Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/109303, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

XR-Gine: A Simulation Engine for Performance Evaluation on XML Enabled Databases

Su-Cheng Haw* and Qing-Feng Ang

Department of Computing and Informatics, Jalan Multimedia, Cyberjaya – 63100, Malaysia;
sucheng@mmu.edu.my, fengip4@gmail.com

Abstract

Background/Objectives: Since most of the enterprises are still using relational database as the back-end database, XML Enabled Database (XED) plays an important role to ensure the seamless integration of XML technology via the relational database. **Methods/Statistical analysis**: In this paper, we will compare the features supported in XED product, namely Microsoft SQL Server, Oracle and IBM DB2. In order to do so, we propose and develop XR-Gine to measure the performance of these storages. **Findings:** Some of the performance comparisons include speed of storing, retrieving, and updating functions in the database approaches. Based on the results obtained, the best performance of storing evaluation is Oracle. On the other hand, in terms of query retrieval, Microsoft SQL Server outperforms especially in supporting huge datasets, which is a necessity with the overwhelming data produced across the web nowadays. **Application/Improvements:** The results generated by the engine are beneficial to aid the community to choose the appropriate XED database system.

Keywords: Benchmarking, Data Storage, Native XML Database, Performance Evaluation, Query Evaluation

1. Introduction

XED is the database that has the additional mapping layer provided by third party or database vendor to map the XML tree into row and column storage basis. When the mapping takes place, it may cause the original XML data and structure lost. XPath, XSLT, DOM and SAX are the technologies that support data manipulation in XED.

There are many products that support XED[1] such as Microsoft Access 2007, IBM DB2, Informix, MySQL, Oracle, Microsoft SQL Server and FoxPro. Some of the products are commercial use (Microsoft Access 2014, IBM DB2, and Microsoft SQL Server) and some are open source (Informix, MySQL, FoxPro, and Oracle). Although, the three products selected to be evaluated are not open source, they fully support XML database on the trial or free version.

For Oracle database, it supports XQuery, Structured Query Language (SQL), eXtensible Stylesheet Language Transformations (XSLT), and XML indexes, Document Object Model (DOM), XML Schemas and XML Data Repository. The Oracle database constructs or deconstructs the XML data in XML Type columns. The XML Type column supports B-tree indexes, XML Index indexes, function-based indexes and Oracle Text indexes. In addition, the Oracle also supports the XQuery function in order to query XML values. Last, the XML Data Repository feature that included in Oracle allows viewing the XML values and other object in the database by using a file-system.

For the Microsoft SQL Server, the DBMS support XML by using the FOR XML clause, the XML data type, XQuery, the OPENXML clause and SQLXML. The SQL server stores the XML value as Binary Large Objects (BLOB) by using an optimized binary format for parsing. The XML schema is used in the SQL server to validate XML values on modification or insertion and also used to optimize storage and queries. The FOR XML clause has four modes to mapping the result set which are RAW mode, PATH mode, AUTO mode and EXPLICIT mode. The OPENXML clause constructing the relational data from XML document with three ways, from attributes,

*Author for correspondence

Table 1. Comparison of three selected XED

XML System	Oracle Database	Microsoft SQL Server	IBM DB2
Developer	Oracle	Microsoft	IBM
License	Commercial	Commercial	Commercial
Storage Type	Object-oriented, CLOB and BLOB	BLOBs	Native XML
Supported Features	XQuery, SQL/XML, XSLT, XML indexes, DOM, XML Schemas, XML DB Repository	FOR XML clause, XML data type, XQuery, OPENXML clause, SQL/XML	Pure XML, Net Search Extender, Web Services Object Runtime Framework named WORF

from child elements and from XPath queries that related to the row element.

For the IBM DB2, it supports XML in pure XML, Net Search Extender and Web Services Object Runtime Framework (WORF) for DB2 which named DB2 WORF. The pure XML consists of XML storage, XQuery, decomposition engine and SQL/XML. DB2 used the Native XML storage to store the XML data which is implemented with a hierarchical storage mechanism. It allows using stored procedure to update XML value by identified the path expression. The decomposition engine in DB2 is used to store the value from XML document into relational table. Net Search Extender of DB2 contains many search technologies which allows searching value easily, while WORF allows user to define Web services. Table 1 shows the comparison of the products.

