id>3</Id>
       <Name>Sarah</Name>
       <age>29</age>
</Minus>
</MinusDetails>
```


Tree View Result:

```
MinusDetails
  Minus
    Id : 1
    Name : Seth
    age : 24
  Minus
    Id : 2
    Name : Eve
    age : 28
  Minus
    Id : 3
    Name : Sarah
    age : 29
```

Based on the following illustration of relational representations, the conversion deploys the following rules.

Query	Relational algebra
select * from x, y where Vol = Size *equivalent to* select * from (x join y on vol = Size)	X **JOIN**$_{\text{Vol = Size}}$ Y

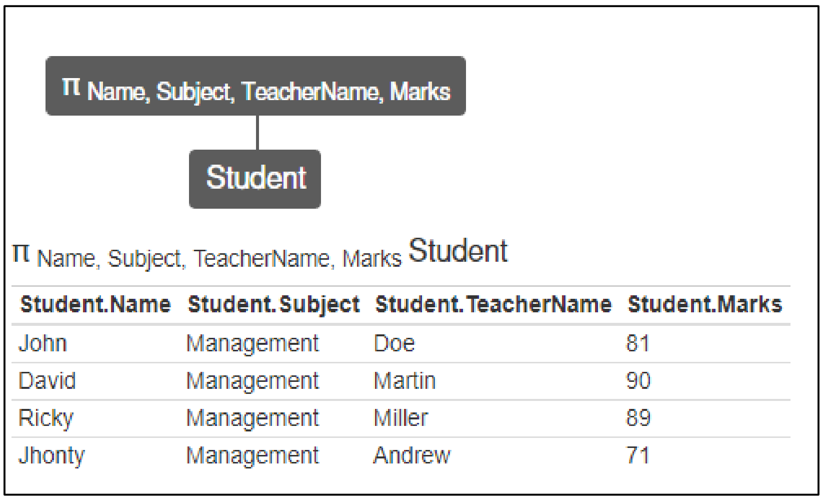
The recursive query, in terms of relational algebra, is impossible, but a special operation (i.e., transitive closure) is proposed to handle unidentified operators in standard SQL (SQL2) that uses a SQL+ combined loop in the host language or recursion.

4.2.2 MS SQL Server

- SELECT query has multiple forms, and the following illustrate the common ones:

```
Select distinct Name, Subject, TeacherName, Marks from
Student;
```

It can be translated into the following RA expression and tree.



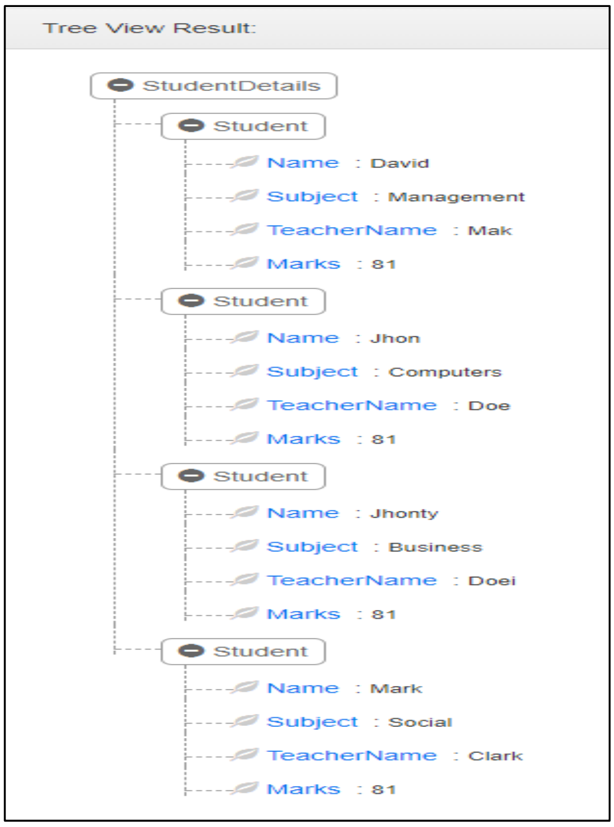
$\pi_{\text{Name, Subject, TeacherName, Marks}}$ Student

Student.Name	Student.Subject	Student.TeacherName	Student.Marks
John	Management	Doe	81
David	Management	Martin	90
Ricky	Management	Miller	89
Jhonty	Management	Andrew	71

Figure 5. RA tree for select query in MS SQL Server

The XML tree and code are represented as follows.

```
<StudentDetails>
  <Student>
    <Name>David</Name>
    <Subject>Management</Subject>
    <TeacherName>Mak</TeacherName>
    <Marks>81</Marks>
  </Student>
  <Student>
    <Name>Jhon</Name>
    <Subject>Computers</Subject>
    <TeacherName>Doe</TeacherName>
    <Marks>81</Marks>
  </Student>
  <Student>
    <Name>Jhonty</Name>
    <Subject>Business</Subject>
    <TeacherName>Doei</TeacherName>
    <Marks>81</Marks>
  </Student>
  <Student>
    <Name>Mark</Name>
    <Subject>Social</Subject>
    <TeacherName>Clark</TeacherName>
    <Marks>81</Marks>
  </Student>
</StudentDetails>
```



```
Tree View Result:
StudentDetails
  Student
    Name : David
    Subject : Management
    TeacherName : Mak
    Marks : 81
  Student
    Name : Jhon
    Subject : Computers
    TeacherName : Doe
    Marks : 81
  Student
    Name : Jhonty
    Subject : Business
    TeacherName : Doei
    Marks : 81
  Student
    Name : Mark
    Subject : Social
    TeacherName : Clark
    Marks : 81
```

The algorithms that comply with conversion include common compositions and can be summarised as the following.

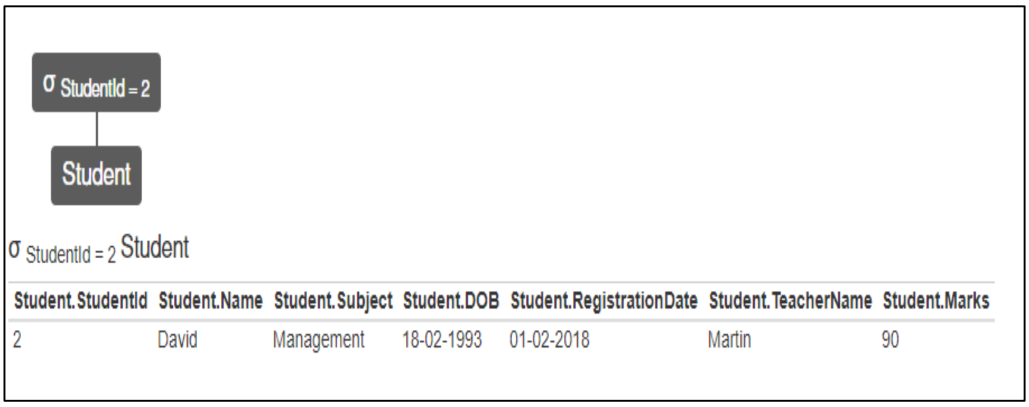
Query	Relational algebra
select Job from Fin	$\textbf{PROJECT}_{\text{job}}(\text{Fin})$
select fy, job from Fin	$\textbf{PROJECT}_{\text{fy, job}}(\text{Fin})$
select * from Stu where Num < 300	$\textbf{SELECT}_{\text{Num} < 300}(\text{stu})$
select * from Rest where Pl < 30 and num >= 15	$\textbf{SELECT}_{\text{Pl} < 30 \text{ and num} >= 15}(\text{rest})$

select stu.name
from fu Ra, Se Stu
where Ra.Fun = Stu.ref
and ra.Title = "hospital"
	$\textbf{PROJECT}_{\text{stu.name}}$ ([$\textbf{SELECT}_{\text{ra.Title= "hospital"}}$($\textbf{RENAME}_{\text{Ra(raref, ratitle, Ra.Fun)}}$(Fin))] $\textbf{JOIN}_{\text{RaFun = Sturef}}$ [$\textbf{RENAME}_{\text{STU(Sturef, stu.name, Ra.Fun)}}$(fin)])
	
	or, in a shortage way,
	
	$\textbf{PROJECT}_{\text{stu.name}}$([$\textbf{SELECT}_{\text{ra.Title = "hospital"}}$ ($\textbf{RENAME}_{\text{Ra}}$(Fin))] $\textbf{JOIN}_{\text{Ra.Fun = Stu.ref}}$ [$\textbf{RENAME}_{\text{STU}}$(Fin)])
	
	or
	
	Ra <- $\textbf{RENAME}_{\text{Ra(raref, raTitle, Rafun)}}$(fin)
	STU <- $\textbf{RENAME}_{\text{STU(Sturef, stuname, stufun)}}$(fin)
	J <- $\textbf{SELECT}_{\text{title = "hosiptal"}}$(Ra)
	C <- J $\textbf{JOIN}_{\text{Rafun = sturef}}$ STU
	R <- $\textbf{PROJECT}_{\text{stu.name}}$(C)

- Update query:

