A Comparison of Concepts between Native Xml and Relational Database Systems

Mohammed Qader Kheder[1], ChnoorMeheadeen Rahman[2], SheneJalil Jamal[3]

[1, 3] Department of Computer, Faculty of Science and Educational Sciences, University of Sulaimani, [2]Department of Science, Faculty of Basic Education, University of Charmo, Sulaimani, IRAQ.

[1] mohammed.kheder@univsul.edu.iq, [2] chnoor.rahman@gmail.com,
[3] *shene.jamal@univsul.edu.iq*

ABSTRACT

The way information is exchanged and combined is dramatically affected by the growth of XML and the Internet. XML changes the way databases are used, even the database vendors did not predict these changes a few years ago. Extensions and mechanisms offered by most of the database systems to support the management of XML data. However, the emergence and importance of native XML database could not be replaced by the extensions to the other database systems. Furthermore, another group of researchers state that XML plays a very important role in the database world, especially in marking up data on the web. This study is going to examine the differences between XML and Relational database systems in terms of representing data, the way both systems are queried and the performance of operating queries on both systems. Moreover, the factors and existing differences between XML and relational database are discussed to help the developers choose the better-suited database for their works.

Keywords: Native XML, RDBMS, Extreme Programming, XQuery, Saxon, XQJ.

INTRODUCTION

Database is one of the most significant branches of computer technology since it can be used to store and manage information for a period of time. Database systemsare used in various computer systems and applications so that the tasks can be accomplished easily and quickly (Osman and Knottenbelt, 2012). Nowadays, with the rise in use of the World Wide Web (WWW) the popularity of XML continues to increase. This in turn has created an urgently need to create Native XML Database (NXD) for storing and managing XML documents efficiently.Conversely, Relational Database Management System (RDBMS) is still a powerful database system that has been used for decades.

Claims have been made against the differences exist in efficiency and ability between NXD and RDBMS. In general, a good database performance means minimizing the response time for each query, and maximizing the throughput of the entire database of an application.Thebest performance could be achived with right choice of a database system.

This paper will focus on the differences between XML and RDBMS. It also tries to find out which database model can provide higher performance. Also, In term of flexibility, does NXD provide a good result or it will be better to choose other technologies, or does working with both technologies at the same time offer a better result. To help answering these questions this paper is come to exist.

Copyright © 2015

Leena and Luna International, Oyama, Japan.
(株)リナアンドルナインターナショナル, 小山市、日本.
ISSN: 2186-8476, ISSN: 2186-8468 Print
www.ajsc. leena-luna.co.jp

METHOD

Methodology imposes a disciplined process upon software development with the aims of creation software development more predictable and efficient(Avison and Fitzgerald, 2006). According toVerma*et al*.(2014), in order to build an informed choice of which method to use, it is necessary to look at the deliverable required from the project together with time scale by which this deliverable have to be accomplished. There are many types of agile methods which can be used for a group project, such as Extreme Programming (XP), Dynamic System Development Method (DSDM), Feature Driven Development (FDD), etc.

Extreme Programming method is a discipline of software development based on values of communication, simplicity, bravery and feedback. The XP works by bringing the whole team together in the presences of effortless practices (Sauter, 2006). Furthermore, it works with enough feedback to facilitate the team to see where are they and where should they go. The following advantages are the reasons of using Extreme Programming method in this study:

- XP works as a group with simple design and fanatically tested code. These properties are important to help the researchers to keep the system integrated and running all the time.
- XP method allows programmers to write all production code in pairs(Beck and Andres, 2004)this helps the team members of this experimentto work together and produce the result.
- XP shares a common and simple picture of what the product looks like, which mean it will be able to know which step is the system in it now.

RELATED WORK

Performance of Relational Databases versus Native XML Databases: This study by Otago University supports that with the storage and management of document-centric (documents that are designed for human consumption such as email, advertisements and books) data there will be significant differences between these two models (University of Otago, 2005).

Within this research a collection of email messages were used in order to restrict the data to be stored and manipulated. PostgrSQL and eXist were used to provide representative evidence on the performance of RDBMSs and NXDs respectively. In this research eleven different tasks are used to test the performance of these databases.

