Indian Journal of Science and Technology, Vol 9(47), DOI: 10.17485/ijst/2016/v9i47/108687, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

# A Comparison and Benchmarking on Data Storage and Query Retrieval for Native XML Databases

**Su-Cheng Haw* and Wai-Lin Chong**

Faculty of Computing and Informatics, Jalan Multimedia, 63100 Cyberjaya, Malaysia;
sucheng@mmu.edu.my, wailin@gmail.com

## Abstract

**Objectives:** Native XML Databases (NXD) is one of the technologies for storing and retrieving XML documents. Basically, a good database should cater for efficient storage and fast retrieval. **Methods/Statistical Analysis**: In NXD, it creates a logical model to store and retrieve XML documents with no mapping required. In another words, the storage and retrieval operation are most optimized in its native format. **Findings:** NativeGine is a simulation engine to compare the performances of three existing open source native XML database products through handling several basic database operations. In distinct, Xindice, eXist and Sedna will be examined in terms of storing and query retrieval speed. Based on the results obtained, Sedna has the best performance for database creation and data insertion. Yet, eXist gains the outstanding performance on query retrieval as it has the minimum response time for all types of queries. **Application/Improvements:** The results generated by the engine are beneficial to aid the community to choose the appropriate database system.

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

## 1. Introduction

There are several NXD systems in this community such as Xindice, eXist, dbXML, TIMBER, Sedna, NextDB, DB4XML, Berkeley DB XML and OrientX. NXD systems can be categorized into two groups, open source and commercial. In this paper, Xindice, eXist and Sedna are chosen to be evaluated because these are open sources. In addition, the descriptions of these three types will be briefly introduced in the following subsection. At the end of the evaluation, the performance results in terms of time taken for storage and retrieval, as well as the weaknesses and strengths of the products will be stated.

Xindice is developed by Apache Group and it is fully written in Java programming language. eXist, however, is an open source product under GNU LGPL license and is written completely by Java programming. On the other hand, Sedna is created by C and C++ programming language but the developer group, Team MODIS have created several APIs for Sedna to extend their future works.

Indexing in Xindice is manually done through command line tool. The main objective of the command line tool is to create a direct interaction with database for retrieving data purpose. Besides that, indexing in eXist is automatically done by storing XML documents as a DOM tree. Numerical indexing schema is provided by eXist to identify structural relationships between node in the fastest way. Moreover, Sedna provide automatic indexing by index manager. Sedna uses the descriptive schema as a storage strategy to enhance the query execution process.

Similarly, these three NXD systems are utilizing model-based storage strategy. Besides that, they are using the same logical unit of XML documents as well, it is collection type. These collections are managed and arranged in a hierarchical style. The advantage of NXDs over relational database is eliminating mapping of XML documents to a query language like SQL.

XPath is the query language uses by all these three NXD systems to query XML documents and store inside a collection. In addition, eXist has an advantage as it sup-

*Author for correspondence

**Table 1.** Features comparison of the three Native XML systems

| NXD | Xindice | eXist | Sedna |
|---|---|---|---|
| ***Developer*** | Apache Group | Wolfgang Meier | Team MODIS |
| ***Technology*** | Java | Java | C++ , Java |
| ***Indexing*** | automatic | automatic | automatic |
| ***Storage Strategy*** | Model-Based | Model-Based | Schema-Based Clustering |
| ***Logical Unit*** | Collection | Collection | Collection |
| ***Query Language*** | XPath | XPath/XQuery | XPath/XQuery |
| ***Update Language*** | XUpdate | XUpdate | XUpdate |

ports most of the XQuery query language. Last but not least, these three NXD systems use XUpdate as their primary update language. Also, XUpdate is a standard update language proposed by the XML:DB initiative for updating XML documents purpose.

Table 1 summarizes the comparison of the three open source NXDs, which is Xindice, eXist and Sedna.

Most of the past literatures[1–3] on NXD systems merely introduces the features, strengths and beauty of the systems. Yet, the researchers rarely evaluate and compare the performance of their introduced NXDs with other systems. Therefore, disadvantages or weaknesses are hardly to be figured out without benchmarking and comparing to other systems[4,5].