```
update Student set Name='Andrew' ,marks=84 where StudentId=2
```

The RA expression and tree is as follows.



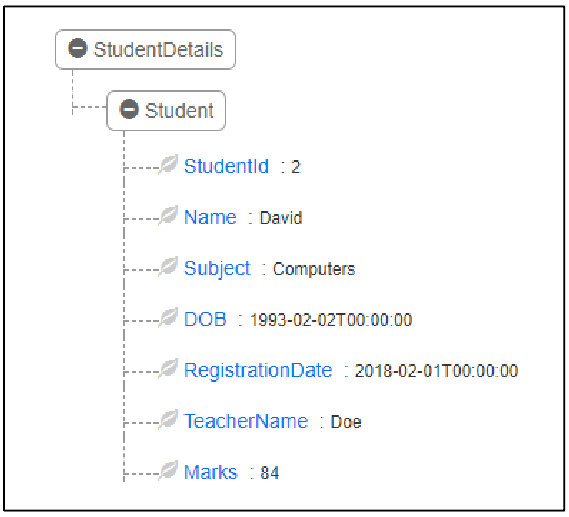
$\sigma_{\text{StudentId} = 2}$ Student

Student.StudentId	Student.Name	Student.Subject	Student.DOB	Student.RegistrationDate	Student.TeacherName	Student.Marks
2	David	Management	18-02-1993	01-02-2018	Martin	90

Figure 6. RA tree for update query in MS SQL Server

The XML is given below.

```
<StudentDetails>
  <Student>
    <StudentId>2</StudentId>
    <Name>David</Name>
    <Subject>Computers</Subject>
    <DOB>1993-02-02T00:00:00</DOB>
    <RegistrationDate>2018-02-
01T00:00:00</RegistrationDate>
    <TeacherName>Doe</TeacherName>
    <Marks>84</Marks>
  </Student>
</StudentDetails>
```

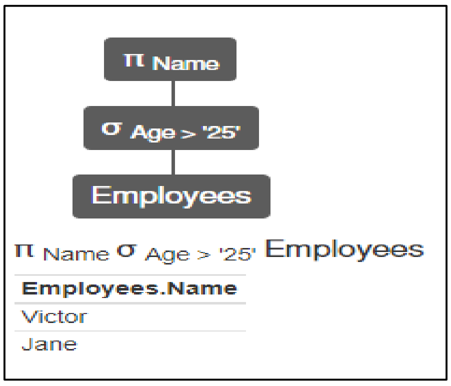


4.2.3 MYSQL

- The following query was extracted from the database as a SELECT query with different compositions.

```
SELECT Name FROM User WHERE Age > 25;
```

The RA expression and tree representation is as follows.



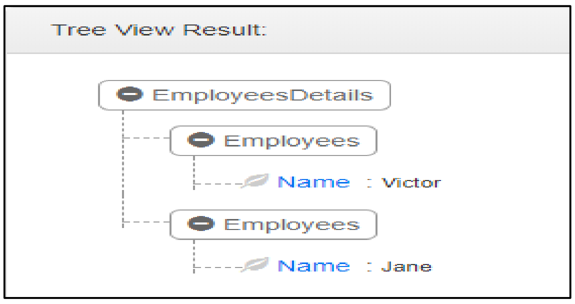
π Name σ Age > '25' Employees

Employees.Name
Victor
Jane

Figure 7. RA tree for select query in MY SQL

The XML representation and code is given below.

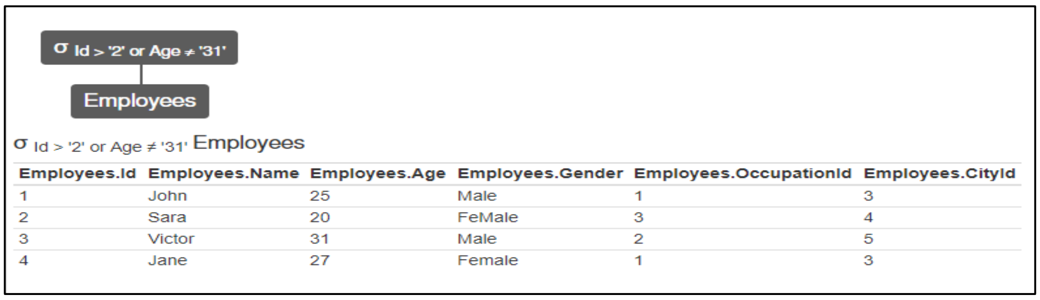
```xml
<EmployeesDetails>
<Employees>
<Name>Victor</Name>
</Employees>

<Employees>
<Name>Jane</Name>
</Employees>
</EmployeesDetails>
```



- The second form of select query is given as follows:

```
SELECT * FROM User WHERE id>2 OR Age != 31;
```

with the following RA expression and tree:



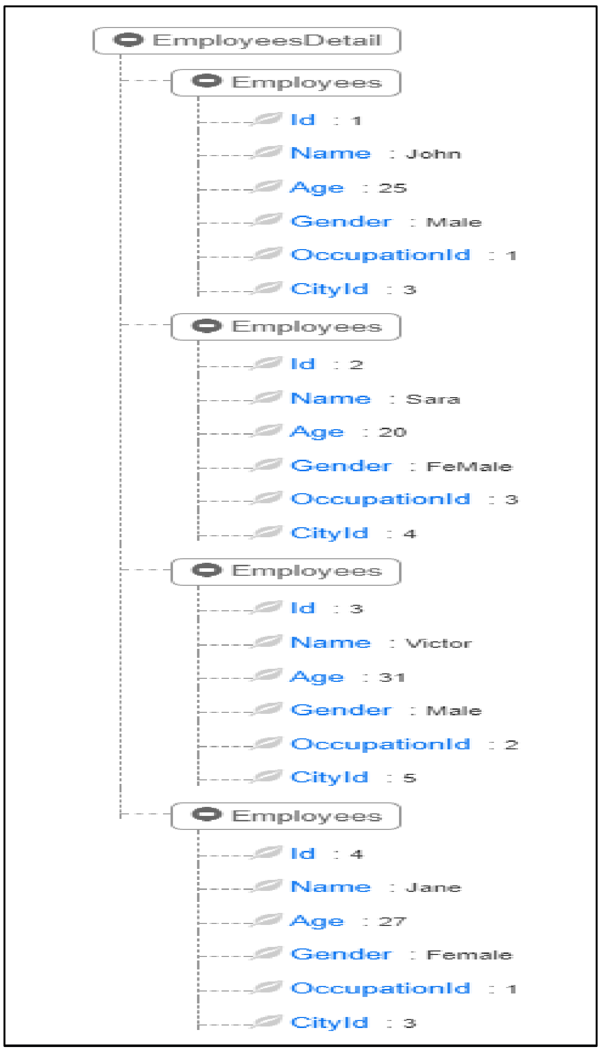
σ Id > '2' or Age ≠ '31' Employees

Employees.Id	Employees.Name	Employees.Age	Employees.Gender	Employees.OccupationId	Employees.CityId
1	John	25	Male	1	3
2	Sara	20	FeMale	3	4
3	Victor	31	Male	2	5
4	Jane	27	Female	1	3

Figure 8. RA tree for second select query in MY SQL

The corresponding XML code and tree for second query are:

```xml
<EmployeesDetail>
<Employees>
<Id>1</Id>
<Name>John</Name>
<Age>25</Age>
<Gender>Male</Gender>
<OccupationId>1</OccupationId>
<CityId>3</CityId>
</Employees>
<Employees>
<Id>2</Id>
<Name>Sara</Name>
<Age>20</Age>
<Gender>FeMale</Gender>
<OccupationId>3</OccupationId>
<CityId>4</CityId>
</Employees>
<Employees>
<Id>3</Id>
<Name>Victor</Name>
<Age>31</Age>
<Gender>Male</Gender>
<OccupationId>2</OccupationId>
<CityId>5</CityId>
</Employees>
<Employees>
<Id>4</Id>
<Name>Jane</Name>
<Age>27</Age>
<Gender>Female</Gender>
<OccupationId>1</OccupationId>
<CityId>3</CityId>
</Employees>
</EmployeesDetail>
```

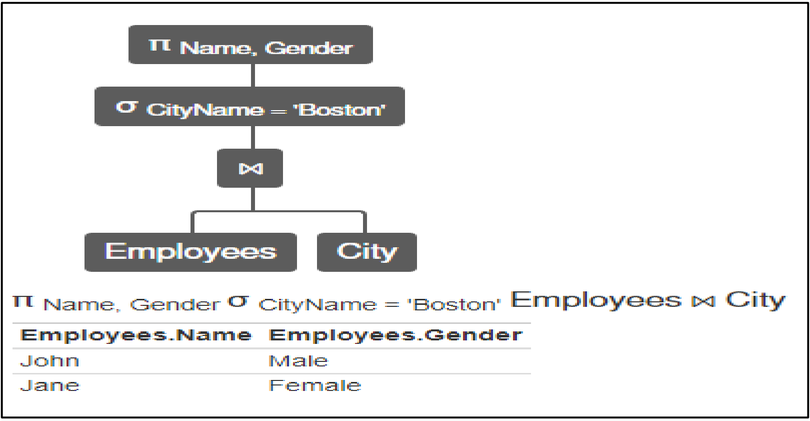


