Towards the Development of XML Benchmark for XML Updates

Binh Viet Phan and Eric Pardede
Department of Computer Science and Computer Engineering
La Trobe University, Bundoora VIC 3083, Australia
Email: {vbphan@students, e.pardede@}latrobe.edu.au

Abstract

Many XML Benchmarks have been proposed to study strengths and weaknesses of any given XML database system. All existing benchmarks can be applied to evaluate data retrieval queries, but cannot be used to evaluate update performance of XML database systems. Therefore, this paper will propose criteria to evaluate the abilities to update XML documents of XML database systems. These criteria will be realized by a data set and corresponding queries that are applied to benchmark XML updates. Finally, the competence of the proposal will be examined by using a case study.

Key Words- XML Benchmark, XML updates, XQuery, XML Database

1. Introduction

In recent years, there have been various XML Database Management Systems (DBMS) being proposed for storing and processing XML data. Each DBMS has particular characteristics with certain strengths as well as shortcomings [1]. The lack of standard for XML as data format also affects the differences in XML DBMS implementations. This fact has raised needs for tools to analyze the capabilities of any given XML DBMS. With these tools, which are also known as benchmark tools, users can find the most suitable XML DBMS for their particular business applications.

Hitherto, some XML Benchmarks have been proposed. These include the Michigan Benchmark [2, 3], XMark [4, 5], XMach-1 [6, 7] and XOO7 [8,9,10]. These benchmarks are applied to examine various characteristics of XML DBMS. Some benchmarks are useful to evaluate performance of primitive XML query operations such as the Michigan Benchmark. Others are employed to assess features of XML query processing or evaluate overall performances of whole XML DBMS. However, the existing XML Benchmarks do not focus on benchmarking update operations, which has become an imperative part of XML DBMS. Additionally, there are many key aspects are not covered in these benchmarks such as time for bulk loading and data parsing, roles of XML schema, and other storage aspects.

Nowadays, many XML business applications such as data in e-auction systems, catalogues in many e-commercial industries, document versioning in academic or governments require intensive update operations. The capability of the DBMS to perform efficient updates can be a crucial factor for users to determine their choice of a DBMS.

This research will propose a tool that can be useful for benchmarking XML update operations. This paper firstly specifies key aspects for benchmarking XML updates. Based on these aspects, performance metrics, the data set and the benchmark queries will be developed (see Fig. 1). The result should be applicable for various XML DBMS types.

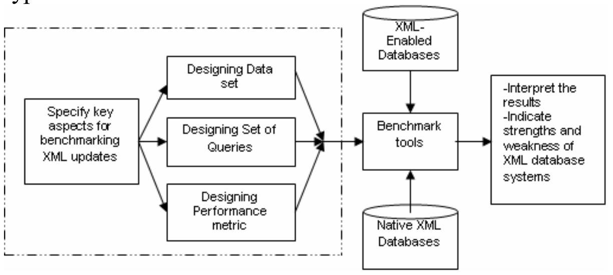

Specify key aspects for benchmarking XML updates
Designing Data set
Designing Set of Queries
Designing Performance metric
XML-Enabled Databases
Benchmark tools
Native XML Databases
-Interpret the results
-Indicate strengths and weakness of XML database systems

Fig. 1 Frameworks of this research

The remainder of this paper is organized as follows. Section 2 will discuss background of the research and existing works. In section 3, criteria for XML update Benchmark will be proposed. The data set and benchmark queries will be designed in section 4. The proposal benchmark in this paper will be evaluated in section 5. Finally, conclusions and future works will be provided in section 6.

2 Background and Existing Works

To be successful in evaluating any given XML database system, XML Benchmarks should satisfy conventional principles and functionalities [5, 11]. There are few major works in the area of XML benchmarks, including the Michigan Benchmark, XMark, X007 and X-Mach1.

Michigan Benchmark [2,3] focuses on evaluating performance of primitive operations by a large number of benchmark queries against a single and large XML document. It applies seven queries containing insertion and deletion operations to assess XML updates. It is noted that this benchmark does not apply any particular scenario for its data set. Measurement metric in the Michigan Benchmark is query response time in second.

