Open Source Native XML Database Architectures - A Comparative Study

Ntima Mabanza
School of Information Technology
Central University of Technology (CUT), Free State
Private Bag X20539, Bloemfontein 9300, South Africa
Email: Nmabanza@cut.ac.za

***Abstract*— Text-based and model-based architectures are two models used by Open Source Native XML databases (NXDs) to physically store collections of XML documents. The question is: which one of these two models is better in terms of performance. This paper provides some insights into these two NXD architectures. Additionally, the metric performance comparison of text-based and model-based architectures it observed on three Open Source NXD products specifically dbXML, eXist, and Bekerley DBXML that implement either of these two NXDs architectures. The different methods used by each of these Open Source NXD products to manage collections of XML documents are discussed and time taken to store different number of XML documents is analyzed and compared.**

***Keywords*— Open Source Native XML database Architectures, XML documents, metric performance comparison.**

I. INTRODUCTION

The success of Extensible Markup Language (XML) [1] technology since its launch in 1998 has led to an augmentation of the volume of XML documents generated in business transactions, as well as on the Web. The contents of XML documents vary from simple, complex, and mixed elements. As a result of that, XML documents are often categorized into two main groups. These include Data Centric and Document Centric documents. The two types of XML document mentioned above differ in their structure and characteristics. In a Data Centric document, XML elements can contain other elements or data. Also, the order of XML elements and overall physical structure are often unimportant. The following are few examples of Data Centric documents: price lists, stock quotes, sales order, and scientific data. Whereas with Document Centric documents the order XML elements and overall physical structure are very important and do normally matter. Advertisements, books, and email represent a few examples of Document Centric documents.

In spite of the differences between Data Centric and Document Centric documents, they both follow XML specification and the distinction between the two is not always clear. For that reason, it is possible to combine them both to form hybrid documents [2]. The dissimilarity in XML documents structure and characteristics has necessitated database researchers to find innovative methods for handling XML documents. Already, there have been numerous efforts amongst database researchers and practitioners for improving, adapting traditional databases (i.e. relational databases, object oriented databases, etc...) to better handle XML documents. There are still some kinds of underperformances when using traditional databases to handle XML documents. Lack of perfect mapping between XML data and the relational schema is one example of traditional databases underperformances. Native XML databases (NXDs) are some kind of specialized databases. They have been specifically created, designed and developed as alternative solutions for addressing the traditional database's underperformances with regards to managing XML documents. The most significant difference between traditional databases and NXDs are that the internal models of NXDs are based on XML.

Consequently this gives NXDs an advantage to manage collections of XML documents efficiently than their traditional databases counterparts. The latter category of database technology will be the focus of this research paper.

The purpose of this paper is to analyse and compare the internal models of open source NXD products in terms of their capabilities to efficiently manage XML documents.

The rest of the paper is organized as follows. Section II introduces and compares the different NXD Architectures. Section III surveys some of the selected Open Source NXD products that implement the studied NXD architectures. Section IV describes how the performance study of the selected Open Source NXD products was conducted. Section V presents the experimental findings about the selected Open Source NXD products. Lastly, section VI concludes the paper with discussion about the findings, including lessons learnt.

II. NXD ARCHITECTURES

Wingenious [3] defines database architecture as "the set of specifications, rules, and processes that dictate how data is stored and how data is accessed by components of a system". In other words, database architecture basically refers to the internal organization of a particular database. Ronald Bourret [4] classified NXD architectures in two categories, explicitly a text-based and a model-based.

The next subsections A and B will introduce and give an overview of text-based and model-based NXD architectures. A brief comparison of these two will follow in section C.

Before doing so, let's consider the simple XML document shown in Figure 1 which contains some information about various Schools in a certain Faculty. Let use it in order to

illustrate the various approaches in which each of the aforementioned NXD architectures will manage that simple document. This will help as well to have an idea about how each of these two NXD architectures work.

```
  <? xml version=”1.0”?>
 <Faculty>
  <School>
    <SchName> IT </SchName>
  </School>
 < School>
    < SchName> Engineering </SchName>
 </School>
 </Faculty>
```

Figure 1. Simple XML document

A. Text-based NXD Architecture

A text-based NXD is a kind of NXD architecture that stores up the entire contents of an XML document in a text structure. In reality, the physical internal organization of a particular text-based NXD and the kind of the database being used determine the manner in which the text structure will be stored. As result, text-based NXDs use different methods for storing the text structure (i.e. contents of an XML document). For example some text-based NXD store the text structure as a file in a file system; or as a Binary Large Object (BLOB) in a relational database, while other uses a proprietary text format to store it.