[6]Compared different types of NXDs and experiment them with INEX dataset[7]. In their study, eXist and Xindice have been selected to become the experiment products. They used 536 MB of INEX dataset (version 1.4), which is roughly 12,107 articles from 6 IEEE transactions and 12 journals from year 1995 to 2002. A set of XPath queries, such as selecting nodes, paths and so forth has been prepared for the evaluation. Xindice took 25 minutes to load the dataset into database while eXist took 97 minutes.

However,[8] compared a benchmark between a native XML database and relational database. Due to the fundamental differences between two databases, the benchmarking has been presented in the different methods. The professional native XML database product, X-Hive has been selected as subject for the benchmarking, while Microsoft Access 2003 has been chosen for representing the relational database. In their evaluation, five sets of queries such as standard queries, hierarchical, sequential, and relational and string format were composed in XQuery format. In their evaluation results, there is a significant difference in terms of performance; Microsoft Access 2003 does run much faster than X-Hive. As such, they have concluded that relational XML database is a better suited for handling XML document compared to native XML database.

Additionally,[9] presented a performance comparison of the currently existing alternatives for XML document in databases. A new benchmark, HYBE has been introduced by the authors to include hybrid systems in the performance analysis and supports various query alternatives, like XQuery, XPath, XUpdate and so forth. There are several query sets prepared in this experiment, such as inserting 100 XML documents into database, retrieving a full document specified by an ID and so on. IBM DB2 9.5 has been selected to be the experiment tools as it supports shredding, clobbing and the hybrid storage. The final result shows that the hybrid XML storage has performance to native XML storage and performs better for data-oriented documents than document-oriented documents.

On the other hand,[10] presented an implementation of a Resource Directory (RD) based on a native XML database. An open-source native XML database, Sedna and a relational database, MYSQL have been selected as subjects for the experiment. The first set of tests is to process a various number of resource descriptions in the database with several requests, while, the second set of tests is performed on the real RD data from the EcoBus system. The study shows that the XML based RD gives more flexible control of the resources and shorter execution time, while SQL based RD provides shorter response time.

[3]Compared the performances of three open source NXDs in handling standard database operations. eXist, Xindice and Berkeley DBXML have been chosen in the experiment. The experiment was conducted on a small scale of dataset. The experiment has been divided into two parts, which is database creation and database querying on various sizes of database collections. At the end of the

experiment, the winner of database creation are Berkeley DBXML, follow by Xindice and the slowest with eXist. Whereas, eXist have the best response time in querying database and Xindice is the slowest among three of it.

In a more recent study,[4] examined the differences between XML and relational database in the perspectives of data representation, query writing and query evaluation. Their study revealed that although processing data in XML database is relatively fast, parsing and interpreting the XML needs time. Hence, if an application required heavy loading, then the data should be stored in relational instead of XML.

From the review, most researchers do not conduct the performance test on large dataset. This is critical especially we are in the data-oriented paradigm. In the absent of this, we propose NativeGine to perform the evaluation to test the performance of the selected NXDs on various sizes of dataset, especially large size.

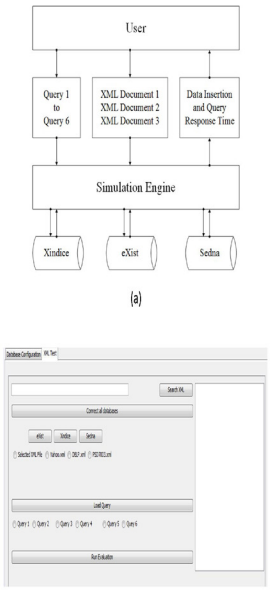



**Figure 1** (a). Architecture diagram of NativeGine. (b) Main screen of NativeGine.

## 2. NativeGine: The Simulation Engine

Figure 1(a) depicts the system architecture of NativeGine, while Figure 1(b) depicts the main screen of NativeGine. The evaluation processes in different databases can be done via the procedures as follows: 1. Connecting to the chosen database, 2. Inserting XML documents into the chosen database, 3. Executing various type of queries to the database, and 4. Calculating the time taken for data insertion and query retrieval.

