IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5168

This paper appears in *Managing Modern Organizations Through Information Technology*, Proceedings of the 2005 Information Resources Management Association International Conference, edited by Mehdi Khosrow-Pour.

A Benchmark Comparison Between Native XML and Relational Databases

Alexander van der Linden
University of Liverpool, Laureate Online Education BV, Arena Blvd. 61-75, 1011 DL Amsterdam ZO, The Netherlands,
s.van.der.linden@yourel.nl

Paul Darbyshire
School of Information Systems, Victoria University, PO Box 14428, Melbourne City MC, Victoria 8001, Australia,
paul.darbyshire@vu.edu.au

ABSTRACT

The growth of the Internet and of XML has dramatically affected the way information is exchanged and combined. An indicator of the extent is that every major database vendor has added XML support to their product. XML is changing the way databases are used, often in ways that database vendors would never have anticipated just a few years ago. Most commercial relational, object-relational, and object-oriented database systems offer extensions and other mechanisms to support the management of XML data. In addition to supporting XML within existing database management systems, we have seen the emergence of native XML databases. These are designed for seamless storage, retrieval, and manipulation of XML data and integration with related technologies, and have been proposed as a solution to the complex storage of XML structured data. While some researchers claim that native XML databases will play a major role in the database world, others claim that as soon as SQL databases have developed XML capabilities, native XML databases will cease to exist due to poor performance. This paper presents the results from a comparative investigation into the performance of a relational database against an existing native XML database. Benchmark tests are based on the XMark1 benchmark, are used to compare the relative performance.

INTRODUCTION

As the use of XML has grown, it is now generally accepted that XML is not only useful for describing new document formats for the Web but is also suitable for describing structured data. Examples of structured data include information that is typically contained in spreadsheets, program configuration files, and network protocols. XML is preferable to previous data formats because XML can easily represent both tabular data (such as relational data from a database or spreadsheets) and semi-structured data (such as a Web page or business document) (Obasanjo, 2003). Popular pre-existing formats such as comma separated value (CSV) files either work well for tabular data and handle semi-structured data poorly, or like RTF are too specialized for semi-structured text documents. This has led to the widespread adoption of XML as the lingua franca of information interchange.

As more and more organisations and systems employ XML within their information management and exchange strategies, classical data management issues pertaining to XML's efficient and effective storage, retrieval, querying, indexing and manipulation arise. From this environment we have seen the emergence of native XML databases. These are designed for seamless storage, retrieval, and manipulation of XML data and integration with related technologies (Noordij, 2002). However, a number of questions arise regarding Native XML Database (NXD) technology. Does it represent a paradigm shift? More importantly, is the performance of NXD technology sufficient to provide an alternative to standard database technology, or will coexistence be the status quo?

This paper reports on research conducted to test the relative performance of comparable relational and native XML databases using benchmarked time-trials. In the following sections, some background information is given on native XML databases, their characteristics and benchmarking,. The selection of a suitable relational database and native XML database is then detailed, followed by a discussion on the construction of the benchmark tests and then the presentation of the benchmarking results. Finally, details of further research and some conclusions are presented.

BACKGROUND

XML is rapidly becoming a standard in interchanging data via electronic means and as the use of this file format is growing, there is a demand for a structured approach when looking at data storage. As the XML file format and its use differs somewhat from the standard relational database approach, Native XML Databases (NXD's) have been developed in contrast. However there are many criticisms levelled against them, chiefly their suitability for data storage and their performance in terms of storage and retrieval.

Native XML Databases

Currently there is no formal, standard definition of an XML database (Cox, 2001), however the XML:DB Initiative (www.xmldb.org) describes such a database as one that defines a logical model for an XML document (not for the data in the document), and manages documents based on that model. We can also consider the document itself as a database (Udell, 2001). An XML document as a perfect container for structured data, nodes and siblings acting as the counterpart of relational database rows and columns.

Can an NXD be described as a database in the sense of a logical model of structured data? Bourret describes it on a document level: "an XML document is a database only in the strictest sense of the term; that is, a collection of data" (Bourret, 2002). A more useful question to ask is whether XML and its surrounding technologies constitute a "database" in the looser sense of the term, that is, a database management system (DBMS). XML and its related technologies provide many of the things found in databases: storage (XML documents); schemas (DTD's, XML schemas); query languages (XQuery, XPath, XQL); programming interfaces (SAX, DOM, JDOM) (Staken, 2001).