The obtained results showes that eXist will provide higher performance when the query conditions are kept to only one or two conditions. However, PostgrSQL will provide higher performance with basic range access. This work concludes that the performance is based on the type of queries and the structure of the data. The performance of the different databases varies according to the type of data required and the number of the conditions in the queries will have affects on the performance.

CHOSEN DATABASE

Database is simply series of tables that include data or information. These tables can be small, or contain thousands of records. The main purpose of a database is to make it quick to manage and control the data(Connolly and Begg, 2014). According to Connolly and Begg (2014), a database enables users to efficiently store, receive and sort data. There are lots of Database Management Systems (DBMS) which can be used to store and receive data. One of

Copyright © 2015

the most popular relational database systems is MySQL. MySQL is used in the most commonly websites and systems. It is an extremely fast and robust RDBMS. It allowsmultipuleaccess to the data, and can ensure that only authorized users can get access. Consequently, MySQL is a multithread and multiuser server (Schwartz *et al.*, 2012). To query data in MSQL database, SQL (Structured Query Language) is used to write the queries. This paper focuses on MySQL as a relational database and compares it to XML as a tree-structured database model. Inthis study MySQL is used as a relational database to be compared to XML as a tree-structured database model.

WHY MYSQL

MySQL provides many useful features to systems and applicationsThe features that are discussed below are the main reasons behind choosing MySQL over the other RDBMS in this study.

- MySQL is platform independent; it can run on many various operating systems (Denton and Peace, 2003).
- MySQL can support many types of data that can be stored in DBMS, namely, Numeric data, Boolean, Date and Time and String data.
- MySQL is Open Source. It can be obtained as a free, and can be used by anyone without any license under the GNU (General Public License) (Connolly and Begg, 2014). This enables developers to modify the source code according to their requirements and customize the database for their usage.
- MySQL is scalable: According to Schwartz *et al.*(2012), MySQL can handle approximately any amount of data or information, up to as much as fifteen millions of records or rows and more.
- MySQL has other good features including easy installing and using, robust, security, fast to perform tasks(Denton and Peace, 2003).

XML VERSUS RELATIONAL

When a database is designed, it should be asked if the data is better suited to the relational or XML model. The issue of choosing the best database model in terms of flexibility, performance and efficiency is always controversial. It could be said that the answer to “what is better XML or RDBMS?” is always “it depends”. In the following sections the differences between XML and relational data and the factors that may influence the choice will be discussed.

Xml And Relatıonal Differences

Despite of having some common aspects, such as storing data and techniques for extracting data,there are many differences between XML and relational database (Font, 2005). The major differences between these two are:

- In relational database the data is stored in a collection of tables (rows and columns).In the tables the data is related using the concept of “foreign keys”. It is well suited to structured data (University of Otago, 2005). However, in native XML database, XML document is used as a fundamental unit of storage. XML document contains elements, attributes and PCDATA (Vohra, 2006)(Font, 2005). Native XML database is well suited to nested, complex and semi-structured documents (Vohra, 2006).

Copyright © 2015

Leena and Luna International, Oyama, Japan.
(株)リナアンドルナインターナショナル, 小山市、日本.
ISSN: 2186-8476, ISSN: 2186-8468 Print
www.ajsc. leena-luna.co.jp

- Relational data is represented in a model of logical relationships while XML data is hierarchical. With the relational model, creating a relationship between parent table and dependent table forms the relationship of data. However, information about the relationships of data items in an XML database is saved in a form of hierarchical structure of data (IBM Software Group, 2005).
- XML data is self-describing, but the relational data is not. XML document also contains tagging to explain what the data is. Different types of data can be saved in a single XML document. While, in relational database model the column definition is used to define the content of the data. All saved data in a column must have the same type of data.
- Ordering is inherited in XML document, whereas it is not in relational data. For an XML document, it is assumed that the order of the specified items is the same as the order of the data in the document. Often, there is no other ways to specify order within the document. For relational data, unless you specify an ORDER BY clause on the column(s) the order of the rows is not guaranteed.