## 3. Performance Evaluation

### 3.1 Experimental Setup

In the evaluation, the elapsed time for storing and retrieval using three sets of XML files on various sizes (small, medium and large) will be measured. These datasets were obtained from University of Washington repository[11] as depicted in Table 2. All the experiments are performed on Intel i7-2630QM processor with 8 GB RAM running on Windows 7 64bits.

**Table 2.** XML datasets details

| Dataset | Document Name | File Size | Title |
|---|---|---|---|
| **Yahoo** | yahoo.xml | 25 KB | Small |
| **DBLP** | dblp.xml | 130.726 MB | Medium |
| **Protein** | psd7003.xml | 722.585 MB | Large |

For the retrieval evaluation, we have prepared three path queries and three twig queries with different levels of complexity for each NXDs. In another words, we have 18 sets of queries. Path queries are simpler and direct query that return outputs in only one direction of the tree, whereas twig queries are more complex and it tends to return outputs from two and more branches of the tree.

### 3.2 Performance Results and Discussions

#### *3.2.1 Storing Evaluation*

Table 3 shows the data insertion time for all the XML datasets and NXDs in terms of milliseconds. From the result, Sedna is the fastest in data storing, while eXist is the slowest among all of them.

#### *3.2.2 Retrieval Evaluation*

Table 4 lists the query description and the corresponding graphical representation of query on Yahoo dataset, while Table 5 illustrates the XPath notations the for retrieval evaluation. From Table 5, we noted that the XPath notations are almost similar on all the approaches. As such,

for the evaluation on the next dataset, we will omit this from the paper.

**Table 3.** Data loading and insertion time

| NXDs | Data Storing Time (ms) | | |
|---|---|---|---|
| | Yahoo (Small) | DBLP (Medium) | Protein (Large) |
| **eXist** | 281 | 93559 (1min 33sec) | 423890 (7 min 4sec) |
| **Xindice** | 94 | 53896 (54sec) | 448975 (7min 28sec) |
| **Sedna** | 93 | 82675 (1min 23sec) | 158418 (2min 38sec) |

**Table 4.** Query description and query node on Yahoo dataset

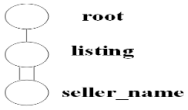
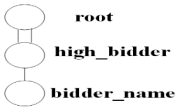
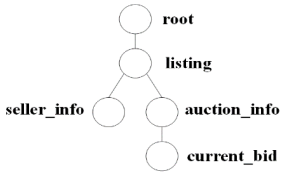
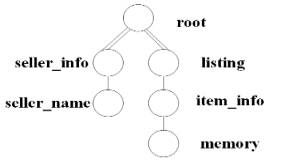
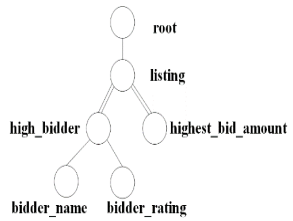
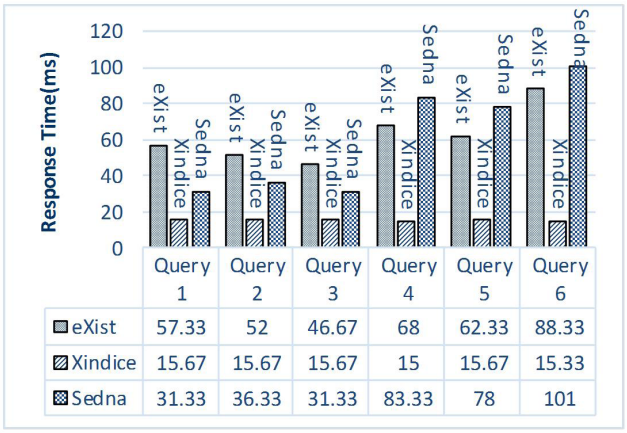
| Query No. | Query Description | Query Node |
|---|---|---|
| Q1 | Retrieve all the information for the *current_bid* which is belong under *listing* or *auction_info* |   |
| **Q2** | List any *seller_name* under *listing* node |   |
| **Q3** | List *high_bidder* which consists of immediate node *bidder_name* |   |
| **Q4** | Retrieve all the information which consists of both *seller_info* and *current_bid* under *listing* node |   |
| Q5 | Retrieve all the information which consists of both *seller_info* and *item_info*, which their immediate nodes *seller_name* and *memory* |   |
| **Q6** | List *high_bidder* which consists of two immediate nodes *bidder_name*, *bidder_rating* and *highest_bid_amount* which is belong under *listing* node |   |

