Going Native: Native XML Database vs. Relational Database

Raghad Yaseen Lazim[a], Adzhar Bin Kamaludin[b]

[a,b] *Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang*
Raghad.altaai@yahoo.com, adzhar@ump.edu.my

Abstract

Database is a set of logical data elements that related with each other by a mathematical relationship. The success of any database depend on some important things such: Storage procedure - the speed and ease of query - reduce non-useful information - reduce repetition - fixed records - the organization and strategy and so on.Relational databases have been improved for long time to handle such things very high efficiency. However, despite the maturity of relational database products and the Significant growth in computer power over the past years, the projects still fail because of the performance of the relational database used is just not good enough. Anyhow, always there is another way. Mostly it can be faster to store the data permanently and natively in XML to avoid the repeated conversion from relational format to XML format in the application. Because Native XML Database (NXD) provides a powerful environment for the development of web applications based on XQuery and related standards. This paper discusses the need for going native, and the difference between Native XML database and relational database to achieve high level of the performance.

Keywords: XML, Native XML Database, Relational Database.

1. Introduction

A database is an organized collection of data in digital form, that can be accessed immediately and manipulated by a data-processing system for a specific purpose. Mostly the data are organized in a model that relevant aspects of reality, in a way that supports processes requiring this information.

The database system is manages the data to some level of quality (measured in terms of accuracy, availability, usability, and resilience) and this in turn often implies the use of a general-purpose database management system (DBMS) [1].

DBMS is typically a complex software system that meets many usage requirements to properly maintain its data which are often large and complex [2].

XML is established as the preferred method for exchanging structured data between enterprises, but still there is layer of complexity between XML documents and the database.

There are two different ways to store XML documents in a database. The first way is to map the document's schema to a database schema, and transform XML documents into a relational database (mean decomposed into relational columns, and transfer data depending on the mapping), that can then be queried by using SQL. Such relational databases are called XML-enabled databases. The second way, we can store XML documents by using a fixed set of structures that can store any XML document, and it's the best choice for storing, and this method called native XML databases (NXD) [3].

2. Traditional Relational Database (RDB)

Relational database is a collection of tables designed specifically for storing the data. When storing XML data in a relational database, we partition the XML document in small parts, mean XML needs to be shredded to relational tables (By using the XMLTABLE function that is part of the SQL standard) rather than stored natively in an XML column, and this process is called shredding. shredding process mean convert the data from a document in the database and then reconstructing the document from that data - often results in a different document [4,5].

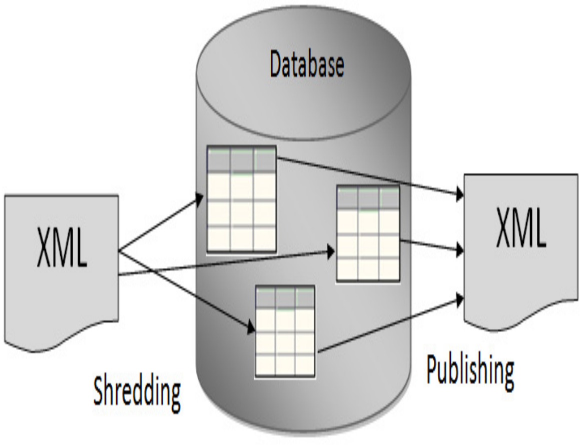

Fig (1) Shredding XML document to relational tables.

Regular we can store both unstructured data and highly structured data in relational structures. Both of data structures do not change oftentimes. But a inability of relational databases is in storing documents with a semi-structured data (Can have many degrees of freedom in the order of the elements of the document and the way in which these elements are nested within each other) you are likely to end up with a specific data structure that changes frequently, a generalized data structure that loses descriptiveness of the labels or with an abstract model such as those that content management systems use [6].

For storing these parts of XML document, there is a need for the mapping of the XML data to the relational tables, it is important to map the XML document schema (XML Schemas, DTD, etc.) to the database schema [7]. That's mean translate the XML-document schema into the relational schema before accessing the corresponding tables [8]. The need for these mappings can lead to some problems, when the mapping is based on the XML schema or DTD of the document. For every type of XML schema or DTD a mapping is needed. furthermore, the documents that are without exists maps cannot be used before a mapping is created. Any changes in the XML schema that will lead to changes in the mapping as well [9]. Since XML data are stored in relational tables, translated SQL may include retrieval and combine operations, which may expend great CPU power and cause performance degradation [9,10].