The Figure 2 illustrates the manner in which text-based NXD handles the contents of the simple XML document shown in Figure 1.

```
<? xml version=”1.0”?>
 <Faculty>
  <School>
    <SchName> IT </SchName>
  </School>
 < School>
    < SchName> Engineering </ScnName>
 </School>
 </Faculty>
```

Figure 2. Text-based views of simple XML document

The Figure 2 above is similar to Figure 1. This indicates that the text-based NXD architecture does not transform the contents of the simple XML document shown in Figure 1 rather it basically stores it as it is, without altering any information. In other words, the text-based NXD architecture approach kept the contents of the simple XML document shown in Figure 1 “intact”.

B. Model-based NXD Architecture

The model-based NXD is a kind of NXD architecture which first determines and builds an internal structure model of the contents of an XML document before storing it. Actually, the way in which the model-based NXDs store the built internal structure model differs depending on the type of database being used. There are different ways of storing the built internal model. Some databases use either relational, or object-oriented database storage model, while other use proprietary format storage model. Although the storage models differ, but whatever physical representation storage used, it needs to support the mixed content and semi-structured nature of a complex XML document.

Figure 3 shows the technique that the model-based architecture will use to build the internal structure model representation of the simple XML document shown in Figure1.

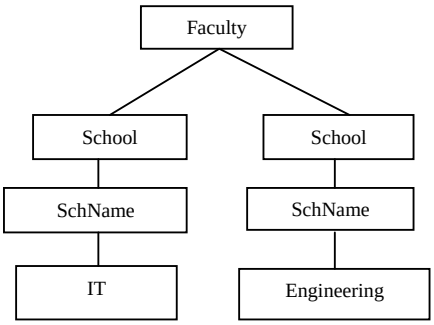

Figure 3. Model-based views of simple XML document

Figure 3 above illustrates the model-based internal object representation model views the simple XML document shown in Figure 1 as a hierarchical structure of nodes. These nodes can be of the type element, attribute, character data, and so on. Additionally to that, these nodes can have different distinctiveness. For example, an element node has a name and may have children. An attribute node has a name and carries text. Finally, character data node carries text but has no name. Looking at the Figure 3, Faculty, School, and SchName are elements whereas IT and Engineering are character data. Additionally, the relationship between these elements is as follow: School and SchName are siblings to each other and also children of Faculty, while all those who appear below these elements are descendants of Faculty.

C. Text-based Vs Model-based NXDs

Text-based and model-based NXDs are the two main architectures for NXDs. They both serve the same purpose which is to manage collection of XML documents. Additionally to that, they also provide some sort of database functionalities for managing the contents of XML documents.

However, these two NXD architectures differ in their ways of handling and storing the contents of XML documents. For example, Text-based NXDs stores XML as a text whereas Model-based NXDs first builds an internal structure model of the XML document before storing it.

The Table 1 provides a brief comparison of the general characteristics of text-based and model-based NXDs architectures in terms of their internal arrangement and storage models.

TABLE 1. ARCHITECTURES GENERAL CHARACTERISTICS

Architectures	Internal Arrangements	Storage Models
Model-based	Build internal object model of XML document	Relational database, object-oriented, or proprietary database.
Text-based	Keep XML document "intact" as text format	A file system, a BLOB in a relational database, proprietary text format.

Three selected Open Source NXD products that implement either of the discussed architectures will be introduced and discussed in the next section.

III. OPEN SOURCE NXD PRODUCTS

Currently there are many open source NXD products. Ronald Bouret [2] makes available a long list of Open Source NXD products available. Therefore, it was necessary to define the selection criteria with regards to the Open Source NXD products used in this study.

Four selection criteria were identified. These criteria included inclusive documentation, project status, community support, including programming language. Based on this, three Open Source NXD products were considered. These include Berkeley DB XML [5], dbXML [6], and eXist [7].
The next three subsections introduce and give a short overview of each of these aforesaid Open Source NXD products.