**Table 5.** XPath notation on Yahoo dataset

| Query | eXist | Sedna | Xindice |
|---|---|---|---|
| Q1 | doc('yahoo.xml')/root/listing/auction_info/current_bid | doc('yahoo')/root/listing/auction_info/current_bid | /root/listing/auction_info/current_bid |
| Q2 | doc('yahoo.xml')/root/listing//seller_name | doc('yahoo')/root/listing//seller_name | /root/listing//seller_name |
| Q3 | doc('yahoo.xml')/root//high_bidder/bidder_name | doc('yahoo')/root//high_bidder/bidder_name | /root//high_bidder/bidder_name |
| Q4 | doc('yahoo.xml')/root/listing/seller_info/seller_name\|/root/listing/auction_info/current_bid | doc('yahoo')/root/listing/seller_info/seller_name\|/root/listing/auction_info/current_bid | /root/listing/seller_info/seller_name\|/root/listing/auction_info/current_bid |
| Q5 | doc('yahoo.xml')/root//seller_info/seller_name\|/root//item_info/memory | doc('yahoo')/root//seller_info/seller_name\|/root//item_info/memory | /root//seller_info/seller_name\|/root//item_info/memory |
| Q6 | doc('yahoo.xml')/root/listing//high_bidder/bidder_name\|/root/listing//high_bidder/bidder_rating\|/root/listing//highest_bid_amount | doc('yahoo')/root/listing//high_bidder/bidder_name\|/root/listing//high_bidder/bidder_rating\|/root/listing//highest_bid_amount | /root/listing//high_bidder/bidder_name\|/root/listing//high_bidder/bidder_rating\|/root/listing//highest_bid_amount |

Figure 2 depicts the query retrieval results on the small dataset which is Yahoo dataset. Based on the figure,

Xindice has the rapid response time on the small datasets, while eXist and Sedna has almost the similar performance results. It is also observed that Xindice supports twig queries well, while eXist and Sedna are much slower (about 6 to 7 times much slower than Xindice).




| | Query 1 | Query 2 | Query 3 | Query 4 | Query 5 | Query 6 |
|---|---|---|---|---|---|---|
| eXist | 57.33 | 52 | 46.67 | 68 | 62.33 | 88.33 |
| Xindice | 15.67 | 15.67 | 15.67 | 15 | 15.67 | 15.33 |
| Sedna | 31.33 | 36.33 | 31.33 | 83.33 | 78 | 101 |

**Figure 2.** Query response time on Yahoo dataset.

The next part of the evaluation is to compare the performance on various approaches using a medium size dataset, i.e. DBLP dataset. The query on DBLP dataset is depicted in Table 6.