```
EmployeesDetail
  Employees
    Id : 1
    Name : John
    Age : 25
    Gender : Male
    OccupationId : 1
    CityId : 3
  Employees
    Id : 2
    Name : Sara
    Age : 20
    Gender : FeMale
    OccupationId : 3
    CityId : 4
  Employees
    Id : 3
    Name : Victor
    Age : 31
    Gender : Male
    OccupationId : 2
    CityId : 5
  Employees
    Id : 4
    Name : Jane
    Age : 27
    Gender : Female
    OccupationId : 1
    CityId : 3
```

- The third form of select query is

SELECT Name, Gender **FROM** User NATURAL JOIN City **WHERE** CityName = ”Boston”;

This query is represented in RA as



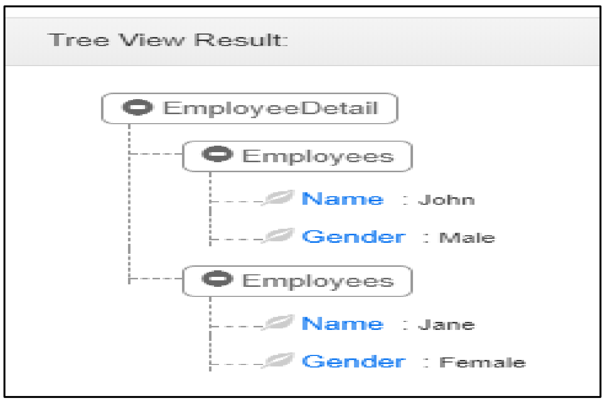
$\pi_{\text{Name, Gender}}\ \sigma_{\text{CityName = 'Boston'}}\ \text{Employees} \bowtie \text{City}$

Employees.Name	Employees.Gender
John	Male
Jane	Female

Figure 9. RA tree for third select query in MY SQL

The corresponding XML tree and code is represented as below:

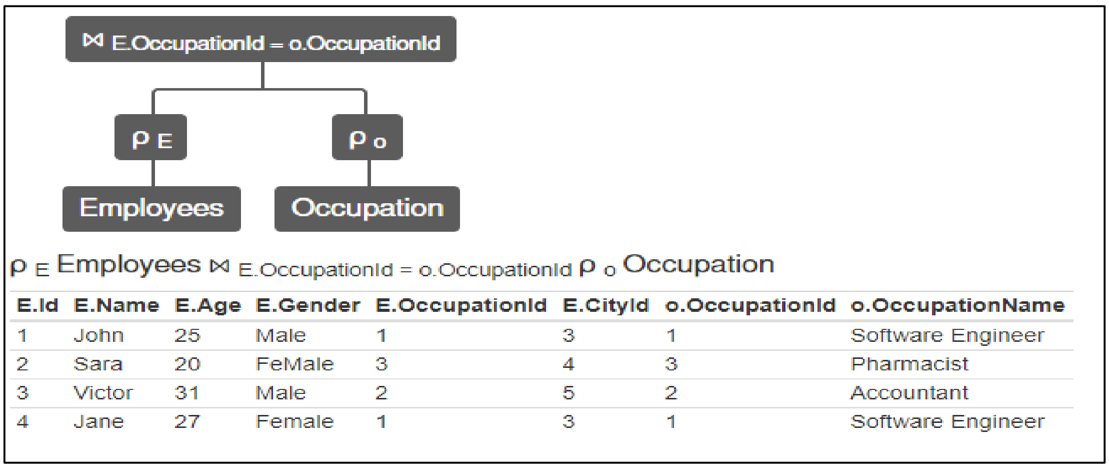
```xml
<EmployeeDetail>
<Employees>
<Name>John</Name>
<Gender>Male</Gender>
</Employees>
<Employees>
<Name>Jane</Name>
<Gender>Female</Gender>
</Employees>
</EmployeeDetail>
```



- The fourth form of select query is

SELECT * **FROM** Employees **as** E **join** Occupation **as** o on E.OccupationId = o.OccupationId

With a RA representation of



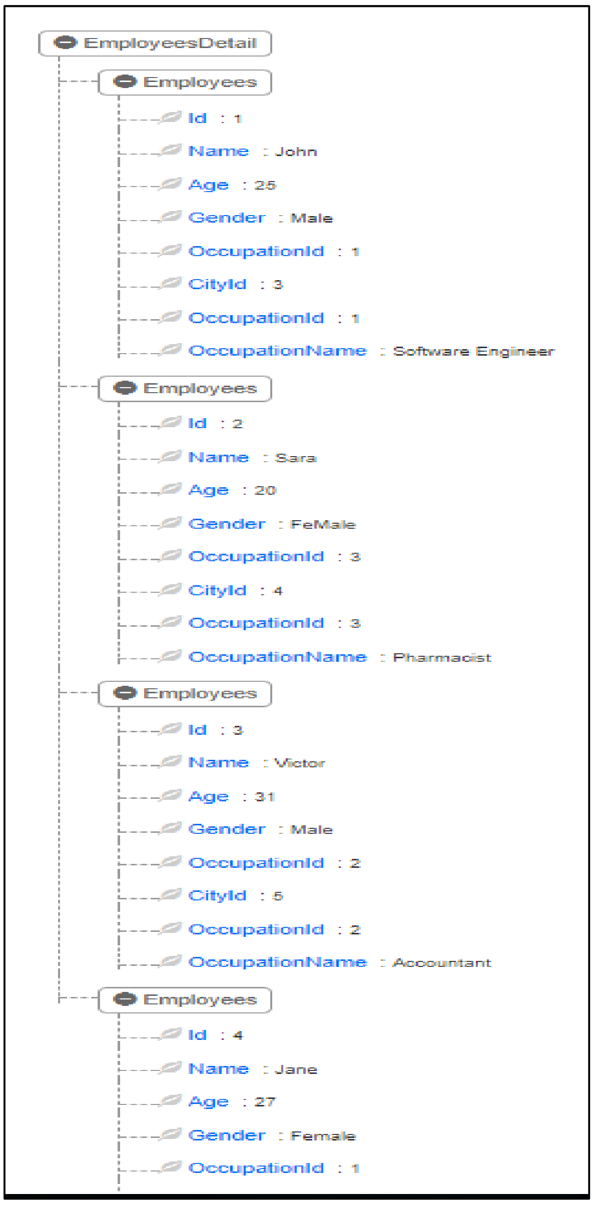
ρ_{E} Employees $\bowtie_{E.OccupationId = o.OccupationId}$ ρ_{o} Occupation

E.Id	E.Name	E.Age	E.Gender	E.OccupationId	E.CityId	o.OccupationId	o.OccupationName
1	John	25	Male	1	3	1	Software Engineer
2	Sara	20	FeMale	3	4	3	Pharmacist
3	Victor	31	Male	2	5	2	Accountant
4	Jane	27	Female	1	3	1	Software Engineer

Figure 10. RA tree for fourth select query in MY SQL

corresponding XML code and tree for fourth query are as follows.

```
<EmployeesDetail>
<Employees>
  <Id>1</Id>
  <Name>John</Name>
  <Age>25</Age>
  <Gender>Male</Gender>
  <OccupationId>1</OccupationId>
  <CityId>3</CityId>
  <OccupationId>1</OccupationId>
  <OccupationName>Software
Engineer</OccupationName>
</Employees>
<Employees>
  <Id>2</Id>
  <Name>Sara</Name>
  <Age>20</Age>
  <Gender>FeMale</Gender>
  <OccupationId>3</OccupationId>
  <CityId>4</CityId>
  <OccupationId>3</OccupationId>

<OccupationName>Pharmacist</OccupationName>
</Employees>
<Employees>
  <Id>3</Id>
  <Name>Victor</Name>
  <Age>31</Age>
  <Gender>Male</Gender>
  <OccupationId>2</OccupationId>
  <CityId>5</CityId>
  <OccupationId>2</OccupationId>