A native XML database should define a logical model for an XML document and have the document as the fundamental unit of storage (Staken, 2001). Most NXD's aren't really standalone databases at all as they don't store the XML in true native textual format. But using a relational DBMS to store XML documents can create serious performance problems for large scale applications. Data hierarchy, context

and semantics are often lost when XML documents are retrieved and processed with SQL (Liotta, 2003).

Benchmarking

Basically, a benchmark is used to test the peak performance of a system. Different aspects of a system have varying importance in different domains. Hence it is important that a benchmark captures the characteristics of the system measured. To be useful, a domain-specific benchmark must meet four important criteria (Gray & Catell, 1993). It must be able to measure the peak performance of systems when performing typical operations within that problem domain; it should be easy to implement the benchmark on many different systems and architectures; it should be scalable; it should be understandable, otherwise it will lack credibility. There are many domain-specific benchmarks. And each of these is suitable for benchmarking elements of their own domain.

SQL domain benchmarks include:

- The Wisconsin benchmark is widely used to test the performance of relational query systems on simple relational operators (Bitton, DeWitt, & Turbyfill, Nov 1983).
- The AS3AP benchmark provides a more complete evaluation of relational database systems by incorporating features such as testing utility functions, mix batch and interactive queries (Turbyfill, Sep. 1987).
- The Set Query benchmark is used for testing the ability of systems to process very complex queries and is representative in data-mining environments (O'Neil, 1993).
- TPC-D is used for on-line transaction processing (OLTP), and the most recent TPC-W measures server-based transactions in E-commerce environments (Anonymous, 1996).

The need for standard measures for XML data-processing environments has led to the development of XML-specific benchmarks. These include:

- The XOO7 benchmark is an XML version of the OO7 Benchmark with new elements and queries added to test the features that are unique in XML. The XOO7 benchmark tests focus on the data-centric query capabilities of object-oriented database systems (Bressan, Lee, Li, Lacroix, & Nambiar, Nov 2001). However this benchmark relies on an intricate Entity Relationship Diagram and XML does not support such relationships.
- The XMach-1 Benchmark is a multi-user benchmark designed for business-to-business applications and Web applications using XML as follows (Böhme & Rahm, 2001a). This benchmark limits the XML data to a simple structure. It supports both schema-based and schema-less XMS and allows implementing some functionality at application level.
- The XMark Benchmark consists of an application scenario that models an Internet auction site and 20 XQuery challenges designed to cover the essentials of XML query processing. The XMark Benchmark is a perfect benchmark to test a complete system, with front and back end, web-based (Schmidt et al., 2002).

The SQL domain benchmarks however, are not suitable for evaluating XML storage in SQL systems. They lack the support of XML-specific details, such as tree-traversing, parent-node navigation, structure preservation, decomposing and creation of XML documents. Comparing two fundamentally different paradigms such as NXD and relational databases requires a non-standard approach.

SELECTING SUITABLE TEST DATABASES

In order to study the behaviour of relational databases in comparison to XML data, we have to select two databases. Given the constraints of the testing hardware, these two databases needed to comply with the following rules:

- Represent a functionally different technical paradigm(NXD and Relational);
- Operate on an Intel/Windows platform (XP Professional)
- Experiencing an active lifecycle;
- Able to run stand-alone (no other software necessary).

The short lists of databases considered for selection is as follows:

Native XML Database Systems

- IPEDO by Ipedo (Commercial);
- Tamino by Software AG (Commercial);
- Exist by Exist Software (Open Source);
- Xindice (Open Source);
- X-Hive by X-Hive Corporation (Commercial).

XML Enabled Database Systems

- DB2 in combination with XML Extender by IBM (commercial);
- SQL Server 2000 by Microsoft (commercial)
- Oracle 9i by Oracle (commercial);
- Access 2003 by Microsoft (commercial, part of the Office Professional Suite).

Ipedo and Tamino are both sophisticated products, developed to act in co-operation with other products to deliver professional XML support on high volume transaction platforms. However, any attempt to run these database management systems on a single PC will not lead to a stable environment. Therefore, these databases do not comply with the rules we have defined. eXist and Xindice are both products of the Open Source community. Both products can handle the job well, though ease of use and installation could be enhanced but an administrator utility is not supported, which makes them unfit for the benchmark trials. X-Hive is a professional product, with some very pleasant features and the administrator utility supports XQuery queries and the response time is measured. The X-Hive native XML database was chosen as subject for the benchmark.

