BENCHMARKING AN XML MEDIATOR

Florin DRAGAN, Georges GARDARIN

PRiSM Laboratory University of Versailles 78035 Versailles Cedex, France

Keywords: XML, mediation, benchmark

Abstract: In the recent years, XML has become the universal interchange format. Many investigations have been made on storing, querying and integrating XML with existing applications. Many XML-based commercial DBMSs have appeared lately. This paper reports on the analysis of an XML mediator federating several existing XML DBMSs. We measure their storage and querying capabilities directly through their Java API and indirectly through the XLive mediation tool. For this purpose we have created a simple benchmark consisting in a set of queries and a variable test database. The main scope is to reveal the weaknesses and the strengths of the implemented indexing and federating techniques. We analyze two commercial native XML DBMS and an open-source relational to XML mapping middleware. We first pass directly the queries to the DBMSs and second we go through the XLive XML mediator. Results suggest that text XML is not the best format to exchange data between a mediator and a wrapper, and also shows some possible improvements of XQuery support in mediation architectures.

1 INTRODUCTION

As XML capabilities have become more and more popular, a lot of XML-based products and interfaces have been proposed. Several XML DBMSs that have been developed try, on the one hand, to offer the well known capabilities of a standard DBMS, and on the other hand, to implement new functionalities and reach new levels of performance. In the same time more and more classical DBMSs add new extensions to store and retrieve XML documents.

For measuring and comparing their performances, a lot of XML benchmarks have been proposed that "stress" different parts of the systems, most often the storage engine and the query processor, by means of a generally complex set of queries. Each benchmark is composed of a test database and a set of queries trying to be as general and complete as possible. There are also a few benchmarks specific to a certain domain that propose a specific format of database and a set of queries specific to the simulated applications. The most used metric is the response time for executing a query, but a few other ones (like the size on disk to store a certain document) are also proposed.

The purpose of this paper is to present a simple general mini-benchmark composed of a few queries and a variable data set to evaluate some techniques implemented in the core of the DBMSs under the pressure of an XQuery mediator. We are mostly interested in the implemented indexing and mediation techniques and how they are influenced by the size of the data set. Using our mini-benchmark, we test two native XML commercial DBMSs and one open source XML to relational mapping middleware and analyze their response times. Next, we apply our benchmark to an XML mediator for finding the delays that are introduced by the mediation operations. The conclusions show that XML mediation is a time consuming operation that has to be optimized both in communication and processing time.

The rest of this paper is organized as follows. In the next section we give an overview of XML mediation technology focusing on the XLive full-XML mediator. Section 3 introduces our mini benchmark (query set and data set). In section 4, we present the results of the benchmarking operations using the mediator and XML DBMSs. In conclusion, we summarize our results and suggest some improvements to the mediator architecture.

2 XML MEDIATION

Mediation technology based on XML and XQuery is under development. Some products are already available. In this section, we survey this new technology and describe our XLive mediator (see www.xquark.org for an industrial open source version).

2.1 Basics and Backgrounds

With the advent of XQuery as a standard for querying XML collections (XQuery, 2003), several mediator systems have been developed using XQuery and XML schema as pivot language and model. Examples of full XML mediators are the Enosys XML Integration Platform (EXIP (Papakonstantinou, 2003),), the Software A.G. EntireX XML Mediator, the Liquid Data mediator of BEA derived from EXIP, the e-XMLMedia XML Mediator, a predecessor of our current XLive project (Gardarin, 2002).
XML Mediators are focused on supporting the XQuery query language on XML views of heterogeneous data sources. The data are integrated dynamically from multiple information sources. Queries are used as view definitions. During run-time, the application issues XML queries against the views. Queries and views are translated into some XML algebra and are combined into single algebra query plans. Sub-queries are sent to local wrappers that process them locally and return XML results. Finally, the global query processor evaluates the result, using appropriate integration and reconstruction algorithms.
XQuery is a powerful language, which encompasses SQL and much more. Notably, it is able to query rich and extensible data types; it is a functional language, so that any valid expression applied to a valid expression is a valid query; it will soon incorporate XQuery Text for full text queries. XQuery Text shall provide functionalities as single-word search, phrase search, support for stop words, search on prefix, postfix, infix, proximity searching, word normalization, diacritics, ranking and

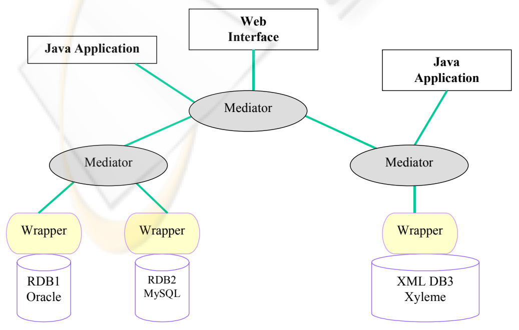

Figure 1: XLive architecture

relevance. All these features will make XQuery an ideal language for querying in an integrated way heterogeneous data sources.