<OccupationName>Accountant</OccupationName>
</Employees>
<Employees>
  <Id>4</Id>
  <Name>Jane</Name>
  <Age>27</Age>
  <Gender>Female</Gender>
  <OccupationId>1</OccupationId>
```

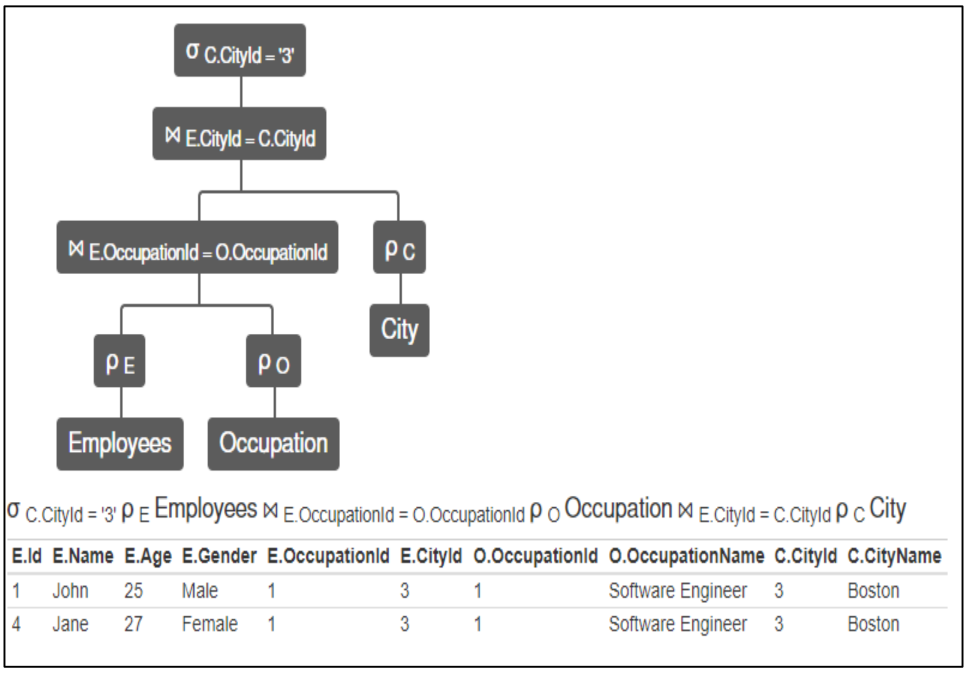


```
EmployeesDetail
  Employees
    Id : 1
    Name : John
    Age : 25
    Gender : Male
    OccupationId : 1
    CityId : 3
    OccupationId : 1
    OccupationName : Software Engineer
  Employees
    Id : 2
    Name : Sara
    Age : 20
    Gender : FeMale
    OccupationId : 3
    CityId : 4
    OccupationId : 3
    OccupationName : Pharmacist
  Employees
    Id : 3
    Name : Victor
    Age : 31
    Gender : Male
    OccupationId : 2
    CityId : 5
    OccupationId : 2
    OccupationName : Accountant
  Employees
    Id : 4
    Name : Jane
    Age : 27
    Gender : Female
    OccupationId : 1
```

- The update query

```
update employees set OccupationId=1 where Id=3
select * from Employees as E
join Occupation as O on E.OccupationId=O.OccupationId
join City as C on E.CityId=C.CityId
where C.CityId ='3'
```

It has the following RA representation,



$\sigma_{\text{C.CityId = '3'}}\ \rho_{\text{E}}\ \text{Employees} \bowtie_{\text{E.OccupationId = O.OccupationId}} \rho_{\text{O}}\ \text{Occupation} \bowtie_{\text{E.CityId = C.CityId}} \rho_{\text{C}}\ \text{City}$

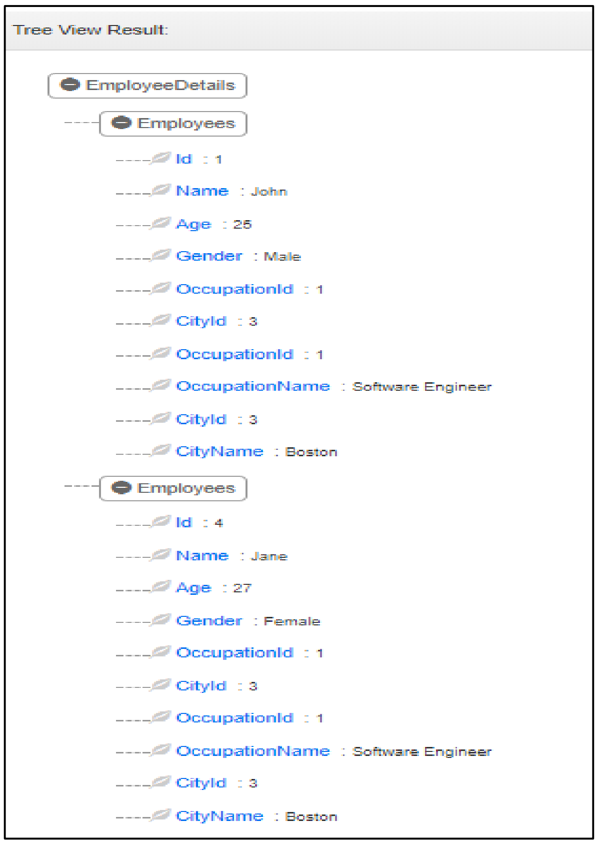
E.Id	E.Name	E.Age	E.Gender	E.OccupationId	E.CityId	O.OccupationId	O.OccupationName	C.CityId	C.CityName
1	John	25	Male	1	3	1	Software Engineer	3	Boston
4	Jane	27	Female	1	3	1	Software Engineer	3	Boston

Figure 11. RA tree for update query in MY SQL

.

XML code and tree representation are:

```
<EmployeeDetails>
<Employees>
  <Id>1</Id>
  <Name>John</Name>
  <Age>25</Age>
  <Gender>Male</Gender>
  <OccupationId>1</OccupationId>
  <CityId>3</CityId>
  <OccupationId>1</OccupationId>
  <OccupationName>Software
Engineer</OccupationName>
  <CityId>3</CityId>
  <CityName>Boston</CityName>
</Employees>
<Employees>
  <Id>4</Id>
  <Name>Jane</Name>
  <Age>27</Age>
  <Gender>Female</Gender>
  <OccupationId>1</OccupationId>
  <CityId>3</CityId>
  <OccupationId>1</OccupationId>
  <OccupationName>Software
Engineer</OccupationName>
  <CityId>3</CityId>
  <CityName>Boston</CityName>
</Employees>
  </EmployeeDetails>
```



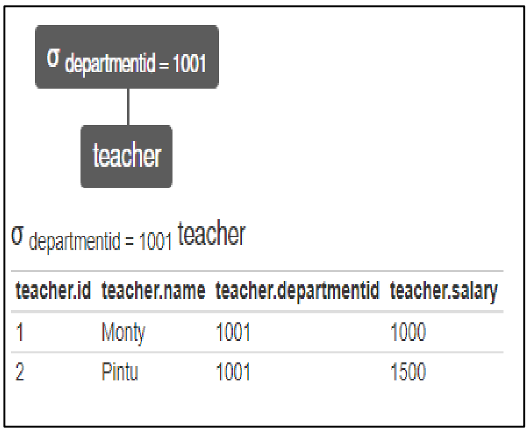
```
Tree View Result:

EmployeeDetails
----Employees
    ----Id : 1
    ----Name : John
    ----Age : 25
    ----Gender : Male
    ----OccupationId : 1
    ----CityId : 3
    ----OccupationId : 1
    ----OccupationName : Software Engineer
    ----CityId : 3
    ----CityName : Boston
----Employees
    ----Id : 4
    ----Name : Jane
    ----Age : 27
    ----Gender : Female
    ----OccupationId : 1
    ----CityId : 3
    ----OccupationId : 1
    ----OccupationName : Software Engineer
    ----CityId : 3
    ----CityName : Boston
```

4.2.4 Oracle

- The SELECT query is given as:

select * **from** teacher **where** departmentid=1001

It has the following RA representation:



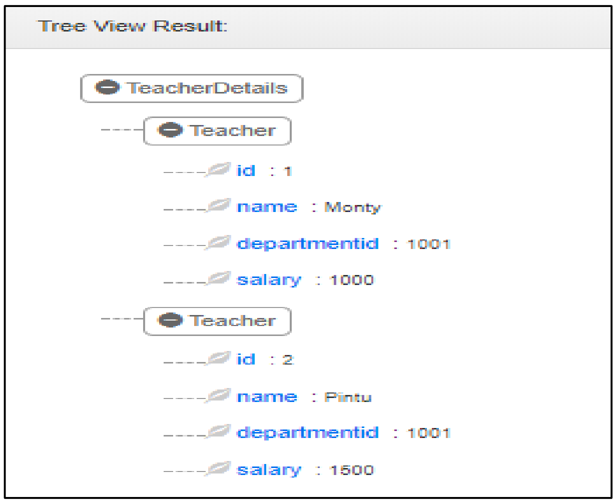
$\sigma_{departmentid = 1001}$ teacher

teacher.id	teacher.name	teacher.departmentid	teacher.salary
1	Monty	1001	1000
2	Pintu	1001	1500

Figure 12. RA tree for select query in Oracle

The XML representation is given below:

```
<Teacher>
<id>1</id>
<name>Monty</name>
<departmentid>1001</departmentid>
<salary>1000</salary>
</Teacher>
<Teacher>
<id>2</id>
<name>Pintu</name>
<departmentid>1001</departmentid>
<salary>1500</salary>
       </Teacher>
```

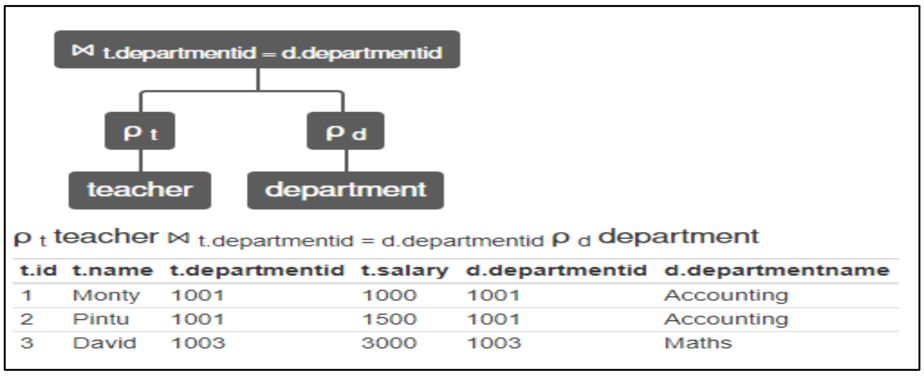