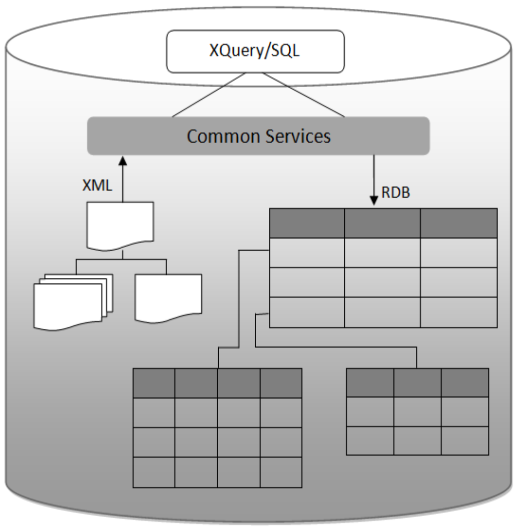

Fig (2) Relational database architecture

3. Native XML database (NXD)

Native XML database are database of specially designed to store XML documents, by using a set of fixed structures, Native XML Database can store any XML document (unstructured data and semi structured data), and updating and retrieving document-centric XML documents is usually a native XML database (NXD). NXDs store XML documents as logical units, and retrieve data by using specific query languages such as XPath or XQuery [11].

Native XML databases like other databases, they support features like transactions, security, multi-user access, programmatic APIs, query languages, and so on. The difference between NXD and databases is that, the model of NXD is based on XML and not something else, such relational model. NXD are most commonly used to store document-centric documents. Because of Native XML database support of XML query languages, which allow you to ask questions like, "give me all documents in the third paragraph after the start of the section contains a bold word," or for just to limit full-text searches to specific portions of a document. Like these queries are surely not easy task to ask in a language like SQL. In addition maintain things like document order, processing instructions, and comments, and often maintain CDATA sections, entity usage, and so on, while XML-enabled databases do not [12].

Native XML database's performance stems from its unique approach to managing documents. In contrast to most XML-enabled databases that decompose documents and store them in a relational database or object-oriented database, NXD will not change the original XML file. Documents are stored in their native, unmodified format.

When a document is inserted into the NXD, parse and indexes are built according to the structure of the document. As a result, there is no need of mapping for xml , that's mean there is no specific DTDs or schemas while storing documents in the database - enabling the storage of all document structures in a same place (single environment).

Native XML database's search engine enables fast retrieval of content within any element or feature of XML documents. Thanks to NXD advanced query language, search operators and advanced indexing technology, users benefit from virtually unlimited searching possibilities. NXD delivers the ability to search metadata and the full-text of documents by allowing users to query the data using multiple indexes [13].

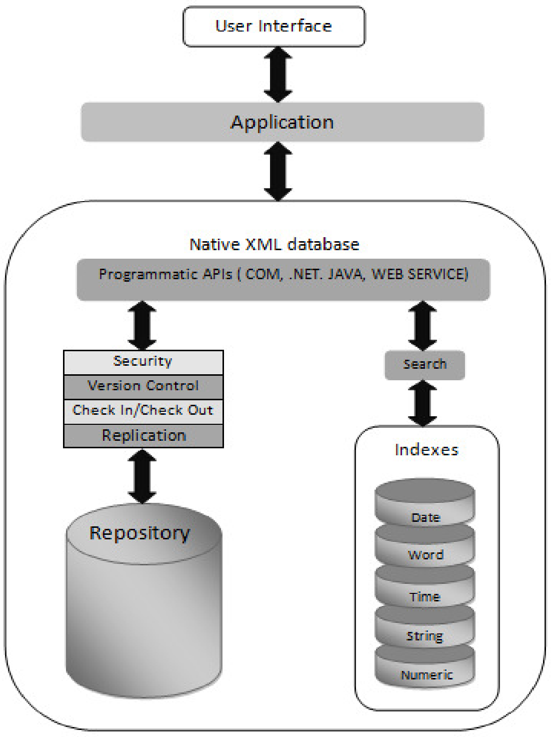

Fig (3) Native XML database architecture

The native XML database stores XML data directly, that's mean the database engine accesses the XML data without any conversion process. This makes great difference between an XML-enabled database and a native XML database. The direct access of a native XML database can reduce processing time and provide better performance. NXD implements performance-enhancing technologies, reduces the workload related to database administration, and it's able to store and query millions of XML documents. In addition, native XML database supports all relevant W3C standards including XML, XPath, Namespaces, DTD/schema, and Unicode [14].