2.2 Overview of XLive Mediator

In the XLive project, we use a mediation architecture to support enterprise information integration shown in Figure 1. It follows the classical wrapper-mediator architecture as defined in (Wiederhold, 1992). The communication between wrappers and mediator follows a common interface, which is defined by an applicative Java or Web service interface named XML/DBC. With XML/DBC, requests are defined in XQuery and results are returned in text XML format.
Our architecture is composed of mediators that deal with distributed XML sources and wrappers that cope with the heterogeneity of the sources (DBMS, Web pages, etc.). The XLive mediator is a data integration middleware managing XML views of heterogeneous data sources. Using XLive mediator one can integrate heterogeneous data sources without replicating their data while the sources remain autonomous.
XLive mediator is entirely based on W3C standard technology: XML, XQuery, XML-Schema, SAX, DOM and SOAP. All information exchanges rely on XML format. XML-Schema is used for metadata representation. Wrappers provide schemas to export information about local data structures. XQuery is employed for querying both the mediator and the wrappers. Connectivity of mediator and wrappers relies on the XML/DBC programming interface, an extension of JDBC to integrate XQuery. More information about the XLive mediator can be found in (Dang-Ngoc, 2003).

3 PROPOSED BENCHMARK

Several benchmarks have been developed for XML DBMSs, among them XMach-1 (XMach-1, 2001), XMark (XMark, 2001) , X007 (X007, 2002), XBench (XBench, 2004). They all have their interests, but are in general too complex for current mediators, both in functionality and size. In this section, we introduce our simpler benchmark.

3.1 Presentation

We propose a simple generic benchmark for testing the basic functionalities of an XML mediator and evaluating the performances of the different join algorithms and indexing schemas of the local sources. The existing benchmarks generally propose a set of complex queries that evaluate many of the properties of the query processor in the same query. By appealing at simple operations, our goal is to stress only certain functions: local indexing, XML transfer and parsing, join algorithms, etc. Another reason for proposing only simple queries is that we used our benchmark to test the XLive mediator that performs basic XQuery to integrate multiple sources. Generally, it takes a long time for a mediator to perform complex join operations (time that depends on the mediator join algorithms and on other external parameters as the network delay, the distant DBMS capabilities, and on the source speed to transfer the results). Yet another reason to use a simple XQuery benchmark is that most tested DBMSs only support the core of XQuery with realistic performance on the computer we are using.

3.2 Data Set

The data set is composed of 2 document models: one data oriented and the other text oriented. With a small depth (of maximum 3) and a small width (of maximum 5), the two documents have a simple structure that facilitates the evaluation of different structural selection queries. The two documents are logically connected, which gives us the possibility to perform simple join operations between documents that are located on different systems. A graphical representation of the schema of the two documents is given in Figure 2 and 3. The schema is variable in the sense that neither the number of "authors" of a book nor the number of paragraphs in the reviews are constant. The textual content is generated from the most popular English words extracted from Shakespeare's plays.

In order to evaluate the performances of the XML systems, we generated 3 data sets with 300/750/1500 documents, each documents having a size less than 2k. We used the utility toXgene (toXgene) and we started from a provided example for generating our data set.