The Factors, Which İnfluence The Choice Between Xml And Relational

The differences between XML and relational data are described. However, the differences are not the only key to choose the way to store the data. There are a number of other factors, which might influence the decision about which model to usesuch as:

- Flexibility:

Relational tables are fairly fixed. This makes it difficult to de-normalize many tables into one, or normalize one table into many. If the data design changes frequently, it will be a better choice to represent it as XML data (Amiano*et al*., 2006).

- Performance:

Interpreting and serializing XML data is expensive. Because of that, relational data will be a better choice if performance is more important than flexibility (IBM Software Group, 2005).

- Whether data attributes apply to only a small subset of the data, or to all data:

If a large number of attributes exist, and only some of which are applied to any specific data values, the XML model is a better option. For this reason, if we have chosen a relational data, we must assign a space for all the attributes whereas it might be only a small number of those attributes are relevant for any given items (IBM Software Group, 2005).

- Referential integrity:

If referential integrity is required, the data should be stored as relational data. While XML columns cannot be defined as part of referential constraints (Amiano*et al*., 2006).

- Whether the data needs to be updated frequently:

Currently, XML can be updated only by replacing full documents. If small fragments of very large documents needed to be updated very often, it can be more efficient to store the data in non-XML columns. However, storing as XML can be efficient as well, if and only if you are updating small documents and only a few documents at a time (Amiano*et al*., 2006).

Copyright © 2015

According to the differences between XML and relational data and also the factors, it can be said some data is better suited to relational and some other is better suited to XML model.

QUERY LANGUAGES

Storing and organizing large amount of data in a database system needs a technique to manipulate and process the data. A query language is a software program that works based on a predefined structure. There are different query languages for querying and processing data that is stored in different database systems. For instance, Structures Query Language (SQL) is used to implement queries in RDBMS, While XPath, XSLT and XQuery are query languages to manage and extract data in XML formatted database system.

Relational Database and SQL: In a Relational Database, the data that is stored in a set of tables (which can be viewed as a grid of rows and columns) is often queried with SQL. As most of the world's important data had been stored in Relational Databases and most applications and users use SQL to find, extract and process that data, So, SQL has been the most popular way of searching across tables of data for the past few decades(Melton and Buxton, 2006). Generally, SQL perform four operations, which are **Selecting** a specific data to be retrieved, **Inserting** data into the database tables, **Updating** the existing data, and **Deleting** data.

The structure of SQL for performing data extraction is based on Select Fields from Tables Where some Conditions (which are used to filter the selection) are True. According to Melton and Buxton(2006), the most important types of **Select** operations are Projection, Selection, Union and Join.

A **Projection** returns only the data in columns, by mentioning the column name in the select clause. The structure is: *SELECT columnName(s) FROM tableName;*

With a **SELECT** command data in some rows of a table can be returned by filtering the result with some conditions. The structure is:

*SELECT * FROM tableName WHERE condition(s);*

A query can be constructed with Projection and Selection for retrieving a specific data from the database. For example *SELECT columnName(s) FROM tableName WHERE condition(s).*

To perform a specific query effectively, sometimes many tables need to be joined together. With **Union** and **Join** data from multiple tables can be joined together (James, 2005).

The **INSERT** command is used to perform adding a new row into a table. The structure of this command is:

INSERT INTO tableName VALUES (value1, value2 ...);

The *VALUES* clause provides the values for the columns of the table in the order in which they are expressed.

UPDATE is another SQL command that is used to update an existing record in a database table. The record can be specified by applying some conditions in WHERE clause. The structure of this command is:

UPDATE tableName
SET column1=value1, column2 = value2,... WHERE condition;

DELETE command is used to remove a specific row in the database table. This command can also specify the record to be deleted by setting a condition in WHERE clause. The structure of this command is:

Copyright © 2015

Leena and Luna International, Oyama, Japan.
(株)リナアンドルナインターナショナル, 小山市、日本.
ISSN: 2186-8476, ISSN: 2186-8468 Print
www.ajsc. leena-luna.co.jp

DELETE FROM tableName WHERE condition;

Figure 1 shows how SQL operates query on relational database model.