XMark [4, 5] and XOO7 [6, 7] emphasize on assessing capability of XML query processors. XMark uses a scenario and data set from an internet auction site, whereas XOO7 does not use any specific application for its scenario. The numbers of benchmark queries are 20 and 18 in XMark and XOO7, respectively. These Benchmarks are applied in single-user environments against a single data centric document. However, XMark and XOO7 totally ignore updates evaluation.

XMach-1 [8, 9, 10] can be used in multi-user environments with a data set consisting of multiple documents. It also focuses on evaluation of document-centric aspects. Update performance of XML database systems is evaluated by 3 complex queries so it is hard to conclude that XMach-1 can cover important characteristic of XML database systems in term of XML updates.

Lack of evaluation of XML updates is the most problematic of current XML Benchmarks. Many Benchmarks such as XMark, and XOO7 ignore evaluating XML update operations. Therefore, these benchmarks are not appropriate for real-world problems that usually need to update XML databases.

In addition, performance metric is an important factor for XML Benchmarks because the result of benchmarking processes will be presented and interpreted by the metric. Measurement metrics applied in existing benchmarks are only single metric such as "query respond time" or "XML query per second" (Xqps). They simply calculate total time for query executions or count number of queries processed per second. Therefore, it is difficult to isolate and specify which processes have poor performance. Particularly, with single metric, results of benchmark processes cannot be used to analyze impacts of algorithms, and techniques applied in XML DBMS.

3 Criteria for Assessing XML Updates

To benchmark XML updates, we need to specify characteristics that have strong impacts on update processes. Eight points applied to evaluate update performance of XML DBMS will be proposed below.

Bulk loading and Data Parsing. They are the processes of loading and parsing XML documents into internal representation of XML DBMS. The processes assist to reduce number of I/O disk operators while executing XML queries. Moreover, some XML DBMS do not support node-based insertion and thus, the XML documents must be loaded into the database systems and parsed into the internal representation [5].

Query Parsing. Amount of time to translate and to parse XML queries in internal language is used to evaluate query parsing. Time for query parsing is varied among different XML DBMS such XML-Enabled DBMS and Native XML DBMS. Amount of time for parsing XML queries should be considered and isolated and it will specify costs of real update processes.

XML Index. Indexing XML has central impacts on structural joins that is applied to find relationships among nodes during implementing updates. Also, index can assist to determine node(s) without physically accessing original XML documents. This can reduce invoking I/O disk operators, and assist to efficiently process XML queries. Finally, mechanisms of re-indexing or re-labeling node(s) have significant impacts on whole update process because of time consuming.

XML Schema. Schemas have important roles in three areas: reducing of casting data types, mapping XML data, and constraint validations of XML documents. Schema-based XML will save time in casting data types at query execution time, while in schemaless XML, the inexistence of schema will result in spending more time for coercion of data types. Additionally, [12, 13] indicate that XML schemas bring four advantages including: reduction of disk space consumption, easy query validation by adjusting relative path to absolute path, identification of nodes relationships and improved indexing. Therefore, with many significant effects of XML schemas on critical aspects related to XML updates, it is important to evaluate XML schemas roles while benchmarking XML updates.

Preserving Orders. Costs for preserving orders of elements in XML documents are expensive. Orders of nodes are usually violated when there are some changes in the documents. Hence, impacts of preserving orders need to be considered in benchmarking XML updates.

Missing Elements. There are different methods used to compact and to store the missing elements in XML DBMSs. These compact techniques assist to decrease redundant data so the XML DBMS can reduce amount of disk space consumed to store XML documents. This facilitates efficient storage, course of processing XML document, as well as processing queries [14].

Reconstruction. Processes of reconstruction XML documents are challenging for both XML-Enabled and Native XML database because mechanisms to reconstruct XML documents involve many operations such as in mapping, order preservation, joining tables etc. For those reasons, reconstruction is an important issue needed to evaluate while benchmarking update performance.

XML Storage Issues. This issue will determine the mapping and joining techniques used in the database systems. Also it determines the capabilities of the systems to restructure XML data. With respect to XML update, storage approaches are concerned as important factors. Impacts of XML storage approach will be measured by amount of disk space for storing XML documents and their corresponding XML schemas. Additionally, other effects of the storing approach on XML updates will also be accumulated in query execution time.