2. Related Works

[2]evaluated the performance XED and Native XML Database (NXD) on inserting XML data into the respective storage. In their study, Tamino (NXD) and Oracle (XED) have been selected as the database. They used five different queries on five sizes of XML document for the experimental evaluation. Firstly, the five XML documents are generated from 100KB to 10MB. Secondly, the two XML database systems are prepared and the XML documents are imported into both databases. Thirdly, they prepared five queries with various conditions to test the benchmark of these two database systems. Lastly, the five queries are executed for five consecutive times, and the average time taken for processing were recorded. The results indicated that Tamino has the best performance to process heavy queries in large size of XML document. On the small sized dataset, the performance of Oracle and Tamino are comparable.

On the other hand,[3] compared the NoSQL and XML approaches (both NXD and XED) for clinical data storage. The reference model of this experiment is HL7-CDA which is the XML-based standard document for real clinical records[4]. HL7-CDA has five thousand records with about 120MB storage size. In their study, Microsoft SQL Server 2008 was chosen to represent the NoSQL and XED, while eXist represents the NXD. The results revealed that the NoSQL database has the best performance, which is 2 to 8 times faster than the XED and NXD, especially when the database has more records. Nevertheless, the NoSQL database size grows very huge, while the other two database approaches have better size controlled. These experimental results were also supported by another study done by[5].

In another study,[6] proposed mapping XML document into key-value database which can improve the speed of processing query. In their study, Redis database, an open source and key-value database, will be used to develop XED and the competitor will be eXist, Berkeley DB XML and BaseX which used to develop NXD. In their study, four different sizes of XML documents (ranging from 1MB to 30MB) were used to measure the memory usage. The time durations of the different mapping types and memory usage for different database approaches with various document sizes were recorded. The results indicated that eXist and BaseX have the shortest time (about 12 – 20 times) faster than Redis when saving and loading XML document with 30MB of document size. Nevertheless, the memory usage of Redis is smaller than the three NXDs; especially the Berkeley DB has the biggest memory usage which is 250MB when the XML document size is up to 30MB.

Recently,[7] listed the factors such as flexibility, performance, attributes sizes and so on, which influence the choice between XML and relational database. In addition,

they also run several experimental evaluations to compare the databases.

3. XR-Gine: The Simulation Engine

XR-Gine is a simulator engine to measure the performance of the XED approach through handling several basic database operations. In distinct, Microsoft SQL Server 2014, Oracle 11g Database and IBM DB2 9.7 are selected as the database products for comparison. The evaluations carried out include the (1) storage size, (2) time taken to load data, and (3) time taken for query retrieval. The experimental results or XR-Gine provide insight on the strengths and weaknesses of three selected XED on various cases: (i) dataset sizes, (ii) dataset types, and (iii) query types.

Firstly, the database connections need to be configured (see Figure 1). User needs to key in the host, port, database name, user name and password to log in the database. After enter all the necessary data, user needs to click the connect button to connect the database.

Once the connections are established, the performance evaluation can be carried out under the XML test tab. The screen allows user to load any XML file into database or they may choose the predefined files obtained from University of Washington repository[8]: Yahoo.xml (small-sized), DBLP (medium-sized), and PSD7003 (large-sized).

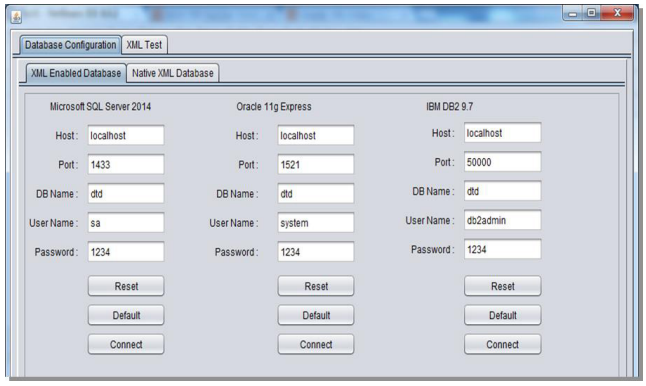


Figure 1. Database configuration screen.