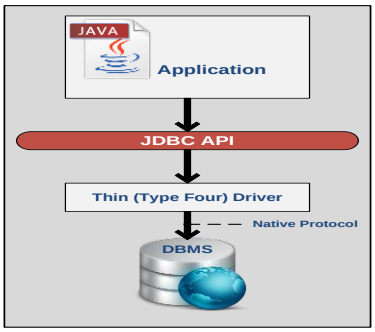

Figure 1: SQL on Relational Model

XML Query Languages: There are differences between the way in which data is expressed in the traditional Relational Database and XML formatted database. Therefore, the query languages and techniques that are developed to process XML data are different from SQL. Querying XML data needs navigating through the tree structure of the XML file which may or may not be well defined in terms of structure and type (Melton and Buxton, 2006). In this paper three XML query languages will be presented which are **XPath**, **XSLT** and **XQuery**.

XPath: is a common path-based XML query language which has the ability to navigate, select and extract nodes in an XML document. This language is used in constructing XSLT and XQuery. XPath specifies the path of node to be selected. This path expression can be absolute or relative. With absolute path a node is addressed starting from the root element. For instance*: /nodeName/*childElement[1] /@attributeValue* consists of three elements, its starts from root and selects the mentioned node names; the second part selects the first child among the returned node names, and the third part gets the value of the named attribute value. In contrast, relative path can start from any element. For example *elementName/attribiteValue* returns all the attribute values that belong to the element name. XPath recognizes element, attribute, text, namespace, processing-instruction, comment and document nodes which are seven types of nodes (Ciravegna, 2012).

XSLT: information expressed in XML can be translated from one terminology to another. XSLT makes the XML data to be transformed to another structure in order to make it work with a specific application. It inherits navigation ability from XPath.

XQuery: it is a query language that can find and retrieve information from XML documents. A query in XQuery can be seen as a three part expression. In the first part any node in the XML document can be selected with XPath specifications, in the second part search criteria is added to the selected nodes and the third part is an Application Programming Interface to execute the query in a programming language environment (McLaughlin, 2008). In this study queries are performed on an XML database using XQuery expressions, as XQuery can play the same role as MySQL does in Relational Database. It can extract data from NXD database based on a single or multiple searching criteria. XQuery defines a control structure called **FLWOR** which support variable binding and iteration trough the XML elements. This expression stands for **F**or, **L**et, **W**here, **O**rder by and **R**eturn (Kay, 2011). **FLWOR** also has the ability to joins multiple documents and filter the results of the queries (ibm.com, 2012). For example:

Copyright © 2015

```
for $oddValues in (1 to 20)
let $var := 2
where $evenValues % $var = 1
order by $oddValues decending
return $oddValues
```

XQuery contains the build-in functions that are defined in XPath, such as distinct-values() in (*for $child in distinct-values(doc($docName)/rootName/childName/*/name()*) return $child). It also contains the functions that are known as aggregation functions in SQL. Users are also allowed to define function to execute a query (Cheng, 2002). For instance:

```
declare function local:findSmallest($p1 as xs:decimal?, $p2 as xs:decimal?)
AS xs:decimal? {
let $largest := max(($p1, $p2))
return ($largest) };
```

There are a number of XQuery processors that allows XQuery to be used in programming language environments, namely, Saxon, BaseX, Zorba and Sedna are XQuery processors that work in Java. The below figure illustrates performing XQuery expressions on XML database using Saxon processor.

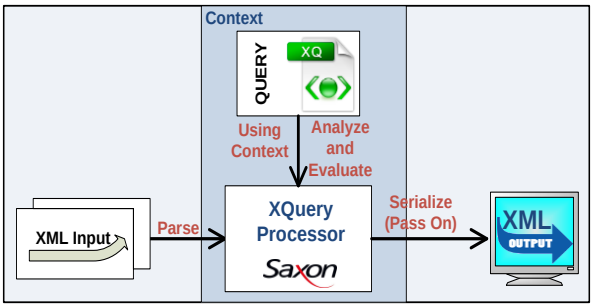

Figure 2: XQuery on XML

PERFORMANCE COMPARISON

In terms of performance, it cannot be said that which model (XML or RDBMS) is better for all systems. It mainly depends on the structure of data stored the system. In this section the performance of both models are discussed.

Despite the fact that XML database processing is relatively fast, according to IBM Software Group (2005), when the highest performance is required, the better choice for most of the time is relational database, even if the data is in XML form. The reason is that interpreting and parsing XML database will cause some expenses. So that, considering storing data as relational data instead of XML is preferred for applications which cannot give a risk even of a small incremental expense with processing XML data.