```
Tree View Result:

TeacherDetails
----Teacher
    ----id : 1
    ----name : Monty
    ----departmentid : 1001
    ----salary : 1000
----Teacher
    ----id : 2
    ----name : Pintu
    ----departmentid : 1001
    ----salary : 1500
```

- The DELETE query:

```
delete from teacher where teacherid=2
 select * from teacher as t join department as d on t.departmentid    =d.departmentid
```

The RA representation is as follow:

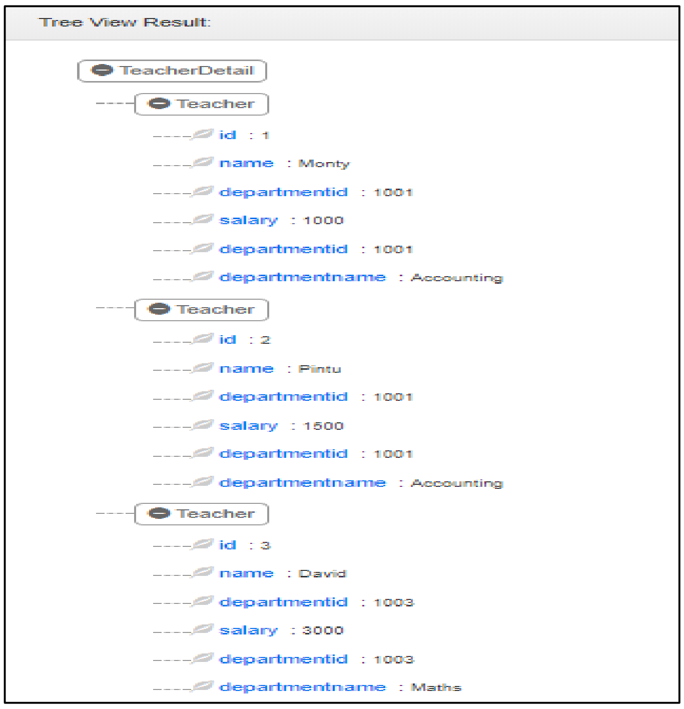


ρ t teacher ⋈ t.departmentid = d.departmentid ρ d department

t.id	t.name	t.departmentid	t.salary	d.departmentid	d.departmentname
1	Monty	1001	1000	1001	Accounting
2	Pintu	1001	1500	1001	Accounting
3	David	1003	3000	1003	Maths

Figure 13. RA tree for delete query in Oracle

XML representation is given below:

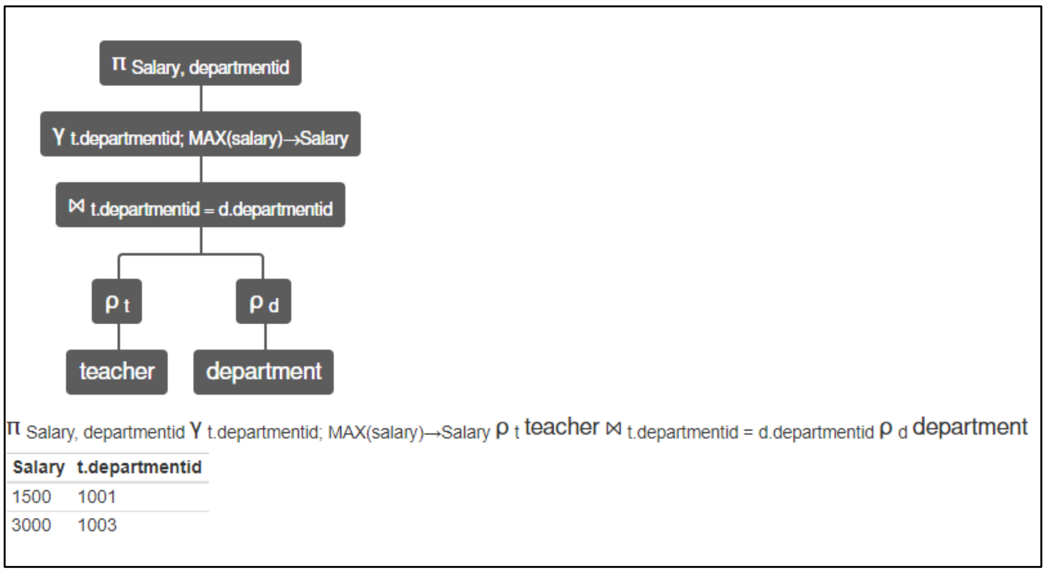


```
TeacherDetail
----Teacher
    ----id : 1
    ----name : Monty
    ----departmentid : 1001
    ----salary : 1000
    ----departmentid : 1001
    ----departmentname : Accounting
----Teacher
    ----id : 2
    ----name : Pintu
    ----departmentid : 1001
    ----salary : 1500
    ----departmentid : 1001
    ----departmentname : Accounting
----Teacher
    ----id : 3
    ----name : David
    ----departmentid : 1003
    ----salary : 3000
    ----departmentid : 1003
    ----departmentname : Maths
```

- The SELECT query with a JOIN function is given as

```
select max(salary) as Salary, departmentid from teacher as t join department
as d on t.departmentid=d.departmentid
group by t.departmentid
```

The RA representation is as follow:


π Salary, departmentid
γ t.departmentid; MAX(salary)→Salary
⋈ t.departmentid = d.departmentid
ρ t
ρ d
teacher
department

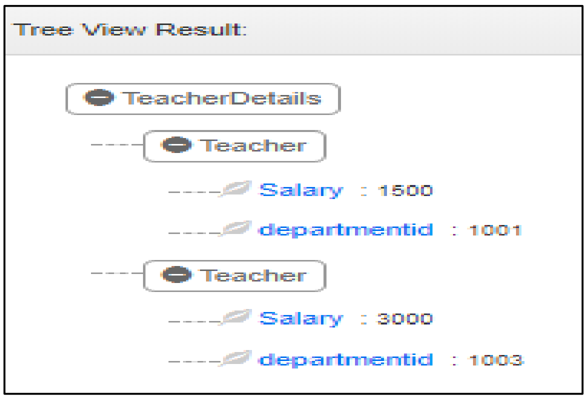
$\pi_{\text{Salary, departmentid}}\ \gamma_{\text{t.departmentid; MAX(salary)}\rightarrow\text{Salary}}\ \rho_{t}\ \text{teacher} \bowtie_{\text{t.departmentid = d.departmentid}} \rho_{d}\ \text{department}$

Salary	t.departmentid
1500	1001
3000	1003

Figure 14. RA tree for select query with join in Oracle

and the following is XML representation.

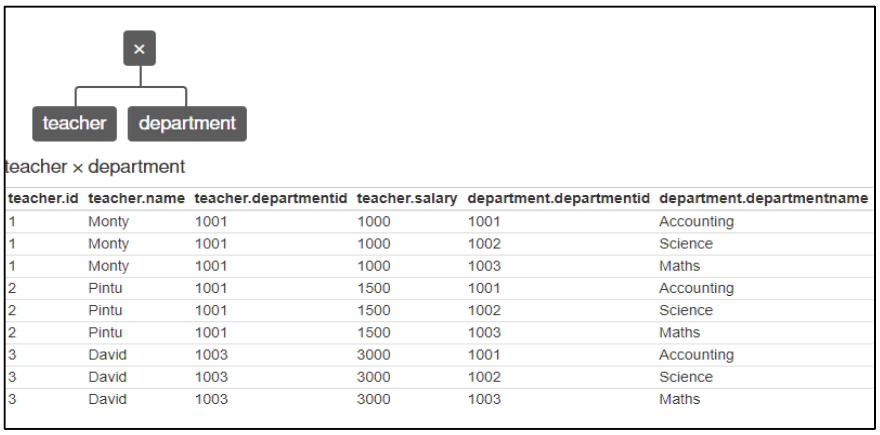
```
<TeacherDetails>
<Teacher>
  <Salary>1500</Salary>
  <departmentid>1001</departmentid>
</Teacher>
<Teacher>
  <Salary>3000</Salary>
  <departmentid>1003</departmentid>
</Teacher>
</TeacherDetails>
```



- The SELECT query with CROSS and JOIN functions is given as