Native XML databases differ from XML-enabled databases in three main ways:

1- Reduced complexity, because shredding any but simple XML documents is a complicated task. Native XML databases can maintain material structure (such as CDATA sections) as well as comments, PIs, DTDs, etc. While XML-enabled databases can do this in theory, this is generally not done in practice.

2- Increased flexibility, because native XML databases can store XML documents without knowing or using schema (DTD), presumably one even exists. XML-enabled databases could generate schemas, but this is impractical in practice, especially when transaction with schema-less documents.

3- Native XML databases are accessed through XML-based technologies, such as XQuery, XPath, or the DOM, and use XML-specific APIs. XML-enabled databases access data through non-XML technologies, such as SQL and Open Database Connectivity (ODBC).
4- Improved performance, because of the avoid of conversion between two different data representations, and guaranteed document accuracy, because the original XML business record stays in its XML format [15,16].
5- Native XML databases are used in many companies and in a wide number of fields, such as: Genetics, Health care, Insurance, and in the technologies like: Data integration, Messaging, Web sites, and so on.

We will give three examples for the usage of NXD, quoting sources used:

1- Since some years, the NoSQL (commonly interpreted as "not only SQL") movement has developed a variety of open-source document stores. Most of them focus on high availability, horizontal scalability, and are designed to run on commodity hardware. Because of most of NoSQL solutions are strong for modelling and processing hierarchical or graph data . NoSQL devotes a separate section to hierarchical data modelling. These products have obtained great traction in the industry to store great number of flexible data (mostly JSON (JavaScript Object Notation)). In the meanwhile, XQuery has evolved to a standardized, XML natively support for complex queries, indexed, compressed files, updates, full-text search, and scripting. Moreover, JSON has recently been added as a first-level data type into the language. As of today, it is without doubt the most robust and productive technology to process flexible data [17].

2- A Native XML Database in Health care**:**

Health-care institutions are gaining an increasing interest in exploiting the data that are gathered through electronic medical records. Current electronic medical records do not have the ability to store and retrieve data of this complexity in a suitable way.

Has been proposed an XML database for the storage of clinical documents to meet the document research needs outlined. This database Have been implemented support the research needs at Columbia University and the University of Connecticut (UConn) [18].

BJC HealthCare is one of the largest non profit health care organizations in the United States, they run 13 hospitals, multiple community health facilities, and they have more than 26,000 employees and a net revenue of $3.2 billion. Given the diverse and evolving nature of medical data, BJC decided to use XML as the format for patient medical records, lab results, and other clinical information.

This decision provides two benefits to their application. First, NXD provides the flexibility with the variable data and future schema evolution. in addition, using NXD has allowed BJC to design a simple and intuitive database schema with less than 10 tables. Storing the same information in a fully relational database schema would have required over 100 tables and many queries would have to join 20 or more tables, which is complex and can often be inefficient [19].

The use of NXD in clinical institutions for the following reasons:

- The adoption of the NXD in health-care institutions gives us speed, because of the ability of the fast process queries against text (key word searches), and the ability for the fast process queries against annotations or ‘mark up’ added to the text.
- A standard method for querying the documents, because of NXD is a standard means of query.
- The ability to select documents along many different axes of interest (within or across patients, over time, etc), because of NXD includes the join operations necessary, by using Clinical Document Architecture. The standard is defined in XML, making the database forward compatible. It includes standardized definitions for data such as patient and provider identifiers, demographics, and document type, which are required for the selections [18,19].

3- **Business records with NXD:**

The release of native XML storage was one of the big topics at IBM’s Information on Demand conference in Las Vegas and at the IDUG Europe conference in Vienna in December 2010. Introduced pureXML with native XML storage, XML indexing, XML Schema support, XML queries, and XML support in various utilities such as load, unload, and others.

Many leading companies in banking, insurance, telecommunications, manufacturing, logistics, and other industries still rely on the mainframe’s unsurpassed reliability and performance for transaction processing. And the wide-spread adoption of XML does not exclude the mainframe world. DB2 10 is the brand-new version of the DB2 database for mainframe computers running the z/OS operating system.

The results are shown in the figure. On average the XML solution achieved 55% higher throughput then the normalized relational schema. In the relational test, the CPU capacity of the system under test was exhausted with 84 concurrent users performing about 3800 transactions per second. With the NXD solution, the same system supported up to 150 concurrent users and almost 6000 transactions per second. At the same time the XML solution showed drastically lower CPU utilization on the client machines than the relational solution [20].