On the other hand, in some cases, XML shows higher performance, precisely because of its flexibility. In relational model, to fit business data into flat, tabular structures, normalization is required. This normalization requires transformation when data is stored and retrieved, and, also leads to multi join queries, which decreases the performance (IBM Corporation, 2011). Moreover, in this study the performance of XML and relational data will be examined to make the differences in performance between the models more clear, and to help the users to choose the better-suited model with higher performance.

Copyright © 2015

Leena and Luna International, Oyama, Japan.
(株)リナアンドルナインターナショナル, 小山市、日本.
ISSN: 2186-8476, ISSN: 2186-8468 Print
www.ajsc. leena-luna.co.jp

CONSTRUCTING QUERIES

Xml Queries In Xquery

There are number of programming languages that can support XML related technologies, such as Java, Perl, Python, C++, Delphi and some others. A large number of Application Programming Interfaces (API) have been developed in Java to organize and manage XML data(Vohra, 2006). Therefore, XQuery API provided by Java (XQJ) is used in this paper to perform some XQuery queries on a Native XML database in a Java environment

To implement XQuery expressions using XQJ, First, the program must be able to connect to the data source, and then, the queries will be constructed. Finally, the result returned by the queries can be processed.

that are needed to process an XQuery expression are:

1. The **XQDataSource:** The starting point of an XQJ application must be declaring an object from **XQDataSource** class.

```
XQDataSource datasource = new XQDataSource();
```

This object creates a session with a specific application, processes the XQuery expressions and finally returns the result.

2. **Establishing a connection:XQConnection** interface with an object from **XQDataSource** class can represent a connection with a particular XQuery implementation.

```
XQConnection connection = datasource .getConnection();
```

3. **Building the query expressions:** To evaluate XQuery expressions, XQJ uses an object from **XQExpression** class which can be defined in **XQConnection** context. This object executes the queries by calling **executeQuery()** method. The object can also be reused whenever another XQuery expression needed to be executed. The result of the query will be a sequence of XML format and represented in **XQResultSequence** object (Katz *et al*., 2003).

The following expression is an example of constructing an XQuery expression.

```
XQResultSquence xqs;
XQExpression xqe= connection.createExpression();
xqs=xqe.executeQuery("doc('cd_books.xml')//book[isbn = '17487435']");
xqs=xqe.executeQuery("doc('cd_books.xml')//book[isbn = '19853123']");
```

In XQJ a query can be prepared once and executed several times. This preparation requires uses external variables and bounding different values to them, and several other processes happen such as parsing and validating the query and optimizing the execution plan (Hua Liu and Melton, 2009).

Mysql Queries Via Jdbc

To access a relational database (MySQL in this paper) from a Java environment, Java Database Connectivity (JDBC) is used, which is an interface allows creating a connection with a database, execute MySQL queries and return the result.

The basic expressions that are needed to process a MySQL query are:

1. **Loading the MySQL driver:** Each database has its own driver. The following

Copyright © 2015

line represents the operation of loading MySQL driver in Java.

```
Class.forName("com.mysql.jdbc.Driver");
```

2. **Establishing connection:** The following expression creates a connection with a specific MySQL database.

```
Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost/dbName?","user","password"
);
```

Issue queries: To issue MySQL queries an object from PreparedStatement is used.

```
PreparedStatement statement=conn. prepareStatement("select * from books where
genre = \"computer\" ");
```

3. **Executing the quires and returning the result:** The query is executed and the results can be returned by using an object from ResultSet as shown in the following expression.

```
ResultSet result = statement.executeQuery();
 while(result.next()){
   System.out.println(result.getString(1)+"  "+
                      result.getString(2)+"  "+ result.getString(3)+"  "+
                      result.getString(4)+"  "+ result.getString(5)); }
```

RESULT AND DISCUSSION

To retrieve data from both database systems, several queries (MySQL and XQuery) expression have been executed. The following are some of the query expressions and their results.