**Table 6.** Query description and query node on DBLP dataset

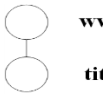
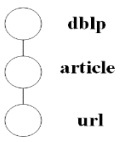
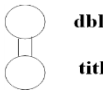
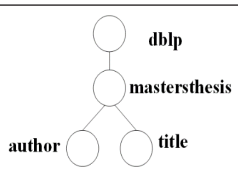
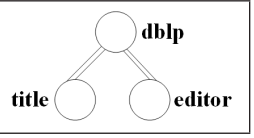
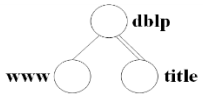
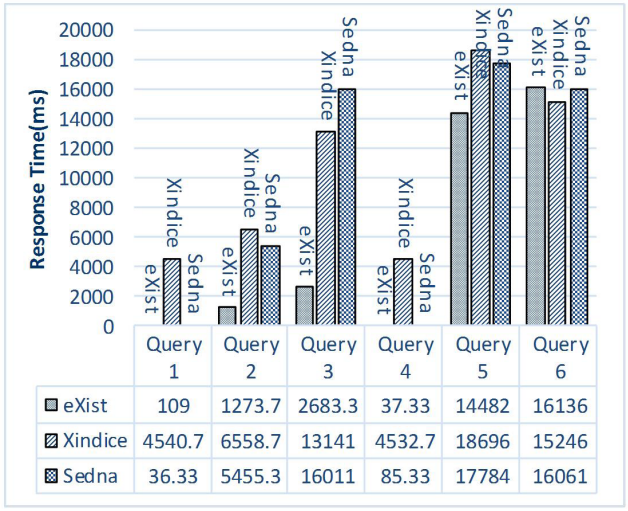
| Query No. | Query Description | Query Node |
|---|---|---|
| Q1 | List any *titles* which belongs is belong to *www* node |   |
| Q2 | List any *url* which is belong to *article* node |   |
| **Q3** | List any *title* under *dblp* node |   |
| **Q4** | Retrieve all the information which consists of both *author* and *title* under *mastersthesis* node |   |
| **Q5** | Retrieve all the information which consists of both *title* and *editor* |   |
| **Q6** | Retrieve all the information which consists of both *www* and *title* |   |

Figure 3 shows the query retrieval results on the DBLP dataset. eXist gains the best performance to retrieve medium size dataset, while Sedna got the second ranking for it. From the figure, it shows that it takes longer time to retrieve output through Xindice. This is consistent with the experimental results done by[6], whereby Xindice has totally failed in the experiment due to the memory leak issue. Thus, we can conclude that the retrieval speed of Xindice operates well in small datasets but not in large datasets. On the opposite site, eXist shows much better behavior by providing a user-friendly interface for database management.



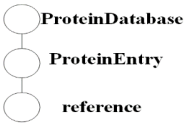
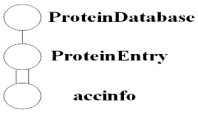
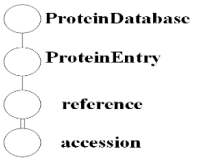
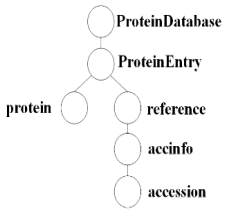
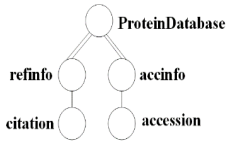
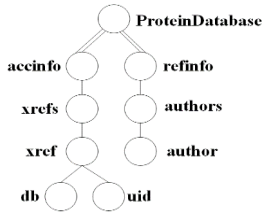

| | Query 1 | Query 2 | Query 3 | Query 4 | Query 5 | Query 6 |
|---|---|---|---|---|---|---|
| eXist | 109 | 1273.7 | 2683.3 | 37.33 | 14482 | 16136 |
| Xindice | 4540.7 | 6558.7 | 13141 | 4532.7 | 18696 | 15246 |
| Sedna | 36.33 | 5455.3 | 16011 | 85.33 | 17784 | 16061 |

**Figure 3.** Query response time on DBLP dataset.

Table 7 illustrates the query description and XPath notation of the query on Protein dataset.

Figure 4 depicts the overall retrieval results on Protein dataset. Based on Figure 4, eXist has the best performance over the other two NXDs. Sedna has the worst performance among all with a significant difference. This clearly indicates that it takes longer to retrieve output through Sedna than the others. This is obviously observed on complex query Q6. This is because the prebuild function of Sedna, called SednaSerializedResult cause the slowness issue. Furthermore, the cache memory in Sedna is lesser compared to other NXDs, it limits the fetching result and gaining low performance to retrieve millions of data.

