Comparison of Native XML Databases and Experimenting with INEX

Petr Kolář and Pavel Loupal

Dept. of Computer Science and Engineering
FEE, Czech Technical University
Karlovo náměstí 13, 121 35 Praha 2
Czech Republic
kolarp3@fel.cvut.cz, loupalp@fel.cvut.cz

Abstract. The aim of the article is to summarize and compare approaches of design and architecture of native XML databases. We discuss our results accomplished by utilizing the INEX data set in two open source database systems - eXist and Apache Xindice. There is also a basic performance comparison outlined as a basis for discussion about suitability for particular database system and for our consecutive experiments.

1 Introduction

XML documents can be stored in a native XML database. Storage of semi-structured data in a native XML database (NXD) has an advantage in the fact that it has a regular structure but the structure varies enough that what means that mapping this structure into a relational database results in either a large number of columns with null values which wastes spaces or a large number of tables which is inefficient. Another advantage of storing data in a native XML database is the retrieval speed. It is much faster to retrieve data from a native XML database than relational database.

The term *native XML database* is used in different ways by various groups. For our purposes we consider the XML:DB definition – but, to distinguish from XML-enabled databases, we require a native XML database to have the following two properties as well:

- The XML data model (either in the XML Infoset or the XQuery/XPath Data Model) is the fundamental logical data model both used internally by the database and exposed to database users when XML is the data type.
- The XML data model is the fundamental unit of physical storage of all XML data, without mapping to a different data model.

This narrowed definition means that XML is more than an externalized data type - it is how the data is handled both logically and physically. The data is represented as XML right down to its physical storage schema on the disk. This model is the best for efficient searching of the XML data.

V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2006, pp. 116–119, ISBN 80-248-1025-5.

2 Comparison of Exist and Xindice XML Native Database

Due to limited space we mention only basic attributes and features of two database systems in following table. In our work we consider Xindice XML database version 1.0 [1] and eXist XML database version 1.0-dev-20060124 [2]. We would like also test Timber or Sedna database, but we decided not to test these databases. Both Timber and Sedna database accept only load of one XML document into database container.

Feature	eXist	Xindice
Technology	Java	Java
Data storage	B+-trees and paged files. Persistent DOM	Natively as indexed text files, Hoffman codes
Binary files	No	No
Transaction Support	No	No
Authorization	Unix like, permissions at collection and document level	No Support
Supported Standards	XPath/XQuery, XUpdate, Xinclude/XPointer	XPath, XUpdate, AutoLinking
APIs	XML:DB	XML:DB, command line
Client GUI	Yes	No
Indices	Structural, Fulltext, Range	

3 Experiments, basic performance comparison

3.1 INEX Dataset

For our experiments we use the INEX XML data set. The INEX data set (we use version 1.4) has 536MB of XML data. It is exactly 12,107 articles from 6 IEEE transactions and 12 journals from years 1995 to 2002. Pictures are not included – data set consists only of XML formated text.

Data set is organized in a file structure. Root directory consists of two subdirectories – *dtd* (holds structure information - DTD specification article element) and *xml*. Each journal/transaction has its own two-letter named subdirectory inside xml directory. Journal/transaction is further divided into the directories by the year of publication. Finally each article is stored in an individual xml file, which name consists of a letter followed by four-digit number and xml suffix.

In average each article contains 1,532 XML nodes, where the average depth of node is 6.9. See [5] for detailed characteristics of data set.

3.2 XPath

XPath [3, 4] is a language for finding information in an XML document – navigating through elements and attributes in an XML document. XPath is a major

element in the W3C's XSLT standard - and XQuery and XPointer are both built on XPath expressions. So an understanding of XPath is fundamental to a lot of advanced XML usage.

We prepared set of XPath queries in following categories:

Selecting nodes. XPath uses path expressions to select nodes in an XML document – e.g. $/article$ or $/article/fm/hdr/hdr1/crt/issn$. Queries 1 to 3 in Table 1.