```
select * from teacher cross join department
```

The RA representation is:



teacher × department

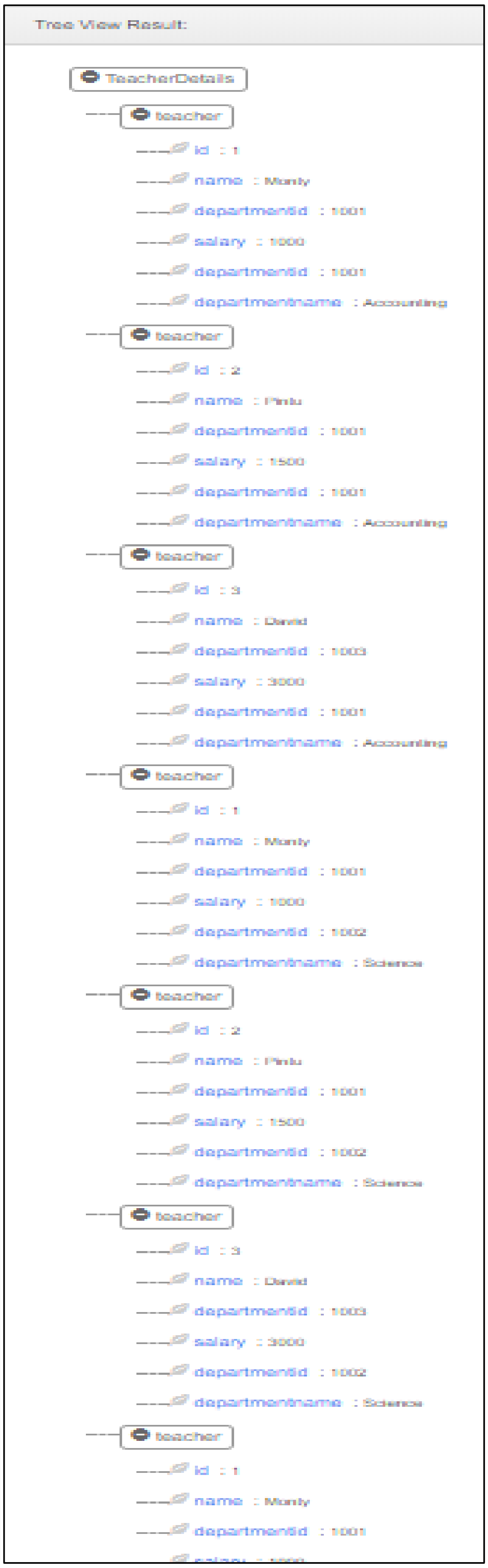
teacher.id	teacher.name	teacher.departmentid	teacher.salary	department.departmentid	department.departmentname
1	Monty	1001	1000	1001	Accounting
1	Monty	1001	1000	1002	Science
1	Monty	1001	1000	1003	Maths
2	Pintu	1001	1500	1001	Accounting
2	Pintu	1001	1500	1002	Science
2	Pintu	1001	1500	1003	Maths
3	David	1003	3000	1001	Accounting
3	David	1003	3000	1002	Science
3	David	1003	3000	1003	Maths

Figure 15. RA tree for select query with cross and join in Oracle

the following is XML representation

```
<TeacherDetails>
       <teacher>
  <id>1</id>
  <name>Monty</name>
  <departmentid>1001</departmentid>
  <salary>1000</salary>
  <departmentid>1001</departmentid>
  <departmentname>Accounting</departmentname>
</teacher>
<teacher>
  <id>2</id>
  <name>Pintu</name>
  <departmentid>1001</departmentid>
  <salary>1500</salary>
  <departmentid>1001</departmentid>
  <departmentname>Accounting</departmentname>
</teacher>
<teacher>
  <id>3</id>
  <name>David</name>
  <departmentid>1003</departmentid>
  <salary>3000</salary>
  <departmentid>1001</departmentid>
  <departmentname>Accounting</departmentname>
</teacher>
<teacher>
  <id>1</id>
  <name>Monty</name>
  <departmentid>1001</departmentid>
  <salary>1000</salary>
  <departmentid>1002</departmentid>
  <departmentname>Science</departmentname>
</teacher>
<teacher>
  <id>2</id>
  <name>Pintu</name>
  <departmentid>1001</departmentid>
  <salary>1500</salary>
  <departmentid>1002</departmentid>
  <departmentname>Science</departmentname>
</teacher>
```

```
<teacher>
  <id>3</id>
  <name>David</name>
  <departmentid>1003</departmentid>
  <salary>3000</salary>
  <departmentid>1002</departmentid>
  <departmentname>Science</departmentname>
</teacher>
<teacher>
  <id>1</id>
  <name>Monty</name>
  <departmentid>1001</departmentid>
  <salary>1000</salary>
  <departmentid>1003</departmentid>
  <departmentname>Maths</departmentname>
</teacher>
<teacher>
  <id>2</id>
  <name>Pintu</name>
  <departmentid>1001</departmentid>
  <salary>1500</salary>
  <departmentid>1003</departmentid>
  <departmentname>Maths</departmentname>
</teacher>
<teacher>
  <id>3</id>
  <name>David</name>
  <departmentid>1003</departmentid>
  <salary>3000</salary>
  <departmentid>1003</departmentid>
  <departmentname>Maths</departmentname>
</teacher>
</TeacherDetails>
```



Tree View Result:

```
TeacherDetails
  teacher
    id : 1
    name : Monty
    departmentid : 1001
    salary : 1000
    departmentid : 1001
    departmentname : Accounting
  teacher
    id : 2
    name : Pintu
    departmentid : 1001
    salary : 1500
    departmentid : 1001
    departmentname : Accounting
  teacher
    id : 3
    name : David
    departmentid : 1003
    salary : 3000
    departmentid : 1001
    departmentname : Accounting
  teacher
    id : 1
    name : Monty
    departmentid : 1001
    salary : 1000
    departmentid : 1002
    departmentname : Science
  teacher
    id : 2
    name : Pintu
    departmentid : 1001
    salary : 1500
    departmentid : 1002
    departmentname : Science
  teacher
    id : 3
    name : David
    departmentid : 1003
    salary : 3000
    departmentid : 1002
    departmentname : Science
  teacher
    id : 1
    name : Monty
    departmentid : 1001
```

The SELECT query can be combined with the aggregation function to discard duplicates by ignoring null values.

SQL	Relational algebra
select sum(marks) **from** D	$F_{sum(marks)}(D)$
select count(marks) **from D**	Fcount(Marks)(D)
select count(distinct mark) **from D**	Fcount(mark)(PROJECTmark(D))

However, aggregation can be grouped as follows.

SQL	Relational algebra
select sum(mark) from d group by aps	${}_{aps}F_{sum(mark)}(D)$
select sum(mark), count(*) from D group by aps	${}_{aps}F_{sum(mark), count(*)(D)}$

CHAPTER FIVE: CONCLUSION

5.1 Conclusion

This study identified the types of data that has to be preserved in the long term and proposed a methodology for preserving queries. The majority of databases satisfy long-term preservation of data. However, RDBs do not have only data, they are composed of many components such as queries, pages, metadata and so forth. Thus, this research proposed a methodology for preserving queries. The proposed approach was based on representation of all data-manipulating languages among RDBs as XML files, which preserve relational algebra trees for queries as XML queries. This simplifies the data retrieval process, regardless of when initial data preservation took place, because data associated with these queries can be read or extracted without using the original language standards. This research also provided a small case study using the MS Access database to retrieve an SQL query and convert it into relational algebra presented as an XML tree.

The data collection analysis revealed that database management depends on SQL support that constrains and restricts feature and function capabilities. The SQL dependency of databases means that SQL is a predominant query language defined and recognised in the applications' cores and the database optimisation process. This research proposes freeing databases of SQL dependency to exploit the unique features of each database's functionality for long-term data preservation.

Database management translates SQL declaratives as subsets of procedural operations similar to relational procedural operation series. Based on this, the research suggests that query optimisation could be enhanced by converting SQL statements into relational algebra. Additionally, relational algebra operators are commonly recognised and defined by both users and devices. This will give databases the rationality-complete features that have been restricted by SQL support in each database. Thus, this research attempts to normalise query support for the various features of databases by interpreting queries as relational algebra trees in XML format.

XML was chosen as a long-term data-preserving format due to several different features of query storage and archiving. These make XML capable of being adapted regardless of any preservation strategy. XML is a simple and normalised format for data exchange, regardless of software, format, or time. This proposed approach supports long-term preservation of RDBs where queries play a vital role in the database management process. It will also enhance storage-space utilisation and reduces the possibility for inconsistent data.

5.2 Future Work

Future work will contribute to this approach by implementing the strategy in large RDBs to evaluate the retrieval process over a period of time, including the processing time required to retrieve saved queries compared to the time required for newly written queries. Future studies should look for new kinds of data-preservation methods that align with data preservation in RDBs or that provide the best performance in combination with the XML format. However, future research should also investigate the differences in SQL compliance standards among RDBMSs and how to overcome these differences using relational algebra. Additionally, an automated version of the current study rather than manual conversion should be explored to facilitate the process and save time; automation can be integrated with database management and any improvements can be observes. Finally, future research should investigate alternatives to XML, such as JSON, to determine suitability for long-term data preservation.

REFERENCES

Acronics. (2008). *MS SQL Server: An overview*. Available at: http://hosteddocs.ittoolbox.com/AcronisSQL012308.pdf. (Accessed: 23 August 2017).

Amano, S., David, C., Libkin, L. and Murlak, F. (2014). 'XML schema mappings', *Journal of the ACM*, 61(2), pp.1–48.

Ashraf, T. & Kumar, N. (2016). *Interdisciplinary digital preservation tools and technologies*, 1st edn. Hershey, PA, USA: IGI Global.

Backus, J., Cartolano, R., Drummond, C., Gebert, A., Hanson, B., Hilton, J., Martone, M., Michalak, S., Ovenden, R., Pritchard, S. and Scheman, R. (2016). Report from the repositories and preservation workgroup, *Proceedings of the Open Scholarship Initiative*, Location: George Mason University, 1, pp. 2–6.

Ball, A. (2006). *Briefing paper: File format and XML scheme registries*. Available at: http://www.ukoln.ac.uk/projects/grand-challenge/papers/registryBriefing.pdf. (Accessed: 27 September 2017).

Beaumont, R. (2011). *Introduction to LibreOffice Base (LOB)-6: Queries using multiple tables*. Available at: http://www.floppybunny.org/robin/web/virtualclassroom/chap8/libreoffice/libreoffice_base_tut6.pdf. (Accessed: 1 August 2018).

Blaha, M. (2005). *Referential integrity is important for databases*. Available at: http://www.odbms.org/wp-content/uploads/2005/11/007.02-Blaha-Referential-Integrity-Is-Important-For-Databases-November-2005.pdf. (Accessed: 16 August 2017).

Bourgeois, D. (2014). Data and databases, *Information system for business and beyond*. Washington, DC, USA: Sayllor Academy Open Textbook Publisher.

Chen, Y., Davidson, S., Hara, C. and Zheng, Y. (2003). -RRXS: Redundancy reducing XML storage in relations, *Proceedings of the 2003 VLDB Conference,* pp. 189–200.

Cox, P. (2009). 'Action queries: Manipulating your data', *Strategic Finance,* 91.1, pp. 56–57.

Date, C. (2013). *Database design and relational theory*. Sebastopol, CA, USA: O'Reilly.

Date, C.J. (2014). *Introduction to database system*, 8th edn. Pearson.

Delaney, K. Randal, P., Tripp, K.L., Cunningham, C., Machanic, A. and Nevarez, B. (2009). *Microsoft SQL Server 2008 internals*. Redmond, Washington, USA: Microsoft Press.

Dignös, A., Böhlen, M., Gamper, J. and Jensen, C. (2016). 'Extending the kernel of a relational DBMS with comprehensive support for sequenced temporal queries', *ACM Transactions on Database Systems*, 41(4), pp.1–46.

Elhai, J., Levine, J. and Hall, B. (2017). 'Anxiety about electronic data hacking', *Internet Research*, 27(3), pp. 631–649.

Franchitti, J.C. (2014). Relational algebra, relational calculus, and SQL: Presentation based on textbook slides, in Ramez E. and Shamkant N. (eds.) *Fundamentals of database systems*, 7th edn. New York University, pp. 7–130.

Freitas, R.A.P. and Ramalho, J.C. (2010). Significant properties in the preservation of relational databases, *Proceedings of the Research and Advanced Technology for Digital Libraries*, *14th European Conference*, Glasgow, UK. September 2010. Springer.

Gordon, L.C. and Chaczko, Z. (2015). Digital patterns for heritage and data preservation standards, *Computational Intelligence and Efficiency in Engineering Systems*. Switzerland: Springer International Publishing, pp. 47–59.

Gorsel, M.V., Leenaars, M., Milic-Frayling, N. and Palm, J. (2014). Evaluation and strategies of digital preservation and UNESCO's role in facing the technical challenges, *Proceedings of the 2nd Annual Conference of the ICA*, Girona. October 2014

Halvorsen, H.P. (2016). Structured query language. Available at: http://home.hit.no/~hansha/documents/database/documents/Structured%20Query%20Language.pdf. (Accessed: 24 August 2017).

Hardesty, J. (2016). 'Transitioning from XML to RDF: Considerations for an effective move towards linked data and the semantic web', *Information Technology and Libraries*, 35(1), p. 51.

Hedstrom, M., Dawes, S., Fleischhuer, C., Gray, J., Lynch, C., Mccrary, V., Moore, R., Thibodeau, K. and Waters, D. (2002). 'It's about time: Research challenges in digital archiving and long term preservation', *Preserving our digital heritage plan for national digital information infrastructure and preservation program*, USA, Congress Library, pp. 205–220.

Hellerstein, J., Stonebraker, M. and Hamilton, J. (2007). 'Architecture of a database system', *Foundations and Trends in Databases*, 1(2), pp.141–259.

Hunter, D., Watt, A., Dukkett, J., Ayres, D., Chase, N., Fawcett, J., Gaven, T. and Patterson, B.

(2004). *Beginning XML*. USA: Willey.

Hunter, J. and Choudhury, S. (2003). Implementing preservation strategies for complex multimedia objects, *Proceedings of the International Conference on Theory and Practice of Digital Libraries ECDL*, Norway, Trondheim. August 2003. Berlin, Heidelberg: Springer, pp. 473–486.

IBM Knowledge Centre. (2017). *Database design with denormalization*. Available at: https://www.ibm.com/support/knowledgecenter/en/SSEPEK_10.0.0/intro/src/tpc/db2z_denormalizationforperformance.html . (Accessed: 4 July 2017).

IBM. (2010). ‘Proven strategies for archiving complex relational data’, (white paper). USA: IBM Software Thought Leadership.

IBM. (2013). *Database performance and query optimization, IBM i Version 7.2, Licensed internal code*. IBM Corp.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). An introduction to statistical learning, *Springer Texts in Statistics*. New York: springer.

Jarke, M., Mylopoulos, J., Quix, C.H., Rolland, C., Manoplopoulos, Y., Mouratidis, H. and Horkoff, J. (2014). Advanced information system engineering, *Proceedings of the 26th International Conference CAiSE*, Greece, 16–20 June 2014. Springer.

Kender, R. (2016). Most popular relational databases (2016 edition). Available at: https://dzone.com/articles/most-popular-relational-databases-2016-edition. (Accessed: 2 August 2018).

Kolahi, S. (2008). *Design guidelines for reducing redundancy in relational and XML data*. PhD. University of Toronto.

Koopman, M. and de Jager, K. (2016). ‘Archiving South African digital research data: How ready are we?’, *South African Journal of Science*, 112(7/8), pp.1– 6

Kremser, J., Kovacova, B., Pavlovska, M., Hejtmanek, L. and Antos, D. (2012). *Long term preservation of digital data—Background research*, (technical report) Czech Republic: CESNET.

Kumar, A. (2014). *Digital signal processing*. 2nd edn. New Delhi, Delhi: PHI Learning Pvt. Ltd.

Kytes, T. (2005). *Expert Oracle*. signature edn. USA: Springer.

Lake, P. and Crowther, P. (2013). *Concise guide to databases*, London: Springer.

Lauesen, S. (2011). *Microsoft-Access tutorial*. Available at:

https://www.itu.dk/~slauesen/UID/AccessTutorial.pdf. (Accessed: 22 August 2017).

Lemons, M. (2016). *Microsoft Access 2016*. Available at: https://dit.ie/media/ittraining/msoffice/MOAC_Access_2016.pdf. (Accessed: 22 August 2017).

Lindley, A. (2013). Database preservation evaluation report: SIARD vs. CHRONOS. *Proceedings of* the *10th International Conference on Preservation of Digital Objects*. Lisbon. 2 September 2013.

Locuratolo, E. and Palomäki, J. (2015). Perspective for database preservation, *Encyclopedia of Information Science and Technology,* 3rd edn, pp.1855–1866.

Loney, K. (2009). *Oracle database 11g: The complete reference*. New York, US: Oracle Press, McGraw-Hill Companies.

lv, J. and Ren, H. (2016). 'Heterogeneous database synchronization mechanism based on ETL and XML', *RISTI (Revista Iberica De Sistemas E Tecnologias De Informacao)*, 17A, p. 153.

Mahmood, N., Burney, A., Ahsan, K. (2010). *A Logical Temporal Relational Data Model*, International Journal of Computer Science, 7(1).

McFadyen, R. (2015). *Relational databases and Microsoft Access*. Available at: http://www.acs.uwinnipeg.ca/rmcfadyen/CreativeCommons/Relational%20Databases%20and%20 Microsoft%20Access%20V2.0.pdf. (Accessed: 23 August 2017).

Microsoft Office. (2010). *Data programming with Microsoft Access 2010*. Available at: https://msdn.microsoft.com/en-us/library/office/ff965871(v=office.14).aspx. (Accessed: 27 September 2017).

Miller, V. (2014). *Understanding digital culture*. Sage Publications.

Mistry, R. and Misner, S. (2012). *Introducing Microsoft SQL Server, Microsoft Corporation*. Available at: https://www.google.jo/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwjK2MPotO3VAhUFuhoKHWyxCzgQFghBMAM&url=http%3A%2F%2Fdownload.microsoft.com%2Fdownload%2Ff%2Ff%2F6%2Fff62cae0-ce38-4228-9025-fbf729312698%2Fmicrosoft_press_ebook_introducing_microsoft_sql_server_2012_pdf.pdf&usg=AFQjCNECw3mIohTUH7VUSpKfYoMEI5bROg. (Accessed: 23 August 2017).

Moilanen, K., Niemi, T., Näppilä, T. and Kuru, M. (2015). 'A visual XML dataspace approach for satisfying ad hoc information needs', *Journal of the Association for Information Science and Technology*, 66(11), pp. 2304–2320.

MSDN Library. (2017). *MS Access architecture*. (Accessed: 23 August 2017).

MySQL. (2017). *MySQL 5.7 reference manual including MySQL NDB Cluster 7.5 and NDB Cluster 7.6*. Available at: https://downloads.mysql.com/docs/refman-5.7-en.pdf. (Accessed: 24 August 2017).

National Library of Scotland. (2014). *Long term preservation protects the availability and accessibility*. Scotland: Scotland Government.

NCDC. (2017). *Preserving historical data in the NCDC archive: National Centers for Environmental Information (NCEI) formerly known as National Climatic Data Centre (NCDC)*. Available at: https://www.ncdc.noaa.gov/news/preserving-historical-data-ncdc-archive. (Accessed: 4 July 2017).

Oracle. (2013). *Oracle migration workbench reference guide for Microsoft Access 2.0, 95, 97, 2000 migrations release 9.2.0 for Microsoft Windows 98/2000 and Microsoft Windows NT*. Available at: https://docs.oracle.com/cd/B10501_01/win.920/a97262/ch2.htm. (Accessed: 24 September 2017).

Oracle. (2017). *Oracle database online documentation, 10g Release 2 (10.2)*. Available at: https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm#i60803. (Accessed: 24 August 2017).

Park, J. and Brenza, A. (2015). 'Evaluation of semi-automatic metadata generation tools: A survey of the current state of the art', *Information Technology & Libraries*, 34(3)

Perrin, J., Winkler, H. and Yang, L. (2015). ‘Digital preservation challenges with an ETD collection—A case study at Texas Tech University’, *The Journal of Academic Librarianship*, 41(1), pp. 98–104.

Plew, R. and Stephens, R. (2002). *Introduction to the SQL database query*. 6th edn. Indiana, USA: Pearson Education Publisher.

Roland, L. and Bawden, D. (2012). ‘The future of history: Investigating the preservation of information in the digital age’, *Library & Information History*, 28(3), pp. 220–236.

Roman, S. (1999). *Access database design & Programming*. 2nd edn. USA: O'Reilly Publishing.

Sabău, G. (2007). ‘Comparison of RDBMS, OODBMS and ORDBMS’, *Informatica Economica*, 4(44), pp. 83–85.

Saikia, A., Joy, S., Dolma, D. and Mary, R. (2015). ‘Comparative performance analysis of MySQL and SQL server relational database management systems in Windows environment’, *IJARCCE*, 4(3), pp. 160–164.

Schaefer, S., Smorul, M., Minor, D. and Ritter, M. (2016). A decade of preservation: System migrations in Chronopolis*, Proceedings of the 13th International Conference on Digital Preservation*, Bern. 3-6 October 2016. Swiss National Library, pp. 15–17.

Seoane, C.G. (2014). *Digital preservation in the age of cloud and big data*. (white paper) Spain: Atos Scientific Community.

Shirish, K. C. (2010). *Dynamical modeling of MySQL database server*. Master's thesis. Lund University.

Singh, P. and Pottle, B. (2009). *Oracle database 11g: SQL fundamentals I (electronic presentation).* Available at: https://www.computer-pdf.com/database/121-oracle-database-11g-sql-fundamentals-course.html. (Accessed: 24 August 2017).

Stajano, F. (1998). *An introduction to relational databases*. 6th edn. Cambridge, UK: Addison-Wesley.

Stancic, H., Rajh, A. and Brzica, H. (2015). 'Archival cloud services: Portability, continuity, and sustainability aspects of long-term preservation of electronically signed records', *Canadian Journal of Information and Library Science*, 39(2), pp. 210–227.

Stefanova, S. (2013). *Scalable preservation, reconstruction, and querying of databases in terms of semantic web representations*. Sweden. Uppsala: Acta Universitatis Upsaliensis.

Swiss Federal Archives. (2017). *SIARD (Software Independent Archiving of Relational Databases) Version 1.0*. Available at: http://www.digitalpreservation.gov/formats/fdd/fdd000426.shtml. (Accessed: 4 July 2017).

Thomson, S. (2016). *Preserving transactional data.* (report) UK: DPC Charles Beagrie Ltd, The Economic and Social Research Council.

Trissl, S. (2012). *Cost-based optimization of graph queries in relational database management systems*. PhD. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II.

Tutorials Point. (2016a). *MS SQL Server*. Available at: https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_tutorial.pdf. (Accessed: 24 August 2017).

Tutorials Point. (2016b). *MYSQL database management system*. Available at: https://www.tutorialspoint.com/mysql/mysql_tutorial.pdf. (Accessed: 24 August 2017).

Tutorials Point. (2017). *MS Access tutorial*. Available at: https://www.tutorialspoint.com/ms_access/ms_access_indexing.htm. (Accessed: 23 August 2017).

Upadhyaya, P., Anderson, N., Balazinska, M., Howe, B., Kaushik, R., Ramamurthy, R. and Suciu,

D. (2013). Stop that Query! The need for managing data use, *Proceedings of the CIDR 2013, 6th Biennial Conference on Innovative Data Systems Research (CIDR '13)*, Asilomar, California, USA. 6–9 January 2013. pp. 1–4.

Van Tassel, J. (2013). *Digital TV over broadband*. Hoboken: Taylor and Francis.

Woods, K. (2010). *Preserving long term access to United States government documents in legacy digital formats*. PhD. Indiana University.

Wu, W., Chi, Y., Zhu, S., Tatemura, J., Hacigümüs, H. and Naughton, J.F. (2013). Predicting query execution time: Are optimizer cost models really unusable?, *Proceedings of the Data Engineering (ICDE), 2013 IEEE 29th International Conference on IEEE*. pp. 1081–1092.

XML. (2002). *Normalizing XML*. Available at: https://www.xml.com/pub/a/2002/11/13/normalizing.html. (Accessed: 8 February 2018).

APPENDICES

Appendix: Conversion Algorithm

The following SQL was revised to find which schools had teachers of English courses.