- The XQuery expressions are executed against an XML database (which is shown in figure (3)).

```
<CD>
    <TITLE>Red</TITLE>
    <ARTIST>The Communards</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>London</COMPANY>
    <PRICE>7.80</PRICE>
    <YEAR>1987</YEAR>
</CD>
<CD>
    <TITLE>Unchain my heart</TITLE>
    <ARTIST>Joe Cocker</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>EMI</COMPANY>
    <PRICE>8.20</PRICE>
    <YEAR>1987</YEAR>
</CD>
<book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications
</book>
<book id="bk102">
<author>Ralls, Kim</author>
<title>Midnight Rain</title>
<genre>Fantasy</genre>
```

Figure 3: A sample of the XML database

The following XQuery expression uses the concept of prepared queries.

Copyright © 2015

Leena and Luna International, Oyama, Japan.
(株)リナアンドルナインターナショナル, 小山市、日本.
ISSN: 2186-8476, ISSN: 2186-8468 Print
www.ajsc. leena-luna.co.jp

```
final String sep = System.getProperty("line.separator");
String queryExpression =
  "declare variable $docName as xs:string external;" +sep+
  "for $catalog in doc($docName)/*/book"+
  " where $catalog/genre = \"computer \""+
   " order by $catalog " +
  return $ catalog ";
  XQDataSource datasource = new XQDataSource();
  XQConnection connection=datasource .getConnection();
  XQPreparedExpression expr = connection.prepareExpression(queryExpression);
  expr.bindObject(new QName("docName"), XMLfilename, null);
  XQResultSequence rs = expr.executeQuery();
```

This query is processed using Saxon XQuery processor in Java. Saxon is an XQuery and XSLT processor, which is developed by Michael Kay. The processor is embedded in the Java application using XQJ. The result of the query is shown in figure 5.

```
<book id="bk101">
<author>Gambardella, Matthew</author>
<title>XML Developer's Guide</title>
<genre>Computer</genre>
<price>44.95</price>
<publish_date>2000-10-01</publish_date>
<description>An in-depth look at creating
Applications with XML.</description>
</book>
```

Figure4: The XQuery Result

The java program which processed evaluating the above query took **1180 millisecond** to run.

XQuery expressions can be filtered by adding more conditions in the *Where* clause using and operator *(Katz et al., 2003)*.

A single condition MySQL query expression is examined against a relational database which contains records about books.

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn=DriverManager.getConnection
("jdbc:mysql://localhost/dbName?","user","password");
PreparedStatement statement = conn. prepareStatement("select * from books where genre
= \"computer\" ");
ResultSet result = statement.executeQuery();
      while(result.next()){
     System.out.println(result.getString(1)+"  "+
                        result.getString(2)+"  "+
     result.getString(3)+"  "+
     result.getString(4)+"  "+
     result.getString(5)); }
```

The query returns the following result:

```
bk101 Gambardella, Matthew  XML Developer's
Guide  Computer  45
```

The java program, which processed evaluating the above query, took **800 milliseconds**. In MySQL multiple conditions can also be applied to a query by using *and* operator in the *Where* clause (James, 2005).

Copyright © 2015

The MySQL and XQuery expressions are constructed and implemented in a computer system with the following specifications:

OS Name: Microsoft Windows 7 Home Premium
Version: 6.1.7601 Service Pack 1 Build 7601
System Model: HP Pavilion dv6 Notebook PC
System Type: x64-based PC
Processor: AMD Phenom(tm) II P960 Quad-Core Processor, 1800 Mhz, 4 Core(s), 4 Logical Processor(s)
Installed Physical Memory (RAM): 6.00 GB
Total Physical Memory: 5.74 GB
Available Physical Memory: 2.27 GB
Total Virtual Memory: 11.5 GB
Available Virtual Memory: 8.02 GB

Some of the multiple condition queries that have been tested in this research are shown in the table 1.

Table 1: XQuery and MySQL expressions

Query	*Expression*	*Execution Duration in Millisecond*
XQuery with two search conditions	"declare variable $docName as xs:string external;"+ ***sep*** + "for $cat in doc ($docName) /*/book"+" where $cat/ genre = \"fantasy\" and $cat/author = \"Corets, Eva\" "+ " order by $cat "+ "return $cat";	1180 ms
MySQL with two search conditions	prepareStatement(""select * from books where genre = \"Fantasy\"" +"and author = \"Corets, Eva\" ");	780 ms
XQuery with three search conditions	"declare variable $docName as xs:string external;"+ ***sep*** + "for $cat in doc ($docName) /*/cd"+ " where $cat/PRICE < 9 and $cat/COUNTRY = \"UK\" "+ "and $cat /YEAR < 2000"+ " order by $cat "+ "return $cat";	1190 ms
MySQL with three search conditions	prepareStatement("select author, publish_date from books "+ "where genre = \"Fantasy\" "+ "and price > 5 "+ "and title = \"Midnight Rain\");	782 ms
XQuery based on a unique key	"declare variable $docName as xs:string external;"+ ***sep*** + "for $cat in doc($docName) /*/book"+ " where $cat/id = \"bk101\""+" order by $cat "+ "return $cat";	882 ms
MySQL based on a unique key	prepareStatement("select * from books"+ "where id = \"bk110\" ");	887 ms

As it has been shown in the above table, the performance of MySQL is almost in all the cases is higher than XQuery. However, this does not mean that XQuery will cease to exist due to the poor performance it shows, as the performance totally depends on the structure of the data saved in the database.

CONCLUSION

As illustrated in the previous sections, both Relational Database and XML developed to store and manipulate data. Furthermore, a large number of technologies have invented to organize

Copyright © 2015

Leena and Luna International, Oyama, Japan.
(株)リナアンドルナインターナショナル, 小山市、日本.
ISSN: 2186-8476, ISSN: 2186-8468 Print
www.ajsc. leena-luna.co.jp

and query data in both systems. However, there are some differences between XML and Relational databases in the way that the data is represented, organized and queried.

XML is a common way to marking up data, which means the data is stored in text files that are both machine and human readable (the text files contain meaningful characters that tell the meaning of the data to reader). The syntax of the data has a hierarchical structure and the relationships between the elements are clearly visible. A part from comparing the syntax, performance and data extraction from both XML and relational database, this study examined that data in XML files can be delivered from one party to another with all the information included. This means to extract data from an XML document there is no need to identify a schema or any kind of Meta Data. In contrast, data in relational database model needs a static definition of schema. This is the reason for calling XML "self-describing". Consequentially, XML's ability to transmit self-describing data has made it to become a dominate way to marking up data in Web Services.

This study examined that although processing data in XML database is relatively fast, parsing and interpreting the XML also needs time. Hence, if an application cannot tolerate the cost (caused by time) which may associate with processing XML, then the data should be stored in Relational instead of XML. Furthermore, consider using Relational to handle and maintain large amount of structured data. Using SQL transactions, a large volume of data can be effectively and reliably managed and updated.

REFERENCES

[1] Amiano, M., D'Cruz, C.,Ethier, K., &Thomas, M. (2006).*XML: Problem – Design – Solution*.Canada: Wiley Publishing, Inc.