A. Berkeley DBXML

Berkeley DBXML is an Open Source NXD product developed by SleepyCat Software which implements text-based NXD architecture. Berkeley DBXML's internal organization is based on text-based format. It stores the contents of XML documents in their native formats as text in a proprietary (key-value database) text format (i.e. file). Berkeley DBXML can contain one or more XML documents. Each document stored in Berkeley DB XML can have metadata attributes associated with it.

B. dbXML

dbXMLis an Open Source NXD product developed by the dbXML Group which implements a model-based NXD architecture. dbXML's internal organization is based on internal object model. Prior storing the contents of an XML document, it builds an internal object model based on the contents of that particular XML document and store up that model in a proprietary storage format. dbXML manages XML documents in collections. Collections can be laid in hierarchical fashion similar to an operating system directory structure.

C. eXist

eXist is an Open Source NXD product developed by Wolfgang Meier which implements a model-based NXD architecture. Its internal organization is based on internal object model approach. Like others model-based NXD architectures, prior to storing the contents of an XML document, it builds an internal object model based on the contents of that particular XML document then stores up that model in relational database format. eXist XML documents are stored in a collection. They are managed in hierarchical collections, comparable to storing files in file system.

Table 2 reviews some key characteristics of the Open Source NXD products introduced above.

TABLE 2. CHARACTERISTICS COMPARISON

Developer	Product	Architecture	Database Type
Sleepycat Software	Berkeley DB XML	Text-based	Key-value
dbXML Group	dbXML	Model-based	Proprietary
Wolfang Meier	eXist	Model-based	Relational

The next section describes different steps that were taken for planning and running the experimental performance evaluations of the three selected Open Source NXD products.

IV. BENCHMARK

A benchmark is a comparative performance assessment technique for evaluating the capabilities of a system with the aim of analysing and comparing the pros and cons of different systems architectures. There are numbers of proposed application benchmark techniques for comparing the query processing performances of Native XML database systems. These include X007 [8], XMark [9], and XMach-1[10]. Also there are few studies that attempted to compare the Open Source Native XML database systems against each other. These studies can be categorized into two groups. These two groups are (i) studies comparing the features of Open Source Native XML database systems, and (ii) studies comparing the capacities of some Open Source Native XML database systems for handling common database operations. This paper will focus on the previous studies comparing the capacities of Open Source Native XML database systems in handling common database operations. Some research groups have conducted some performance comparison studies using some of the Open Source Native XML database systems that have been investigated in this paper. This paper, does not intend to use any of the previous studies' results. It conducts its own performance comparison tests. Furthermore, this will help in examining different factors that can affect the performance of Open Source Native XML databases with the aim of exploring the capacities, strengths, weaknesses, and limitations of Open Source Native XML databases.

V. PERFORMANCE COMPARISON

Each one of the open source NXD products discussed in section II was installed and experimental evaluations were conducted on each of those products as well. During the installation process none of the open source NXD products settings were changed (i.e. used default settings).

The following were the aims of the experimental evaluation: (i) to mainly benchmark and compare the metric performances of these Open Source NXD products, (ii) to understand the different factors that can affect the performance of each of these products with regards to their NXD architectures, and (iii) to determine the strength and weaknesses of the products with regards to NXD architectures implemented by each of these products.

Amongst the three experimental evaluation aims mentioned above the emphasis was more on providing a metric assessment of each of these Open Source NXD products with regards to the performance of their internal organization (architecture) when handling different number of XML documents.

In this context, the Open Source NXD architectural metric assessment comparison basically refers to the response time or the time taken by each of these Open Source NXD products for executing a database task (i.e. storage).The response time was obtained by inserting check points within the code of each of these Open Source NXD products. Steven et al. [11] defined a check point as a procedural invocation inserted into the database system's flow control in order to call the performance measurement routines for collection of timing data.