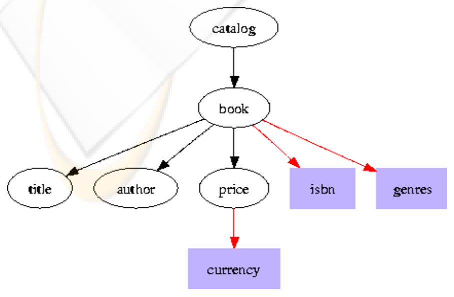

Figure 2: Catalog schema

3.3 Queries

Our benchmark proposes a representative set of XML DBMS query functionalities, which can be grouped as follows:

(i) Simple XPath expressions.
Queries Q1 and Q2 represents XQueries that require selections on the elements and attributes names:
Q1: for $b in collection("catalog") /catalog/book return $b
Q2: for $currency in collection("catalog")/catalog/book/price/@currency return $currency
(ii) XPath with predicates.
Q3, Q4, Q5 introduce predicates to perform simple selections.
Q3 predicate tests for exact equality:
Q3: for $b in collection("catalog") /catalog/book where $b/price/@currency = "CDN" return $b
Q4 contains a "range" predicate:
Q4: for $b in collection("catalog") /catalog/book where $b/price < 100 return $b
Q5 contains the two previous predicates:
Q5: for $b in collection("catalog") /catalog/book where $b/price < 100 and $b/price/@currency = "CDN" return $b
(iii) Recursive Path optimization.
Q6 contains a recursive wildcard "//" expression that tests for the optimality of the path evaluation :
Q6: for $col in collection("catalog") return $col//price
(iv) Result ordering.
For testing the performances of generating an ordered result, we have introduced an order-by XQuery:
Q7: for $col_rev in collection("review"), $rev in $col_rev/review, $rate in $rev/review/@rating order by ($rate) return $rev
(v) Text search.
Q8 contains the "contains" predicate to stress some text indexing capabilities:
Q8: for $b in collection("catalog") /catalog/book where contains($b/author, "Fumio") return $b
(vi) Joins on values.
Q9 and Q10 require joins between the two documents; Q9 performs join and text searching:
Q9: for $col_cat in collection("catalog"), $col_rev in collection("review"), $b in $col_cat/catalog/book, $rev in $col_rev/review, $rev_rev in $rev/review where $b/@isbn=$rev/book/@isbn and contains($rev_rev,"dolphins") return $b/@genres.
Q10 performs equality join:
Q10: for $col_cat in collection("catalog"), $rev_cat in collection("review"), $b in $col_cat/catalog/book, $r in $rev_cat/review where $b/@isbn=$r/book/@isbn return $r/review/@rating
(vii) Result generation.
Q11 tests the performances of the "query processor" to generate new results:

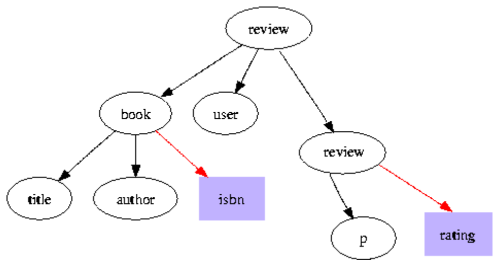

Figure 3: Review schema

```
Q11:  for $col_rev in collection("review"), $rev in
$col_rev/review where $rev/review/@rating <2
return
<lowRateBook>
<title>{$rev/book/title/text()}</title>
 {for $col_cat in collection("catalog"),
 $b in $col_cat/book
 where $b/@isbn=$rev/book/@isbn
 return <price>$b/price/text()</price>}
</lowRateBook>
```

3.4 Metrics

For evaluating a query processor, we measure the query execution time and the size (in bytes) of the result. For running the benchmark and evaluating the different DBMSs, we have used a PC Intel® Pentium® M processor 1600MHz with 1 gigabytes of main memory. All the systems were evaluated using the provided Java API.

4 MEDIATOR EVALUATION

We run the benchmark queries on top of the XLive mediator using two native XML DBMSs as data sources, namely XDBMS1 and XDBMS2. With multiple data sources, times are sensibly the same as with one. Thus, we only report the results for the mediator on top of a unique data source.

4.1 Results of experiments

Tables 1,2 and 3 present the results of evaluating the query using a mediator on top of XDBMS1 and XDBMS2 for all the data sets. Most time in the mediator is taken to iterate on the intermediate results and construct the final result. As XLive exchanges data with sources in text XML (as with Web services), a reparsing of all the partial results is required, which is costly in Java on a small portable computer. Better results could be obtained if the mediator would use a cache for temporary storing source query results in an easy to serialize format.