DB2 is a standard on many platforms and the new release Personal Edition has the XML Extender, which enables XML support. However, the installation of XML Extender unearthed some problems and a running installation could not be established. Microsoft SQL server is a standard on the Windows platform. Its support for XML looks very promising, but unfortunately, SQL server can only be run form Microsoft Server (NT/2000/2003). Microsoft Access is a standard on the PC/Windows platform. As a single tier or multi-tier database (via ODBC) the product has a long history of good performance. The 2003 edition has some interesting features, concerning import and translation of XML files and schemas. Microsoft Access fully complies with our rules and was chosen as the subject for the benchmark

CONSTRUCTING A BENCHMARK

Benchmarking a native XML database and benchmarking a relational database can't be done with exactly the same benchmark, because of the fundamental differences between the two platforms. A benchmark that has the capabilities to evaluate the performance of both type of systems can only be described on a meta-level. That is, the implementation of the queries will be are different on every system. A relational database uses relational tables and SQL as query language; a native XML database uses collections and XML files as a single unit of storage and XQuery or XPath as the query language.

We had to develop our own benchmark that would be relevant to XML and Relational databases and some compromises had to be made. The translation of part of the queries from XQuery to SQL had to me manually performed. Additionally, Microsoft Access cannot automatically translate the results of a query into XML, therefore only the execution time of a query was measured. As a result of this, some queries (producing structured XML data) had to be skipped on the SQL platform. This is clearly indicated in the results.

The developed benchmark trials are based on the X-Mach1 Benchmark by Böhme and Rahm (Böhme & Rahm, 2001b). This is a multi-user

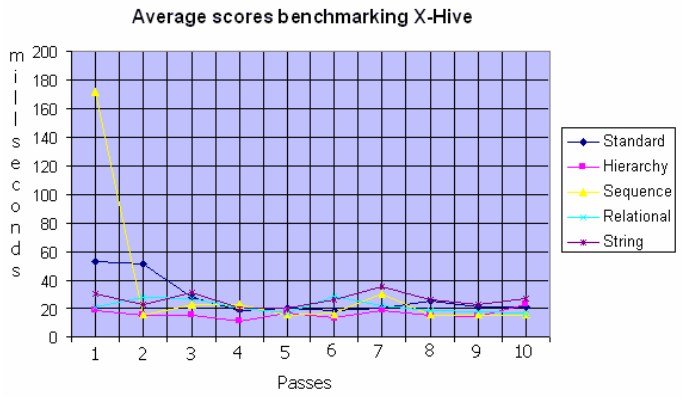
benchmark designed for B2B applications and the XML Query Use Cases (Chamberlin, Fankhauser, Florescu, Marchiori, & Robie, 2003). However, we did not focus on the multi-user component but concentrated on the special properties of the X-Mach1 benchmark for a single-user issuing queries form a local machine. The XMach-1 benchmark limits the XML data to be simple of data-structure and small in size. This suits perfectly well the limitations posed by the test platform. Additionally, the choice was made easier by the relatively easy implementation, the scalability and the fact that the queries were developed with the single user in mind.The XML use Case Queries are composed of 5 sets of queries, composed in XQuery format. These are:

- Standard queries: tests for the ability of the database to produce standard formatted output, based on a simple 'select for' clause. The query consists of 12 different queries.
- Hierarchical: this query tests the ability of the database to preserve the original hierarchical tree in the XML documents. The total test consists of 6 different queries.
- Sequential: although sequence is not significant in most traditional database systems or object systems, it can be quite significant in structured documents. The total test consists of 5 queries.
- Relational: one important use of XML will be the storage of relational data. This query describes a possible way in which this access might be accomplished. The target data consists of 3 linked tables. The total test consists of 18 queries.
- String: this query tests for simple string search in long structured documents, but it makes use of comparison of strings. The total test consists of 5 queries.

To test the behaviour of loading and parsing, a 6th test was added, which loads 2 different XML documents, with a 132 Kbyte and 1.32 Mbyte file size. The native XML database must read and parse the nodes of the document object model and display them. The relational database has to read the XML file and convert it into relational tables. The XML documents and DTD's are provided by the W3C Consortium. Every query set was run 10 times in order to achieve a good average and to omit start-up problems like pre-reading indices, buffering tables etc. X-Hive provides an administration utility that allows for XQueries to be run. After processing a query, the total processing time is displayed with the results.