4. Performance Evaluation on Storage and Retrieval

In the evaluation, all the back-end services of the operating system will stopped to prevent any conflict or any unwanted affect during storing evaluation. In addition, it also prevent out of memory exception.

Table 2 depicts the database creation and loading time. From the table, the small XML datasets which is Yahoo takes few seconds to finish the storing process. In the result, IBM DB2 has the highest time consumed compared to the other two XEDs. Also, we noted that the Microsoft SQL Server and Oracle has almost similar performance in all sizes of dataset.

Table 2. Database creation and data insertion time

XED	Database Creation		
	Data Insertion Time (Seconds)		
	Yahoo	DBLP	PSD7003
MS SQL	1.16	4914.93 (1.36hr)	25592.73 (7.1hr)
IBM DB2	2.59	5398.96 (1.5hr)	45455.03 (12.63hr)
Oracle	1.15	4883.57 (1.36hr)	25211.11 (7 hr)

In the retrieval evaluation stage, six queries were prepared for each dataset. The time taken for retrieval will be measured in milliseconds. In the Yahoo dataset, the response times for six queries in the three XEDs are insignificant because the XML file is small and the number of returned result is small. As such, we have omitted the results in this paper.

We focus our discussion on the DBLP and PSD7003 datasets. Table 3 depicts the query on DBLP dataset, while Figure 2 shows the query evaluation results.

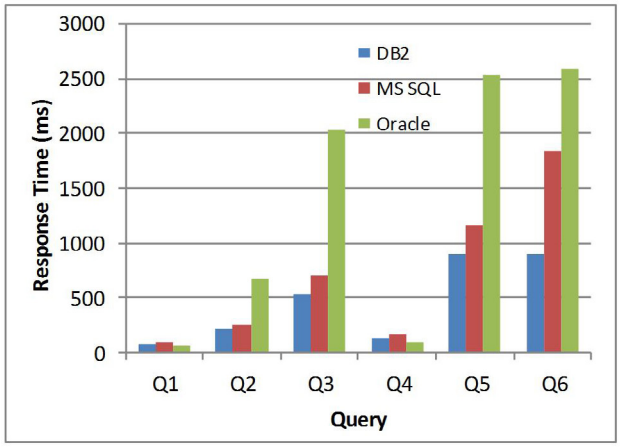
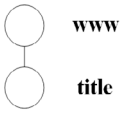
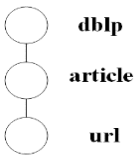
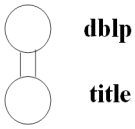
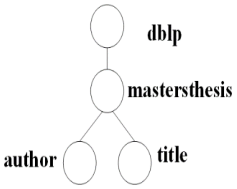
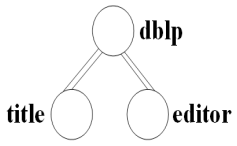
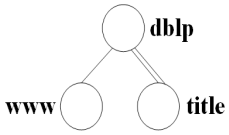


Figure 2. Query evaluation results on DBLP dataset.

From the evaluation, we observed the followings from Figure 2:

Table 3. Query description using DBLP dataset

Query No	Description	XPath	Corresponding SQL query	Number of returned result
Q1	List out all the information which consists of www, with any node title.		select a.text from title_dblp a, www_dblp b where a.parentid=b.selfid	38
Q2	List out all the information which consists of article, with its respective immediate node url.		select a.text from url_dblp a, article_dblp b, dblp_ dblp c where a.parentid=b.selfid and b.parentid=c. selfid	111577
Q3	List out all the information which consists of dblp, with its respective immediate node title.		select a.text from title_dblp a left join mastersthesis_ dblp b on a.parentID=b.selfID left join article_dblp c on a.parentID=c. selfID left join phdthesis_dblp d on a.parentID=d. selfID left join www_dblp e on a.parentID=e. selfID left join proceedings_dblp f on a.parentID=f. selfID left join inproceedings_dblp g on a.parentID=g. selfID	328859
Q4	List out all the information which consists of mastersthesis, with its respective immediate node author and title.		select a.text from author_dblp a, mastersthesis_dblp b, dblp_dblp c where a.parentID=b.selfID and b.parentID=c.selfID union all select a.text from title_dblp a, mastersthesis_dblp b, dblp_dblp c where a.parentID=b.selfID and b.parentID=c.selfID	10
Q5	List out all the information which consists of dblp, with any title or editor.		select a.text from title_dblp a left join mastersthesis_dblp b on a.parentID=b.selfID left join article_dblp c on a.parentID=c.selfID left join inproceedings_dblp d on a.parentID=d.selfID left join proceedings_dblp e on a.parentID=e. selfID left join book_dblp f on a.parentID=f.selfID left join incollection_dblp g on a.parentID=g. selfID left join phdthesis_dblp h on a.parentID=h. selfID left join www_dblp i on a.parentID=i.selfID union all select a.text from editor_dblp a left join mastersthesis_dblp b on a.parentID=b.selfID left join article_dblp c on a.parentID=c.selfID left join inproceedings_dblp d on a.parentID=d.selfID left join proceedings_dblp e on a.parentID=e.selfID left join book_dblp f on a.parentID=f.selfID left join incollection_dblp g on a.parentID=g.selfID left join phdthesis_dblp h on a.parentID=h.selfID left join www_dblp i on a.parentID=i.selfID	335247

Q6	List out all the information which consists of dblp, with its respective immediate node www, and any node which has title.		select c.text,b.text,a.[key] from www_dblp a left join url_dblp b on a.selfID=b.parentID left join title_dblp c on a.selfID=c.parentID left join dblp_dblp d on a.parentID=d.selfID union all select a.text,null,null from title_dblp a left join mastersthesis_dblp b on a.parentID=b.selfID left join article_dblp c on a.parentID=c.selfID left join inproceedings_dblp d on a.parentID=d.selfID left join proceedings_dblp e on a.parentID=e.selfID left join book_dblp f on a.parentID=f.selfID left join incollection_dblp g on a.parentID=g.selfID left join phdthesis_dblp h on a.parentID=h.selfID left join www_dblp i on a.parentID=i.selfID	328897

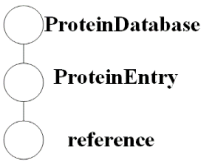
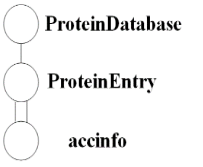
(1) Oracle outperformed IBM DB2 and SQL Server in terms of retrieving query which returning only small number of rows (Q1, Q4), due to the default fetch size of Oracle is small.

(2) For query which returned large number of rows, for example Q2, Q3, Q4 and Q6, Oracle performance is the worst as the default fetches size is small. Oracle allow small amount of data fetch into java program each time which affect the time consuming while the return result is larger. We have run an experiment to proof the fetch size of Oracle database is smaller than others and it will affect the query retrieval time. In the experiment, we change the fetch size to a thousand and query retrieval time was reduced from 672ms to 369.33ms. This result certified that the fetch size affects the performance of retrieving data in Oracle database. But, the results produced by Oracle database in this experiment is still based on the default fetch size which is fair to other two XEDs.

(3) In Q3, the query retrieval time for Microsoft SQL Server is 24% slower than IBM DB2. This is due to the reason that the left join function in the query. We have run an experiment to proof that left join function in the query will affect the query retrieval time. In the experiment, we observed that when there is more than two left join function in the Microsoft SQL Server, the retrieval time will be slower than IBM DB2. Besides, retrieval time of Microsoft SQL Server is 6.82% faster than IBM DB2 when a left join function contains in the query. When there is two, four and six left join function in the query, the IBM DB2 become 11.7%, 17.7% and 30.5% faster than Microsoft SQL.