Table 1: Mediator results for DS1

Query	Time		Results
	XDBMS1	XDBMS2	elements
Q1	444,5	524,8	100
Q2	245,4	213,1	100
Q3	504,2	417,8	100
Q4	333,4	264,9	69
Q5	422,2	306,8	69
Q6	206,9	269,1	100
Q7	992,1	2151,8	200
Q8	137,5	4,61	4
Q9	423,3	1939,6	79
Q10	698,6	3293,5	200
Q11	945,5	1527,5	60

Table 2: Mediator results for DS2

Query	Time		Results
	XDBMS1	XDBMS2	elements
Q1	758,9	2378,5	250
Q2	335,7	313,6	250
Q3	879,1	1804,8	250
Q4	652,1	866,0	168
Q5	674,8	893,2	168
Q6	263,9	263,7	250
Q7	1242,7	3412,0	500
Q8	136,8	7,24	10
Q9	508,0	3128,4	212
Q10	997,5	7986,1	500
Q11	2008,4	2854,8	148

Table 3: Mediator results for DS3

Query	Time		Results
	XDBMS1	XDBMS2	elements
Q1	1106,6	7490,3	500
Q2	388,3	759,5	500
Q3	1144,5	8174,7	500
Q4	759,2	3998,3	339
Q5	739,1	3661,3	339
Q6	933,8	979,3	500
Q7	1887,5	4816,6	1000
Q8	131,8	8,52	15
Q9	829,6	6160,3	428
Q10	1586,4	14962,6	1000
Q11	4411,7	4051,7	285

4.2 Some Discussions

It is important to mention that the mediator evaluation time is strongly influenced by the Java API provided by the mediated DBMSs. The generated XQuery sub-queries are in general the best possible, but compliant with the local source capabilities.
Another important point is that at the mediator level, it is not always possible to benefit from the best indexing techniques of each local data source. For example when evaluating Q8 on XDBMS2, in order to take advantage of the text indexation, it is required to use the non-XQuery function “XDBMS2:fts” of XDBMS2. On the other hand the mediator supports standard XQuery with no specific functions. Thus, an optimized translation from XQuery to XDBMS2 functions would require more parameters and a constant phasing of the wrapper with the vendor's different optimal functions. Another actual problem that penalizes the mediator evaluation is the translation between the XLive XQuery to real DBMSs, which are in reality far from the standards. These points demonstrate the high importance of compliance to standards for efficient mediation of XML DBMSs.
For DS1, total time for running the whole benchmark with XDBMS1 is 499 ms while it is 5353 ms with the mediator on top of XDBMS1. This shows an average factor of 10, mainly due to data transfer and parsing. Total time with XDBMS2 is 71 versus 922 with the mediator on top of XDBMS2. This shows an average factor of 13. The global difference may come from the quality of the wrapper (better optimizations have been made with XDBMS1). Other ratios with the other data sets DS2 and DS3 are a bit better (approximately 7 and 5) for XDBMS1. The more reduced ratios are caused by the fact that the query processing time, at XDBMS1 level, grows “faster” than the time required to parse additional results, at the mediator level.
Figures 4, 5 and 6 gives the detailed ratios between the response time with mediator versus direct response time. The ratio for XDBMS2 increases for

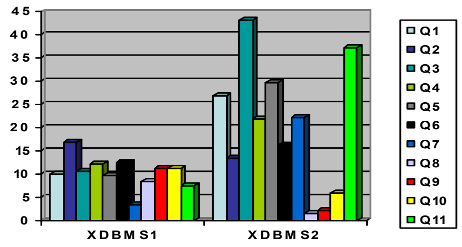

Figure 5: DS2 Mediator time increasing factor

bigger data sets. This means that the time required to analyze more results (due to iteration, parsing, and serialization) grows “faster” than the additional time required by XDBMS2 to generate more results.
The same conclusion appears if we consider the returned result: the first query returns a book (an element node with many children having a big size) and has the biggest increasing ratio for XDBMS2; the second query returns only a text node and the ratio increases very little (the same happens for Q8). This means that indeed the mediator overhead greatly depends on the analysis of the entire result structure.
Q9 and Q10 (the join queries) have a very reduced ratio that is caused by the fact that XDBMS2 does not compute very well the joins.