**Table 7.** Query description and query node on protein dataset

| Query No. | Query Description | Query Node |
|---|---|---|
| Q1 | List any *references* under *ProteinEntry* node |  |
| **Q2** | Retrieve all *accinfo* results under *ProteinEntry* node |  |
| **Q3** | List any *reference* information, which consists of *accession* node |  |
| **Q4** | Retrieve all the information which consists of both *protein* and *accession* under *ProteinEntry* node |  |
| **Q5** | Retrieve all the information which consists of both *refinfo* and *accinfo*, which their immediate nodes *citation* and *accession* |  |
| **Q6** | Retrieve all the information which consists of *xref*, which consists of immediate nodes *db* and *uid*, also list any *author* under *authors* node |  |

## 4. Conclusion and Future Works

In order to construct NativeGine, it is crucial for us to review the open source NXDs features, advantages and disadvantages. As such, the first part of the paper is to provide some background on these approaches. Next, NativeGine is built and experimental evaluation has been conducted. Based on the results obtained, Sedna has the best performance for database creation and data insertion. Yet, eXist gains the outstanding performance on query retrieval as it has the minimum response time for all types of queries. On the other hand, Xindice has the rapid response time on small datasets but not on the large datasets due to the memory leak issue. Thus, we can conclude that the retrieval speed of Xindice operates very well in small datasets but not in large datasets. Conversely, eXist shows much better behavior by providing a user-friendly interface for database management. Therefore, eXist is the winner throughout the experiments.

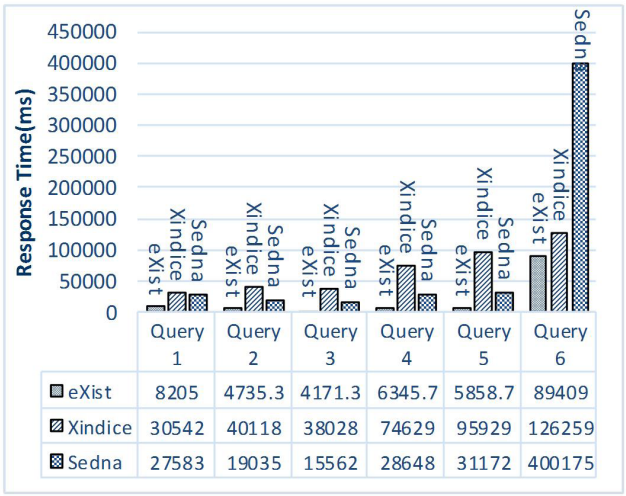



**Figure 4.** Query response time on protein dataset.

In the future works, we wish to extend the experimental evaluation on other Native XML databases. In addition, the experiment could also be carried out to compare the performance of native storage as compared to XML Enabled Database.

## 5. References

1. Meier W. eXist: An open source native XML database. International Workshop on Web-Services and Database Systems; 2002. p. 168–83.
2. Grinev M, Fomichev A, Kuznetsoc S. Sedna: A Native XML DBMS. International Conference on Current Trends in Theory and Practice of Computer Science; 2006. p. 272–81.
3. Mabanza N, Chadwick J. A comparison of open source Native XML Database products. International Conference Advanced Communication Technology; 2013. p. 8–12.
4. Kheder MQ, Rahman C, Jamal S. A comparison of concepts between Native XML and relational database systems. Asian Journal of Natural and Applied Sciences. 2015; 4(4):49–62.
5. Kolar P, Loupal P. Comparison of Native XML Databases and experimenting with INEX. International Conference on Dateso; 2006. p. 116–9.

6. Soujanya KML, Mathi S. Extensible Markup Language Databases: A Study. Indian Journal of Science and Technology. 2016; 9(9):1–7.
7. Fuhr N, Gvert N, Kazai G, Lalmas M. Initiative for the evaluation of XML retrieval (INEX). International Workshop Proceedings; 2003; p. 1–9.
8. Alexander VDL, Darbyshire P. A benchmark comparison between Native XML and relational databases. International Conference on Information Resources Management; 2005. p. 216–9.
9. Hopfner H, Schad J, Mansour E. On analyzing the database performance for different classes of XML documents based on the used storage approach. International Coference on of ICSOFT; 2009.
10. Jokic S, Krco S, Vuckovic J, Gligoric N, Drajic D. Evaluation of an XML Database based resource directory performance. International Conference on Telecommunications Forum; 2011. p. 542–5.
11. University of Washington XML Data Repository. Available from: http://www.cs.washington.edu/research/xmldatasets/