(4) In Q6, which is the longest and most complex query, we observed that Oracle database is slower 2.88 magnitudes than IBM DB2. In the other hand, Microsoft SQL Server is almost 2 magnitudes slower than IBM DB2. This is due to a total of eleven left join functions in the query.

Table 4. Query description using PSD7003 dataset

Query No	Description	XPath	Corresponding SQL query	Number of returned result
Q1	List out all the information which consists of ProteinEntry, with its respective immediate node reference.		select a.text from reference_PD a left join ProteinEntry_PD b on a.parentID=b.selfID left join ProteinDatabase_PD c on b.parentID=c.selfID	314763
Q2	List out all the information which consists of ProteinEntry, with its respective immediate node accinfo.		select a.text from accinfo_pd a left join reference_pd b on a.parentid=b.selfid left join proteinentry_pd c on b.parentid=c.selfid left join database_pd d on c.parentid=d.selfid left join proteindatabase_pd e on d.parentid=e.selfid	312506

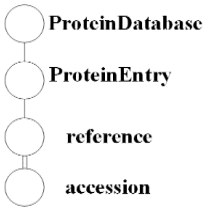
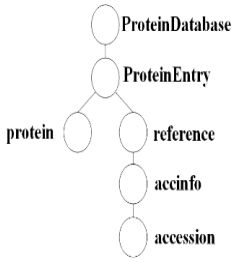
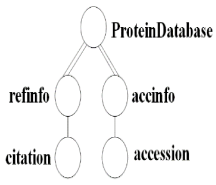
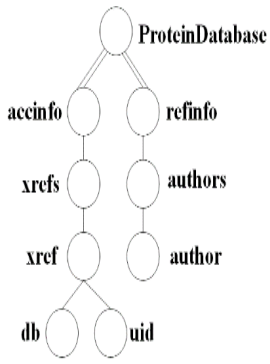
Q3	List out all the information which consists of ProteinEntry, with its respective immediate node reference which is with its respective immediate node accinfo.		select a.text from accession_pd a right join accinfo_pd b on a.parentid=b.selfid left join reference_pd c on b.parentid=c.selfid left join proteinentry_pd d on c.parentid=d.selfid left join database_pd e on d.parentid=e.selfid left join proteindatabase_pd f on e.parentid=f. selfid	312506
Q4	List out all the information which consists of ProteinEntry, with its respective immediate node Protein and reference which consists of the accinfo with its respective immediate node accession.		select a.text from protein_PD a left join ProteinEntry_PD b on a.parentID b.selfID left join Database_PD c on b.parentID=c.selfID union all select a.text from accession_PD a right join accinfo_PD b on a.parentID=b.selfID left join reference_PD c on b.parentID=c.selfID left join ProteinEntry_PD d on c.parentID=d. selfID left join ProteinDatabase_PD e on d.parentID=e.selfID	**575031**
Q5	List out all the information which consists of both refinfo and accinfo, with thier respective immediate node citation and accession.		select a.text from citation_PD a right join refinfo_PD b on a.parentID=b.selfID left join reference_PD c on b.parentID=c.selfID left join Database_PD d on c.parentID=d.selfID union all select a.text from accession_PD a right join accinfo_PD b on a.parentID=b.selfID left join reference_PD c on b.parentID=c.selfID left join Database_PD d on c.parentID=d.selfID	335247
Q6	List out all the information which consists of both accinfo and refinfo, with thier respective immediate node xrefs which consists of xref with its respective immediate node db and uid; and authors which consists of author.		select a.text from db_PD a left join xref_PD b on a.parentID=b.selfID left join xrefs_PD c on b.parentID=c.selfID inner join accinfo_ PD d on c.parentID=d.selfID inner join reference_PD e on d.parentID=e.selfID left join ProteinDatabase_PD f on e.parentID=f. selfID union all select a.text from uid_PD a left join xref_PD b on a.parentID=b.selfID left join xrefs_PD c on b.parentID=c.selfID inner join accinfo_PD d on c.parentID=d.selfID inner join reference_PD e on d.parentID=e. selfID left join Database_PD f on e.parentID=f. selfID union all select a.text from author_PD a left join authors_PD b on a.parentID=b.selfID left join refinfo_PD c on b.parentID=c.selfID left join reference_PD d on c.parentID=d. selfID left join ProteinDatabase_PD e on d.parentID=e.selfID	8068246

Table 4 depicts the query on PSD7003 dataset, while Figure 3 shows the query evaluation results.