5 CONCLUSION

In this paper, we introduced a new benchmark composed of three data sets and 11 queries. It was designed to evaluate commercial DBMSs behaviors in a mediation architecture. Data sets are small in order to make the benchmark fast to run. The data sets are designed to cover several existing XML schemas. The set of queries is composed of simple queries that stress only the most important parts of a DBMS and give fast results. However our query set covers the typically used XQuery functionality. We tried to compose our benchmark according to previous existing benchmarks and to the

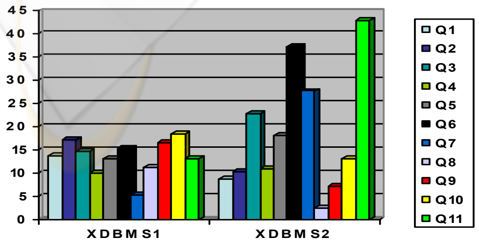

Figure 4: DS1 Mediator time increasing factor

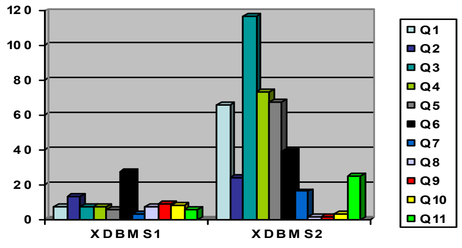

Figure 6: DS3 Mediator time increasing factor

specifications that are already implemented in the considered source DBMSs. We tried to design and run the benchmark without favoring one DBMS versus another. The results were not post processed because we wanted to present the real performances of XML systems.
For presenting the actual "value" of some existing XML DBMS, we ran our benchmark on two popular systems and discovered which are their strong points and weaknesses. Further, we also evaluate the XQuark XQuery bridge, an open source mapping XML-XQuery to table-SQL. The behaviour of the bridge (reported in full paper see www.gardarin.org) is not very different from that of XDBMS1.
We evaluate the XLive mediator with the goal of discovering some means of optimization. Our results can help in the future development of the mediator versions. It is realistic to say that the best optimization should be the replacement of XML text for data exchange by some parsed binary format encoding XML in a compact way and avoiding parsing and reparsing.
As part of our benchmark experience, we like to say that the existence of a mediator facilitates the evaluation of existing DBMSs offering a general integrated platform: one single XQuery implementation with a uniform API, whatever be the DBMS type and specificities. In the same time the mediator may add additional functionalities to some DBMSs that have a weak implementation of the standard query languages (XQuery in our case). According to our tests, the mediator can even accelerate the evaluation of certain queries taking advantage of its query decomposition module.
The benchmark also shows that XLive can be improved by changing the exchange format (e.g., compressed XML avoiding reparsing would be great), adding some new modules (e.g., parameterized query compilation, a persistent cache for storing partial results), optimizing existing ones (e.g., memory allocation techniques, join and sort algorithms), improving wrappers for specific DBMSs (e.g., performing distributed index management for XML sources).

REFERENCES

XQuery, 2004. *XQuery 1.0: An XML Query Language.* http://www.w3.org/TR/2004/WD-xquery-20041029/

Papakonstantinou, 2003. Yannis Papakonstantinou et.al.: XML queries and algebra in the Enosys integration platform, Data Knowl. Eng. 44(3): 299-322

Gardarin, 2002. Georges Gardarin, Antoine Mensch, Anthony Tomasic: An Introduction to the e-XML Data Integration Suite, EDBT 2002: 297-306

Wiederhold, 1992. *Wiederhold G.: Intelligent Integration of Information,* ACM SIGMOD Conf. on Management of data, Washington D.C., USA, 1993, 434-437.

Dang-Ngoc, 2003. Tuyet-Tram Dang-Ngoc, Georges Gardarin.: Federating Heterogeneous Data Sources With XML, IASTED IKS 2003: Scottsdale, AZ, USA, Nov. 2003.

XMach-1, 2001. *Rahm, E., Böhme, T.: XMach-1: A Multi-User Benchmark for XML Data Management,* Proc. VLDB workshop Efficiency and Effectiveness of XML Tools, and Techniques (EEXTT2002), Hongkong, Aug. 2002

XMark, 2001. A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, R. Busse:The XML Benchmark Project, Technical Report INS-R0103, CWI, Amsterdam, The Netherlands, April 2001

X007, 2002. Stéphane Bressan, Mong Li Lee, Ying Guang Li, Zoé Lacroix and Ullas Nambiar: The XOO7 Benchmark, in the Lecture Notes in Computer Science series (ISBN 3-540-00736-9), Springer-Verlag, pp146-147.

XBench, 2004. *B. B. Yao, M. T. Özsu, and N. Khandelwal: XBench Benchmark and Performance Testing of XML DBMSs,* In Proceedings of 20th International Conference on Data Engineering, Boston, MA, March 2004, pages 621-632.

toXgene, 2002. www.cs.toronto.edu/tox/**toxgene**/