The XML data which was used by the queries running on the X-Hive native XML database was imported into Access 2003 and relations between tables were created. The following step was to convert the XQuery commands into SQL. As the XQuery commands used in the benchmark were not very complicated this was not a major undertaking. To obtain the processing time of each query a small function was written in VBA (Visual Basic for Applications) which showed the processing time after the display of the query results. All the queries (for both native XML and relational database) were run in single user mode, on a 2.4 Mhz Pentium IV PC fitted with 512 Mb of RAM, a 120 Gb hard disk and Windows XP Home edition as operating system.

Figure 1. Average Scores for X-Hive Benchmark



RELATIVE COMPARISONS

In this section we have shown the results of the benchmark applied against the native XML database and against the relational database. A comparative chart is then used to highlight the differences.

Figure 1 shows the average scores calculated from all passes for all query types of the benchmark tests for the X-Hive database. This diagram is created by taking the absolute values of every query of the different types and converting them to averages for every pass. Every query type is represented by a different colour, as shown in the legend. As the different parts of the benchmark consist of an unequal amount of queries, producing a coherent diagram is not possible.

Obvious is the overall average scores which stay in a 50 milliseconds bandwidth. All scores stay below a 60 millisecond border, except for the Sequence, pass 1, which took 170 milliseconds. Even the Relational part, where one should expect some difficulties as real relationships do not exist in native XML databases, and various physical files must be accessed, the score showed no significant deviation.

Microsoft Access has 2 different modes supporting queries, in the SQL window, or with the help of the query builder. If the latter is chosen, the query is automatically translated into SQL. The Hierarchical query of the benchmark was not applicable to Access, as Access cannot produce XML output automatically and directly from within a query. To do so, a report has to be defined and exported to XML, which is technically feasible. This action has not been taken into account.

Figure 2 shows the average scores calculated from all passes for all query types (except hierarchical), of the benchmark tests for the MS Access database. All the queries run in Access stay below a 10 millisecond border, which is significantly faster then X-Hive. Access shows a more incoherent behaviour compared with X-Hive. The Relational query showed some higher values when a particular query was executed (inner join).

Figure 2. Average Scores for MS Access Benchmark

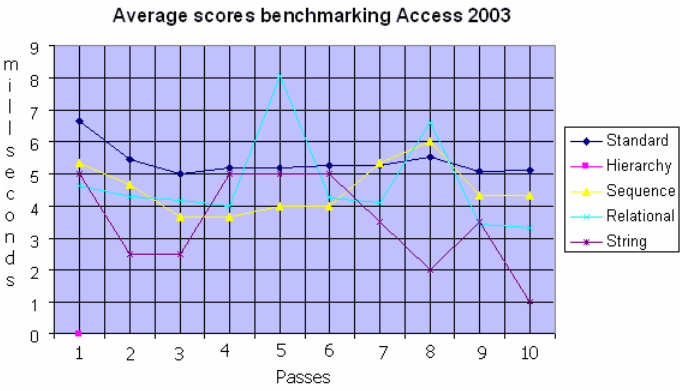
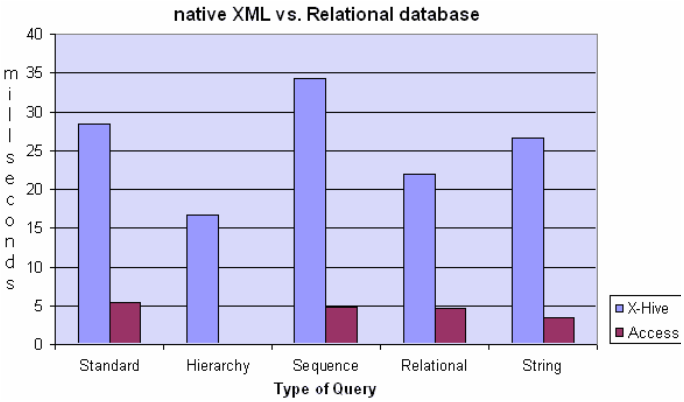


Table 1. Comparative Times for Benchmark of Native XML vs Relational Database

Query Type	X-Hive	Access 2003
Standard	28.48	5.37
Hierarchy	16.73	N/A
Sequence	34.35	4.85
Relational	22.02	4.69
Full string	26.63	3.50

Figure 3. Comparative Bar Graph of Overall Times for X-Hive vs Access



The XML data is read and converted automatically into tables. This imposed no problem as the XML data for the benchmark consists of single files. The import of relational XML data, where separate XML files are related, is disappointing. Access is able to read the relationships out of the W3C consortiums compatible schema and creates the different tables and keys, but does not provide the relationship itself. This needed human intervention. The relationships have been converted into 3rd normal form.