Predicates. Predicates are used to find a specific node or a node that contains a specific value. Predicates are always embedded in square bracket. E.g. $/article/bdy/sec[1]$ or $/article/bdy/sec[position() < 3]$. Queries 4 to 11 in Table 1.

Selecting Unknown Nodes. XPath wildcards can be used to select unknown XML elements – e.g. $/*/*[@*]$. Queries 12 to 14 in Table 1.

Selecting Several Paths. By using the | operator in an XPath expression we can select several paths – e.g. $//article/fm/hdr|//article/bdy/sec$. See queries 15 and 16 in Table 1.

4 Results

We measured duration time of each query five times. Then we discarded the largest and the smallest value and counted arithmetic mean.

The time needed to load INEX data set into database was 25 minutes for Xindice and 97 minutes for eXist. The data on filesystem took 600 MB for Xindice and 1300 MB for eXist. Our hardware configuration was based on a personal computer with Intel Celeron 1.7 Ghz processor, 512MB RAM and Windows XP(SP2) operating system. INEX XML data set in version 2003 (1.4). Detailed information about the data set and its structure is shown in Section 3.1.

4.1 Summary

Our results do not meet our expectations – Xindice has totally failed in our experiments. With regard to our results this database system is impracticable for more extensive XML data sets. Althought we tried to create indices for all elements and attributes but without any significant improvement.

Most of XPath queries running over Xindice returned an empty result set – it seems that Xindice does not fully support the XPath 1.0 specification but only its limited subset. On the contrary, eXist showed much better behavior. This can be induced by its automatically generated structural index that is very efficient. eXist has also an user friendly GUI for both database management and ad-hoc query processing.

No.	Query	Records retrieved	Query duration time [s] eXist	Xindice
1	$/article$	12104	1,3	230
2	$/article/fm/hdr/hdr1/crt/issn$	11666	2,2	98
3	$//issn$	11666	1,3	447
4	$/article/bdy/sec[1]$	11955	1,9	NA
5	$/article/bdy/sec[last()]$	11955	5,6	NA
6	$/article/bdy/sec[last()-1]$	11019	5,8	NA
7	$/article/bdy/sec[position() < 3]$	22974	8,1	NA
8	$//sec[@type]$	868	1,0	more than 10 min
9	$//sec/p/ref[@type =' bib']$	108496	81,3	NA
10	$/article/fm/hdr/hdr2/pdt[yr =' 1995']$	1623	2,6	NA
11	$/article/fm/hdr/hdr2/pdt[yr =' 1995' and mo =' Spring']$	72	4,0	NA
12	$/article/*$	58472	164,3	NA
13	$/*/*[@*]$	49	352,0	NA
14	$//fig[@*]$	52857	70,6	NA
15	$//article/fm/hdr \mid //article/bdy/sec$	77487	8,6	NA
16	$//article/fm/hdr/hdr1 \mid //article/fm/hdr/hdr2$	24208	3,8	NA

Fig. 1. Results of given queries

5 Conclusion

The aim of our experiment – to test some of native XML databases and perform basic performance comparison – was in principle not successful. We were not able to import the INEX data set into all proposed native XML databases. Therefore we carried out only basic tests for the eXist and Xindice databases. Our results show that for further experiments we should consider only the eXist database. Xindice can be used just as an example of a basic native XML database.

We would like to perform further comparisons among other native XML databases. Also, we plan to add some of non-native (or hybrid) XML databases.

References

1. Apache Xindice - Native XML database. http://xml.apache.org/xindice.
2. eXist Native XML database. http://exist.sourceforge.net/.
3. D. Chamberlin, A. Berglund, and e. a. Scott Boag. XML Path Language (XPath) 2.0, September 2005. http://www.w3.org/TR/xpath20/.
4. J. Clark and S. DeRose. XML Path Language (XPath) 1.0, November 1999. http://www.w3.org/TR/xpath.
5. Fuhr, N., Gvert, N., Kazai, G., Lalmas, M. Initiative for the evaluation of xml retrieval (INEX), 2003.