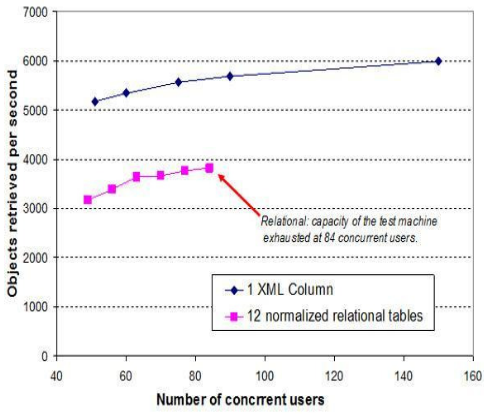

Fig (4) Retrieval performance: NXD vs. Relational DB- IBM

4. Conclusion and suggestions

This paper has looked at how native XML databases are used in the real world, most commonly for managing documents, integrating data, and managing semi-structured data, because of the flexibility of the Native XML database model, and their ability to handle schemaless data.

Although the migration from the XML-enabled databases to Native XML database requires some considerations, it is not very hard. The migration to Native XML database yields dramatic performance benefits and often simplifies your solution. You will need to migrate eventually, and the sooner you do it the earlier you will reap the benefits of significantly better performance and greater flexibility.

So is a native XML database in your future? That question is best answered by quoting Arun Gaikwad. In an article about Xindice, a native XML database from Apache, he wrote: "A [native XML database] is something which you may think is unnecessary but once you start using it, you wonder how you would survive without it."

We suggest this work in the future:

1- Develop and evaluate an XML retrieval system that uses a Native XML database to XML retrieval, to achieve the possibility of get a different answer lists.

2- Develop a retrieval module to achieve the optimal combination of retrieval data and matching elements in the final answer list in Native XML database.

References

[1]http://en.wikipedia.org/wiki/Database

[2] Jeffrey. U. and Jennifer. W, 1997, First course in database systems, Prentice-Hall Inc., Simon & Schuster, Page 1, ISBN 0-13-861337-0.

[3] Bourret. R. 2005. Going native: Use cases for native XML databases. http://www.rpbourret.com/xml/UseCases.htm.

[4] Dodds. L. 2001. XML and Databases? Follow Your Nose. http://www.xml.com/pub/a/2001/10/24/follow-yr-nose.html.

[5] Akmal B.Chaudhri, Awais Rashid, Roberto Zicari. 2003. XML Data Management: Native XML and XML-Enabled Database Systems. Addison Wesley. March 14, 2003. pp: 688.

[6] De Jonge. A. 2008. Comparing XML database approaches. IBM Corporation. 16 Dec 2008.

[7] Papamarkos. G.Zamboulis. L. Poulovassilis. A. 2005. XML Databases. http://www.dcs.bbk.ac.uk/~sven/adm08/xmlDBs.pdf

[8] M. A. Gonçalves, M. Luo, R. Shen, M. F. Ali, E. A. Fox: An XML Log Standard and Tool for Digital Library Logging Analysis, ECDL 2002, 129-143, Rome, Italy.

[9] Henk, J. 2006, Native XML databases, 5th Twente Student Conference on IT , Enschede 26th June, 2006.

[10] Gerritsen. J. Native XML databases. referaat.cs.utwente.nl/TSConIT/download.php?id=102.

[11] R. Bourret, Going Native: Making the Case for XML Databases, http://www.xml.com/pub/a/2005/03/30/native.html.(2005).

[12] Ufimtsev. R. 2005.XML and Databases. Ronald Bourret. September 2005.

[13]http://www.ixiasoft.com/default.asp?xml=/xmldocs/webpages/textml- server.xml.

[14] L,forms, 2009, Build an intelligent eForms solution based on DB2 pureXML, 1-27.

[15] Pavlovic', G, 2006, Native XML Database vs. Relational Database in dealing with XML documents , Kragujevac J. Math. 30 (2007) 181-199.

[16]http://www.rpbourret.com/xml/ProdsNative.htm.

[17] Fuller, J. 2012. BigData and Modern XML. MarkLogic (Gold Sponsor).http://www.xmlamsterdam.com/2012/sessions.

[18] Johnson, B . Campbell, A. Krauthammer, M. and other. 2003. A Native XML Database Design for Clinical Document Research.

[19] Nicola, M. 2010. BJC HealthCare Improves Clinical Research with DB2 pureXML. February 7, 2010.

[20] Malaika, S .Nicola, M. 2012Data normalization reconsidered, Part 2: Business records in the 21st century. IBM. 12 January 2012.