The datasets used in the Open Source NXD architectural metric assessment comparison experiment consisted of different number of generated XML documents set containing diverse information about faculty. Storage operation (i.e. bulk storage) is the only database operation that was considered during metric assessment comparison experiment. Furthermore, the experimental evaluation was carried out on a single machine environment. The following were the characteristics of the machine used: Windows Operating System, 40 GB hard drive space, 512 RAM, and 1.80 GHz of CPU. The principal reason of choosing to carry out the experimental evaluation in such environment was to cut off other issues (e.g. network overhead issue, transformations of the output issue, etc) that might affect the final timing results.
The next section presents and discusses the experimental findings.

VI. EXPERIMENTAL FINDINGS

The graph shown in Figure 4 is a representation of the metric performance results of the three studied Open Source NXD products (i.e. Berkeley XML DB, eXist, and dbXML) with regards to their capabilities to manage different number of XML documents into the database collections.
Referring to Figure 4, numbers display on the horizontal-axis represents the number of XML documents that are stored inside that particular database collection. The numbers on vertical-axis are various times in milliseconds taken by each of the three Open Source NXD products when storing different number of XML documents into the database collections.

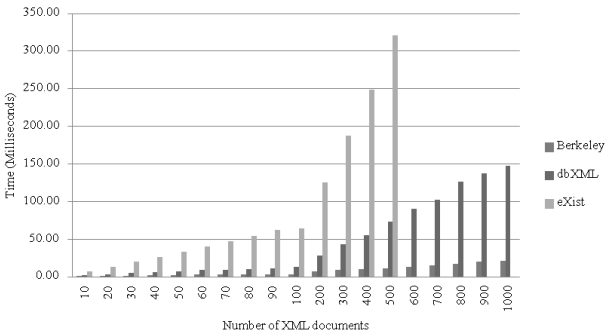

Figure 4. Open Source NXDs metric performance comparison

The results in Figure 4 show that the three Open Source NXD products were able to store XML documents. Additional to this, the comparison graph shows that as the number of XML documents were increasing (i.e. 10 – 1000 XML documents), the metric performance results number also started to increase as well. In another words, as the number of XML documents were increased, Open Source NXD products were also taking longer to store them into the database collection. Furthermore, there was a dissimilar type of variation amongst Open Source NXD products in terms of their metric results and performances. For example, the displayed results in Figure 4 reveals that eXist took very long to store XML documents (i.e. 10-500 XML documents) into the database collection. Also as the number of XML documents were increasing (i.e. 600-1000 XML documents), eXist could no longer store the XML documents. Basically eXist run out of memory. dbXML metric performance results as well revealed some kind of variations. Although, dbXML managed to store all XML documents (i.e. 10-1000 XML documents), its metric performance results almost double when storing from 200 to 1000 XML documents. Lastly, Berkeley XML DB metric performance results showed little variation in metric performance results than the two others Open Source NXD products namely eXist, and dbXML. In other words, Berkeley XML DB metric performance results were much better than dbXML and eXist.

The comparison results graph in Figure 4 show that Berkeley DBXML is the fastest because it performs a lot better than the two other Open source NXDs products (i.e. dbXML, and eXist). It was followed by dbXML which performed better than eXist. eXist was the slowest because its metric performance results was worse compared to Berkeley DBXML and eXist.

There are number of reasons which can be used to explain the different results obtained in the metric performance tests. Berkeley DBXML's better performance can be attributed to the fact that it implements text-based NXD architecture.
As mentioned in section III, the general characteristic of the text-based approach is it stores an XML document is stored

"as-is" as a whole without any conversion. This might the reason that gives Berkeley more advantage compared to the other two model-based Open Source NXD products.

eXist, dbXML, are examples of the model-based architecture NXD implementation. Contrary to Berkeley DBXML which implements the text-based approach, these two build the internal object model of the contents of XML document prior to storing it and after that store that model into the database. The physical representations of that model vary from one database to the other (i.e. it is database dependent). For example eXist stores the internal object model structure as persistent Document Object Model (DOM) objects in relational storage format whereas dbXML uses a proprietary storage format to store it. The main disadvantage of model-based NXD architecture compares to text-based is that when the number or size of XML document increases model-based requires more time and memory for building the internal object model of the contents of XML document prior to storing it.