4 Benchmark Design

In this section we propose the data set and the queries for the proposed benchmark. The query chosen is XQuery as the current W3C standard for XML database [15].

In this proposal, the performance metric used is a combination of "query response time" in second, and "disk space consumed" in Mb. Xqps can be applied when we simulate multi-user environment by executing this benchmark queries in multiple threads.

4.1 Benchmark Data Set

Benchmark data set consists of multiple XML documents including some very large documents. These documents are divided into two main groups named *Authors*.xml* and *Books*.xml*. These groups are based on two XML schemas named *Authors.xsd* and *Books.xsd* (see Fig.3). These XML documents are designed to examine update performance while updating multiple XML documents. The size are considerably large for the purpose of examining bulk loading and data parsing

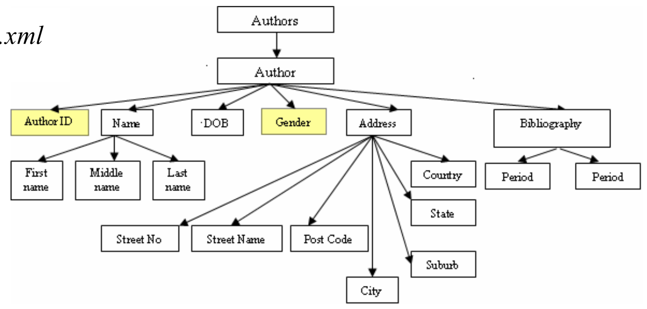
Some important information regarding Author documents is described as follows. *DOB* is designed to use environmental information such as current-date() as a part of update queries. Document-centric aspects are represented by element *Bibliography*. It contains both text and *Period* elements. This element is employed to examine update performance on textual fields. *AuthorID* is used to evaluate update performance in referential cases. Attribute *ID* and *Gender* is structured to benchmark performance of update operations over attributes.

Some important information regarding Book documents is described as follows. Each *Volume* of books has an attribute *ISBN*; and can have its own *Subtitle* along with the title of the book. *Editor* element is designed to exam preserving orders of elements while renaming elements. Elements *Chapter* and *Section* are designed to examine update performance on missing elements. Chapter is designed recursively to increase levels of XML tree. It is also used to assess preserving order of elements.

4.2 Benchmark Queries

There are 28 queries, which are designed against the data set to evaluate the update performance of XML DBMSs. These queries are divided into 6 groups (see Table 1). In each group, one query will be provided as an illustration.