Table 1 shows averaged comparative times over all passes for the different query types of the native XML and relational databases. These are represented graphically in Figure 3.

CONCLUSION

Given the results of the benchmark trials presented in the previous section there is a significant difference between native XML and relational SQL. Access does run the equivalent of the XQuery queries much faster than X-Hive. This nis of course except for the Hierarchical queries. These could not be converted as SQL does not support the count of XML-elements and the converted XML data exists in tables not identifiable as XML data anymore. Access is on average about 6.17 times faster than X-Hive in in all query types.

We can further conclude that XML, converted (or scattered) into relational tables can be accessed much faster (about 7 times faster) than the native XML counterpart, provided that the data has not to be presented in its native format. However, that be a concern only if the relational database is part of a 'chain' in which XML is the native format such as in Web applications. In legacy systems, where XML is only of any importance as intermediate format (middleware), this should be not of any problem.

Currently is seems that the relational database is very well suited for handling XML data, provided that the XML contents/structure is mapped adequately to the applicable tables, rows and columns and that relationships between the tables are established in a proper way. If developers of relational database systems can further enhance the XML functionality, in a sense that handling XML data does not demand an in-depth knowledge of XML as architecture and/or the structure of the applicable XML data, relational SQL databases provide all the necessary functionality needed.

An interesting area for future research would be the applicability of Object Oriented Databases for storing XML documents. We have ignored then in this study, but they may well be better equipped to serve as prime storage for XML documents.

ACKNOWLEDGEENTS

This research project was undertaken as part of a Master of Science degree with University of Liverpool and Laureate Online Education.

REFERENCES

Anonymous. (1996). . *TPC-D Benchmark Specification, Version 1.2.* Transaction Processing Performance Council

Bitton, D., DeWitt, D., & Turbyfill, C. (Nov 1983). *Benchmarking Database Systems, a Systematic Approach.* Paper presented at the Ninth International Conf. on Very Large Data Bases,

Böhme, T., & Rahm, E. (2001a). *Multi-User Evaluation of XML Data Management Systems with XMach-1.* University of Leipzig, Germany.

Böhme, T., & Rahm, E. (2001b, 2001). *XMach-1: A Benchmark for XML Data Management.* Paper presented at the BTW 2001, Oldenburg.

Bourret, R. (2002). *XML and databases*, from http://www.rpbourret.com

Bressan, S., Lee, M. L., Li, Y. G., Lacroix, Z., & Nambiar, U. (Nov 2001). *The XOO7 XML Management System Benchmark* (No. Technical Report TR21/00): NUS CS Dept.

Chamberlin, D., Fankhauser, P., Florescu, D., Marchiori, M., & Robie, J. (2003). *XML Query Use Cases*. W3C Consortium, from http://www.w3.org/TR/xquery-use-cases

Cox, J. (2001). Working out the bugs in XML databases. *Network World Fusion, vol*. http://www.nwfusion.com/news/2002/0107specialfocus.html

Gray, J., & Catell, R. (1993). *The Benchmark Handbook*: Morgan Kaufmann Publishers Inc.

Liotta, M. (2003). *XML and Relational Databases*. Montara Software, Inc., from http://www.montarasoftware.com/go/de2de519-322f-1157-993a-b103537bde48

Noordij, M. (2002). The benefits of XML database. *Computable (Dutch version), vol*, 51.

Obasanjo, D. (2003). *Understanding XML*. Microsoft Developers Network, from http://msdn.microsoft.com/library/default.asp?url=/library/en/dnxml/html/UnderstXML.asp

O'Neil, P. (1993). *The Set Query Benchmark. The Benchmark Handbook For Database and Transaction Processing Systems.* (2nd ed.): Morgan Kaufmann.

Schmidt, A., Waas, F., Kersten, M., , C., M., , M., I., & Busse, R. (2002). *XMark: A Benchmark for XML Data Management.* Paper presented at the 28th VLDB Conference, Hong Kong (SAR CHina),

Staken, K. (2001). *Introduction to Native XML Databases*. O"Reilly & Assoc., from http://www.xml.com/pub/a/2001/10/31/nativexmldb.html

Turbyfill, C. (Sep. 1987). *Comparative Benchmarking of Relational Database Systems, Ph. D. Dissertation*. Cornell University

Udell, J. (2001). *The document is the database*. O'Reilly and Associates, from http://www.xml.com/pub/a/2003/07/09/udell.html