The performance between Open Source NXD products that use model-based NXD architecture can be different as well. As shown, in the comparison graph in Figure 4 there is performance dissimilarity between dbXML, and eXist though both implement the model-based architecture model. This performance dissimilarity can be attributed to the kind of techniques used by each of these products for building an internal structure model of the contents of an XML document before storing it. Additionally to that, this can as well be attributed to the type of storage format model used by each of these products for storing that built internal structure model. For example, eXist uses relational storage model to store the internal structure model whereas dbXML, use a proprietary storage format. Referring to the experimental results obtained, it is concluded that the storage operation is fastest with Berkeley XML DB and slowest with eXist.
Section 6 presents the conclusion.

VII. CONCLUSIONS

This paper presented a comparative study of the two main NXD architecture models namely text-based, and model-based. Three examples of Open Source NXD products that implemented either of these two NXD architectures were considered. These include Berkeley DB XML that implements text-based NXD architecture and others two eXist and dbXML which implement model-based NXD architecture. All of the mentioned Open Source NXD products were deployed and compared with regards to their efficiencies in processing different number of XML documents.

The experimental metric results obtained from the experiments reveal that product that implements text-based architecture (i.e. Bekerley XML) performed better than the others products that implement the model-based architecture when storing different number of XML documents into the database collection.

Additionally, the experimental metric results had shown that there was some kind of metric performance variations amongst the products that implemented the model-based architecture. These performance variations were due to different internal organization techniques used by each of these model-based Open Source NXD products. Lastly, based on the experimental findings, the following lessons have been learnt: the size of XML documents has an effect on the performance of each of these NXD architectures. Therefore, before choosing a suitable NXD architecture issues such as the kind of datasets, including the type of database operations that will be performed on those datasets need to be taken into consideration. This is essential because if a wrong choice is made, an inappropriate NXD architecture will lead to degradation in database performance.

ACKNOWLEDGMENT

The author would like to thank the Open Source Projects (dbXML Group, Sleepcat Software, and Wolfgang) for making their products freely available. Thanks to Mr. GM Muriithi for taking time to proof read this paper. It was much appreciated. My thanks also go to Prof. J Chadwick who had played a vital role in my growth as an academic researcher. Lastly but not least, a hearty thank to my wife Mrs. M K Mabanza for her undying support and encouragement.

REFERENCES

[1] (2012) W3C website. [Online]. Available: http://www.w3.org/XML/. "XML"

[2] Akmal B. Chaudhri, Awais rashid, and Roberto Zicari. *XML Data Management: Native XML and XML-Enabled Database Systems*. 2003. *Addison-Wesley*, Page xxi.

[3] (2005) Wingenious. *Database Architecture*. [Online]. Available: http:// www.wingenious.com/database.pdf.

[4] (2005) Ronald Bourret. [Online]. Available: http://www.rpbourret.com/xml/XMLAndDatabases.htm. "XML and Databases."

[5] (2011) Berkeley DBXML. [Online]. Available: http://freecode.com/projects/dbxml

[6] (2012) dbXML . [Online]. Available: http://www.dbxml.com/.

[7] (2011) eXist . [Online]. Available: http://exist.sourceforge.net/.

[8] (2012) EBH Research Projet—X007 Website. [Online]. Available: http://www.comp.nus.edu.sg/~ebh/XOO7.html . "The X007 Benchmark".

[9] (2012) MonetDB Website. [Online]. Available: http://monetdb.cwi.nl/xml ."Xmark"

[10] (2012) Database Group Leipzig Website. [On-line]. Available: http://dbs.uni-leipzig.de/en/projekte/XML/XmlBenchmarking.html "XMach-1".

[11] Steven A. Demurjian, David K. Hsiao, Douglas S. Kerr, Robert C. Tekampe, Robert J. Watson. 1985. "Performance Measurement Methodologies for Database Systems". In proceedings of the 1985 ACM Annual Conference on The range of computing: mid-80's persepective, Colorado, United States, 1985, Pages: 16-28.

Ntima Mabanza is a Ph.D. fellow in the Department of Computer Science and Informatics at the University of the Free State (UFS) in South Africa, researching on educational agent technologies. He obtained his Masters' degree in Computer Science from the University of Fort Hare (UFH) in South Africa. Currently, he is a member of the Information Technology lecturing staff at the Central University of Technology Free State (CUT). His research interests include Educational agents, Open Source XML Databases, and E-commerce & M-commerce security protocols.