a: Authors.xml



b: books.xml

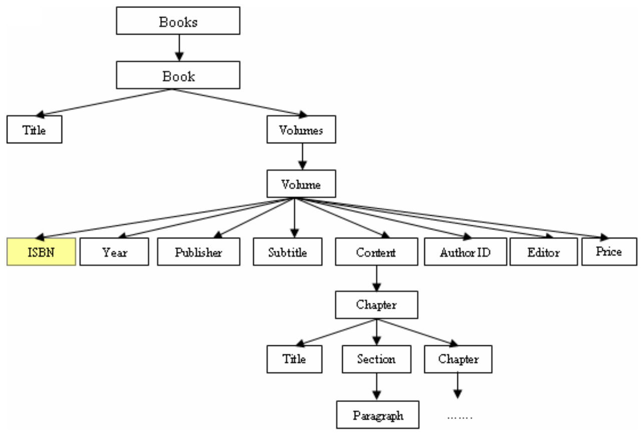


Fig. 3: Data Set Structure

Table 1. Brief Description of Benchmark Queries

Criteria	Query	Descriptions
Bulk loading and Data Parsing	Q 1	Insertion of an attribute
	Q 2	Insertion of a node
	Q 3	Update a node
Evaluation of XML Index	Q 4	Insert a sub-tree as a first child
	Q 5	Insert a sub-tree as a last child
	Q 6	Insert a textual fragment at the shallow level of hierarchy
	Q 7	Insert a textual fragment at deep level of hierarchy
	Q 8	Delete a sub-tree
	Q 9	Delete elements and leaf nodes
	Q 10	Concentrated Insertion
	Q 11	Scattered Insertion
	Q 12	Bulk deletion of nodes that contain a textual fragment specified
Preserver Ordering Elements	Q 13	Rename an element
	Q 14	Bulk deletion of nodes at different position
	Q 15	Insert a textual fragment
Missing Elements	Q 16	Create XML document that is missed similar elements at the first 100 nodes.
	Q 17	Create XML document that is missed similar elements at random 100 nodes
	Q 18	Update on missing elements
Reconstruction	Q 19	Delete elements, which satisfy complex conditions
	Q 20	Exchange name between 2 elements
	Q 21	Replace contents of an element
	Q 22	Reconstruction of a new file from an existed XML document
	Q 23	Bulk deletion of nodes at random positions
	Q 24	Return a number of XML documents
XML Storage	Q 25	Disk space usage: create a new XML document with a lager number of nodes.
	Q 26	Restructuring: exchange positions of a parent nod and its child
	Q 27	Restructuring: Return a set of documents; contents of these returned documents are modified. The returned documents must be conformed to XML schema of the original XML document
	Q 28	Delete a whole document.

Bulk loading and Data Parsing. There is no common bulk-load utility for every XML DBMS. Thus, it is difficult to benchmark the ability to bulk load and parse XML data using bulk load utility. Therefore, insert statements will be used to assess bulk loading and data parsing. Performance metric for assessing is query response time.

```
Example: Query 1
let $attribute in
doc(Authors.xml)/Authors/Author[10]
return do insert attribute Gender "Male" into
$attribute
```

Evaluation of XML Index. Impacts of XML index on updates will be examined by nine queries with various characteristics at different orders and levels. Concentrated insertion, scattered insertion are also employed to examine impacts of XML index for re-labeling in worst cases [16].

```
Example: Query 7
do insert
  <Paragraph>
    ---- The beginning ----
  <Paragraph>
as first into
doc("Books.xml")/Books/Book[5]/Volume[last()]
/Content/Chapter[first()]/Section[first()]
```

Preserving Ordering Elements. Three queries are applied to challenge capability of preserving orders of elements. Measurement metric is query response time in second.

```
Example: Query 13
for $node in
doc("Books.xml")/Books/Book/Volumes/Volume[@I
SBN = 123456]
return do rename $node/Author[1] as "Editor"
```

Missing Elements. There are three queries proposed for measuring this criteria. The metrics used are disk space usage and query respond time.

```
Example: Query 18
for $author in
doc("Authors.xml")/Authors/Author
where not exists($author/Bibliography)
return insert <Bibliography>A new
Author</Bibliography> into $author
```

Reconstruction. Queries in this group will challenge the ability to reconstruct XML data while deleting, renaming, and replacing data in XML documents. Query 19 is complex to ensure that joining among different relations or documents are needed to execute updating.

```
Example: Query 19
for $author in
doc("Authors.xml")/Authors/Author, $book in
doc("Books.xml")/Books/Book
for $authorID in $book/Volumes/Volume/Author
ID
```

```
where ($author/Author ID = $authorID ) and
($authorID >100) and
contains($author/Name/LastName, "Nguyen" ))
and (contains($book/Title, "Database
system"))
return do delete $author.
```

XML Storage. This aspect can be evaluated by inspecting three issues, which are disk space to store XML documents, ability to restructure XML documents, and efficiency of deletion of whole a XML document. The metrics are also combination of disk space consumed and query response time.

```
Example: Query 26
let $chapter :=
doc(Books.xml")/Books/Book[2]/Volumes/Volume[
2]/Content/Chapter[1]
return
(do insert $chapter/Chapter[last()] after
$chapter,
do delete $chapter/Chapter[last()] )
```

Evaluation Roles of Schema and Query Parsing. It is not needed to generate separate queries against the data set in order to evaluate roles of XML schemas. In such cases, impacts of XML schemas on update processes will be assessed by comparing the execution of benchmark queries with and without XML schemas. It is noted that the disk space for a schema-less document is the size of that document, whereas the disk space for a schema-based document includes both size of XML document and its corresponding schemas.