From Figure 3, we observed the followings:

(1) In Q1, IBM DB2 is 88.48% faster than Oracle database due to the reason that the size of PSD7003 dataset is 705MB. The number of returned result is close to hundred thousand rows of data, as such, the same observation was made, where by Oracle is the slowest due to its default fetch size.

(2) In Q2, there is four left joins required. When the query required more than two joins, the performance of Microsoft SQL Server will dropped and becomes slower than IBM DB2.

(3) In Q3, Q4, and Q5, although there are several joins required, we observed that Microsoft SQL Server has the lowest growth rate compared to the others, which is about 10%. The performance of IBM DB2 also degraded due to the fetch row size limit.

(4) In Q6, with the huge number of returned rows (8068246 rows), the performance of Oracle dropped severely. In addition, we observed that the query retrieval time of IBM DB2 is 6% longer than Microsoft SQL Server.

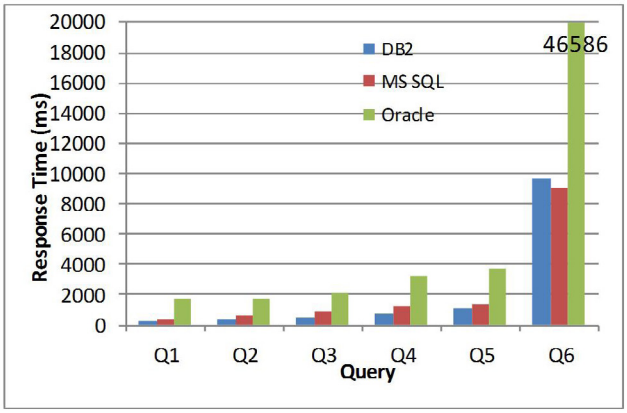


Figure 3. Query evaluation results on PSD7003 dataset.

5. Conclusion and Suggestion

Based on the results obtained, the best performance of storing evaluation is Oracle. On the other hand, the query retrieval time for IBM DB2 is faster compared to the other two on a small to medium sized datasets. We also observed that when the returned results are very huge (thousands of rows), Microsoft SQL Server performs best. As such, Microsoft SQL Server is considered the best as supporting huge dataset is a necessity with the overwhelming data produced across the web.

6. References

1. Soujanya KML, Mathi S. Extensible markup language databases: a study. Indian Journal of Science and Technology. 2016; 9(9):1–7.
2. Noaman AY, Mansour AAA. A comparative study between two types of database management systems: XML-enabled relational and native XML. World Applied Sciences Journal. 2012; 19(7):972–985.
3. Lee KKY, Tang WC, Choi KS. Alternatives to relational database: comparison of NoSQL and XML approaches for clinical data storage. Computer Methods and Programs in Biomedicine. 2012; 110(1):99–109.
4. Freire SM, Teodoro D, Kleiner FW, Sundvall E, Karlsson D, Lambrix P. Comparing the performance of NoSQL approaches for managing archetype-based electronic health record data. *PLOS ONE*. 2016; 11(3): e0150069.
5. HL7-CDA [Internet]. 2015 [cited 2015 Jun 1]. Available from: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7. Date accessed: 01/06/2015
6. Strnadm P, Macek O, Jira P. Mapping XML to key-value database. In the International Conference on Advances in Database, Knowledge and Data Application; 2013. p. 121–7.
7. Kheder MQ, Rahman CM, Jamal SJ. A comparison of concepts between native XML and relational database systems. Asian Journal of Natural and Applied Sciences. 2015; 4(4):49–62.
8. UW [Internet]. 2015 [cited 2015 Jun 1]. Available from: http://www.cs.washington.edu/research/xmldatasets/.