```
SELECT DISTINCT dno
FROM school, course, te.course, teacher
WHERE te.coursetitle = ` English'
AND course.courseno = tecourse.courseno
AND tecourse.teno = teacher.teno
AND teacher.schoolno = school.schoolno;
```

The following is the algebraic formulation.

$\text{PROJECT}_{\text{tename}}$ (school $\text{JOIN}_{\text{schoolno = schoolno}}$ (
$\text{PROJECT}_{\text{teno}}$ (teacher $\text{JOIN}_{\text{teno = teno}}$ (
$\text{PROJECT}_{\text{teno}}$ (tecourse $\text{JOIN}_{\text{courseno = courseno}}$ (
$\text{PROJECT}_{\text{courseno}}$ ($\text{SELECT}_{\text{ctitle = `English'}}$ course)
))
))
))

Symbolic representation

Based on the above example, complicated cases using name operators for a large number of statements, such as PROJECT and JOIN, should use common symbolic notations instead.

- SELECT ->σ
- PROJECT -> π
- PRODUCT -> ×
- JOIN -> |×|
- UNION -> ∪
- INTERSECTION -> ∩
- DIFFERENCE -> -
- RENAME ->ρ

Symbolic Usage

The symbolic operators correspond to the verbal ones.

$\text{SELECT}_{y = 1}(F)$
In symbolic will be : $\sigma_{y = 1}(F)$

However, conditions can be grouped using ^ (AND) and v (OR).

$\text{SELECT}_{\text{F = 1 ^ surname = `Hasan'}}$(Taecher)
In symbolic:
$\sigma_{\text{F = 1 ^ surname = `Hasan'}}$(Teacher)

Thus, using symbols, an abbreviation can be represented as follows.

PROJECT_{F} (School $\text{JOIN}_{F = F}$ (
PROJECT_{no} (Teacher $\text{JOIN}_{no = no}$ (
PROJECT_{Sf} (Subject $\text{JOIN}_{Sf = Sf}$ (
PROJECT_{Cs} ($\text{SELECT}_{Cs = \text{`English'}}$ Subject)))))))

The corresponding symbolic notation is given below.

π_{F} (School |×| (
π_{no} (Taecher |×| (
π_{Sf} (Subject |×| (
π_{Cs} ($\sigma_{Cs = \text{`English'}}$ Subject)))))))

Rename Operator

The rename operator changes the name of an existing relation; for example, ρ_{S}(B) is the relation B with its name changed to S.