[2] Avison, D., & Fitzgerald, G. (2006). *Information System Development: Methodologies, Techniques and Tools*. 4th ed. Maidenhead: McGraw – Hill.

[3] Beck, K., & Andres, C. (2004).*Extreme Programming Explained: Embrace Change*.2nd ed.United States: Addison – Wesley Boston.

[4] Cheng, A. (2002).*XQuery: A Flexible Query Language for XML*. [online] Available at: http://www.drdobbs.com/web-development/xquery-a-flexible-query-language-for-xml/184405242 [Accessed 5th Dec 2014].

[5] Ciravegna, F. (2012)XML, XML Schema and XPath,The University of Sheffield.

[6] Connolly, T., & Begg, C. (2014).*Database Systems: A Practical Approach to Design, Implementation and Management*. 6th ed. USA: Parson.

[7] De Lucia, A., Ferrucci, F.,Tortora, G., &Tucci, M. (2008). *Emerging Methods, Technologies and Process Management in Software Engineering*. Inc, Hoboken, New Jersey: John Wiley & Sons.

[8] Denton, J., & Peace, A.(2003). "Selection and use of MySQL in a database management course". *Journal of Information Systems Education*. *14*(4), pp. 401 – 408.

[9] Dubois, P. (2013).*MySQL: Developer's Library*. 5th ed. United States, New Jersey: Parson Education.

[10] Font, F. (2010).*An Introduction To Designing XML Data Documents*. [online]. Availableat:

Copyright © 2015

http://www.room4me.com/whitepapers/Designing_XML_Data_Documents_FrankFont1a.pdf. [Accessed 14thNov 2014].

[11] Font, F. (2005),*When to use XML instead of a Relational Database*. [online] Available at: http://www.room4me.com/index.php?option=com_content&view=article&id=8:xmlvsdb&catid=2:technology&Itemid=5. [Accessed 5thNov 2014].

[12] Hua Liu, Z., & Melton, J. (2009).*XQJ-XQuery Java API is Completed, Progress Data Direct and Oracle*.[online] Available at: http://www.sigmod.org/publications/sigmod-record/0912/p07.article.cappellen.pdf[Accessed 22nd Oct 2014].

[13] IBM Corporation. (2011)*Best Practices Managing XML Data*. United States: IBM.

[14] IBM Software Group. (2005)*Comparing XML and relational Storage: best practices*. United States: IBM Corporation.

[15] James, L. (2005).*Structured Query Language an Introduction*. Lulu:James McElhannon.

[16] Katz, H., Chamberlin, D., Draper, D., Fernandez, M., Kay, M.,Robie, J.,Rys, M., Simeon, J.,Tivy, J., &Wadler, P. (2003).*XQuery from the Experts: A Guide to the W3C XML Query Language*. Boston:Addison-Wesley Professional.

[17] Kay, M. (2011).*Saxon the XSLT and XQuery Processor*. [online] Available at :

http://saxon.sourceforge.net/ [Accessed 19th Sept 2014].

[18] Linden, A.,&Darbyshire, P. (2005)."A Benchmark Comparison between Native XML and Relational Databases. Managing Modern Organizations with Information Technology," *IRMA International Conference*. United State: Idea Group Inc.

[19] Manning, C., Raghavan, P., &Schutze, H. (2008).*Introduction to Information Retrieval*.Cambridge: Cambridge University press.

[20] McElhannon, J. (2005). Structured Query Language an Introduction. USA: James McElhannon.

[21] McLaughlin, B. (2008).*Use XQuery from a Java environment*.[online] Availableat: http://www.progress.com/products/data-integration-suite/data-integration-suite-developer-center/data-integration-suite-tutorials/learning-xquery/use-xquery-from-a-java-environment [Accessed 1th Sept 2014].

[22] Melton, J., & Buxton, S. (2006).*Querying XML XQuery, XPath and SQUXML in Context*.United States of America: Elsevier, Inc.

[23] New Zealand. University of Otago, Information Science. (2005).*Performance of Relational Databases versus Native XML Databases*.Dunedin: Otago University Research Archive.

[24] Osman, R., &Knottenbelt, W. (2012)."Database system performance evaluation models: A survey".*Performance Evaluation.69*(10), pp. 471 – 493.

[25] Sauter, V. (2006). *Extreme Programming*. [online] Availableat: http://www.umsl.edu/~sauterv/analysis/f06Papers/Hutagalung/ [Accessed 12th Nov 2014].

Copyright © 2015

Leena and Luna International, Oyama, Japan.
(株)リナアンドルナインターナショナル, 小山市、日本.
ISSN: 2186-8476, ISSN: 2186-8468 Print
www.ajsc. leena-luna.co.jp

[26] Schwartz, B., Zaitsev, P., &Tkachenko, V. (2012).*High Performance MySQL*. 3rd ed.USA: O' Reilly Media, Inc. [online] Available at: http://it-ebooks.info/book/676/ [Accessed 8th Oct 2014].

[27] Verma, J., Sunita B., &Himanshu P.(2014). "Develop Framework for Selecting Best Software Development Methodology".*International Journal of Scientific & Engineering Research*. *5*(4), pp. 1067 – 1070.

[28] Vohra, D. (2006).*Pro XML Development with Java™ Technology*. USA:Apress, Inc.

[29] Wicentowski, J. (2011).*An Introduction to XML Databases: Creating the eXist-db XML Database*. U.S. Department of State, Oxford, University of Oxford.

Copyright © 2015