Amount of time for queries parsing is usually reported by XML DBMS themselves. Thus, it is easy to get parsing query time, and analyze it. However, some XML database systems are still not able to inform the amount of time for parsing queries.

5 Evaluations and Case Study

5.1 Evaluations

The proposed benchmark can be used to assess various update operations in detail, covering all update operations supported by current XQuery [15]. On the other hand, existing benchmarks only cover limited update operations. Moreover, the proposed benchmark can be used for multiple documents updates.

It also applies multiple performance metrics including "query response time" and "disk space consumed". Moreover, Xqps will be used when implementing the benchmark queries against the data set in simulation of multi-user environments. Thus, the proposal can cover more update aspects than that can be reported by using other existing XML benchmarks.

In addition, all scenarios in the proposed benchmark follow the latest standard on XML Update Facility [17] and XML Update Facility Use Cases [18].

Finally, this proposal clearly categorizes 8 groups of aspects that are affected by XML updates. Hence, this assists in studying each aspect more easily towards performance optimization. Table 2 shows how the proposed benchmark is compared to the existing works.

5.2 Case Study

In this section we apply a case study of an internet auction site taken from W3C Case Study [17, 18]. Then, we show how the proposed benchmark can be used for benchmarking typical update requirements for the particular case study.

The application requires three main documents for *users*, *items*, and *bids* along with their corresponding schemas [17, 18].

Update operations are frequently performed on multiple documents. Environmental information such as current date and time are also used to update these documents. There are eight typical update requirements in this case study. These requirements include insert, delete, replace and rename operations with various complexity. Due to the page limitation, we do not show each requirement in this paper.

Table 3 specifies how existing benchmarks and our proposed benchmark satisfy the case requirements. For every requirement, the proposal offers various queries for benchmarking. It is needed to note that, both the Michigan Benchmark and XMach-1 only update on a single document, using limited update operations without using environmental information.

Table 2. Comparison of Proposed Benchmark and Existing Benchmarks

	Michigan Bench.	**XMark**	**XMach-1**	**XOO7**	**Proposed Benchmark**
Domain-Specification	Core query Operators	Query Processor	DBMS	Query Processor	Query Processor and Core query operator
Environment	Single-user	Single-user	Multi-user	Single-user	Single-user (Possible for multi-user)
Benchmark data	Data-Centric	Data-Centric	Document-Centric	Data-Centric	Data & Document-Centric,
Number of documents	Single	Single	Multiple	Single	Multiple
Schema Support	DTD	DTD	XML Schema	DTD	XML Schema
Number of Update Queries	7	0	3	0	28
XML Storage Aspects	No	No	Few	No	Yes
Performance Metric	Response Time	Response Time	Xqps	Response Time	Response Time, Disk Space (Possible for Xqps)

Table 3. Requirements are Assesed by Each XML Benchmark.

Requirement. No	Michigan Bench.	XMark	XMach-1	XOO7	Proposed Bench.
1	✓	✗	✓	✗	✓ (Q2, Q4, Q5, Q6, Q7, Q10, Q11, Q15)
2	✓	✗	✓	✗	✓ (Q4, Q5, Q6, Q7)
3	✗	✗	✓	✗	✓ (Q8, Q12, Q14, Q19, Q23)
4	✗	✗	✗	✗	✓ (Q14, Q19, Q23)
5	✗	✗	✗	✗	✓ (Q10, Q11)
6	✗	✗	✗	✗	✓ (Q3, Q18, Q21)
7	✗	✗	✓	✗	✓ (Q3, Q18, Q21, Q13, Q20)
8	✗	✗	✗	✗	✓(Q13, Q20)

The case study has demonstrated that the proposed benchmark has complemented existing benchmarks for analysing update operations.

6 Conclusion and Future Works

Existing XML benchmarks cannot evaluate many important functionalities of XML database, particularly for XML updates operations and its performance metric. In addition, they are lacking of modes to evaluate impacts of XML Index, XML storage approaches, and XML schemas towards XML updates.

In this paper, we established new criteria for benchmarking XML updates. The criteria are realized by designing performance metrics, data set, and benchmark queries. The data set and corresponding queries are carefully designed to cover most functionality for XML query language, and challenge most critical aspects of XML database in term of XML updates. The competence of the proposed benchmark is evaluated and compared with the existing benchmarks in assessing a W3C-based case study.

For future works more criteria can be added, especially by incorporating roles of XML namespace and XML schemas on the updates. In addition, how the impacts of multiple-users updates towards concurrency can also be investigated.

Another way to expand this study is by building a hybrid benchmark based on existing benchmarks such as XMark or XMach-1. While our proposed benchmark can be used for update benchmark, the existing benchmarks are used to evaluate data retrieval.

References

[1] C. Scmauch and T. Fellhauer, "A Comparison of Database Approaches for Storing XML Documents," in *XML Data Management: Native XML and XML-Enabled Database Systems*, Addison-Wesley, 2003, pp. 519-546.

[2] J. M. Patel and H. V. Jagadish, "The Michigan Benchmark: A Micro-Benchmark for XML Query Performance Diagnostics," in *XML Data Management: Native XML and XML-Enable Database systems*, Addison-Wesley, 2003, pp. 499-518.

[3] K. Runapongsa, J. M. Patel, H. V. Jagadish, and S. Al-Khalifa, "The Michigan Benchmark: A Microbenchmark for XML Query Processing Systems," in *LNCS 2590,* 2003, pp. 160 - 161

[4] A. Schmidt, F. Waas, M. Kersten, M. J. Carey, I. Manolescu, and R. Busse, "XMark: a benchmark for XML data management," in *VLDB 2002*, pp. 974-985.

[5] A. Schmidt, F. Waas, M. Kersten, and D. Florescu, "Why And How To Benchmark XML Databases." *SIGMOD Record 30*, 2001, pp. 27-32.

[6] T. Böhme and E. Rahm, "XMach-1: A Benchmark for XML Data Management," in *Datenbanksysteme in BÜRo, Technik Und Wissenschaft (Btw),* 2001, pp. 264-273.

[7] T. Böhme and E. Rahm, "Multi-user Evaluation of XML Data Management Systems with XMach-1," in *LNCS 2590*, Springer, 2003, pp. 148 - 158.

[8] S. Bressan, M. L. Lee, Y. G. Li, Z. Lacroix, and U. Nambiar, "The XOO7 Benchmark " in *LNCS 2590*, Springer, 2003, pp. 146-147.

[9] Y. G. Li, S. Bressan, G. Dobbie, Z. Lacroix, M. L. Lee, U. Nambiar, and B. Wadhwa, "XOO7: applying OO7 benchmark to XML query processing tool," in *CIKM 2001,* pp. 167 - 174.

[10] S. Bressan, M. L. Lee, Y. Li, Z. Lacroix, and U. Nambiar, "Benchmarking XML Management Systems: The XOO7 Way," in *IIWAS* 2001.

[11] S. Bressan, M. L. Lee, Y. G. Li, Z. Lacroix, and U. B. Nambiar, "XML Management System Benchmarks," in *XML Data Management: Native XML and XML-Enabled Database Systems*, Addison-Wesley, 2003, pp. 477-498.

[12] G. Pallis, K. Stoupa, and A. Vakali, "Storage and access control issues for XML documents," in *Web information systems*, Idea Group Pub, 2004, pp. 104-140.

[13] A. Vakali, B. Catania, and A. Maddalena, "XML Data Stores: Emerging Practices," *Internet Computing 9, 2005,* pp. 62 - 69, 2005.

[14] M. Cokus and Santiago Pericas-Geertsen, "XML Binary Characterization Properties," W3C, 2004.

[15] D. Chamberlin, D. Florescu, and J. Robie, "XQuery Update Facility," W3C, 2006

[16] A. Silberstein, H. He, K. Yi, and J. Yang, "BOXes: efficient maintenance of order-based labeling for dynamic XML data," in *ICDE 2005*, pp. 285-296.

[17] D. Chamberlin, P. Fankhauser, D. Florescu, M. Marchiori, and J. Robie, "XML Query Use Cases," W3C, 2007.

[18] I. Manolescu and J. Robie, "XQuery Update Facility Use Cases," W3C, 2006.