THE 3D XML BENCHMARK

Mohammed Al-Badawi, Siobhán North, Barry Eaglestone
Department of Computer Science, The University of Sheffield, Sheffield, UK
m.badawi@dcs.shef.ac.uk,s.north@dcs.shef.ac.uk, b.eaglestone@sheffield.ac.uk

Keywords: XML Benchmark, XQuery Processing, Performance Evaluation.

Abstract: In the context of benchmarking XML implementations, several XML benchmarks have been produced to either test the application's overall performance or evaluate individual XML functionalities of a specific XML implementation. Among six popular XML benchmarks investigated in this article, all techniques rely on code-generated datasets which disregard many of XML's irregular aspects such as varying the depth and breadth of the XML documents' structure. This paper introduces a new test-model called the "3D XML benchmark" which aims to address these limitations by extending the dataset and query-set of existing XML benchmarks. Our experimental results have shown that XML techniques can perform inconsistently over different XML databases for some query classes, thus justifying the use of an improved benchmark.

1 INTRODUCTION

In the context of XML technology, an XML benchmark is a tool for evaluating and comparing the performance of new XML developments with existing XML technology (Lu et al. 2005). Because of the nature of XML databases and the variety of different platforms used to store these databases (e.g. RDBMS and OO-RDBMS), the benchmarking process mainly examines the performance of the underlying storage-model, the associated query processor and the update handler (Lu et al. 2005). In terms of query processing, the literature (Schmidt et al. 2001) identified ten functionalities to be tested: XML data bulk-loading, XML reconstruction, path traversals, data-type casting, missing elements, order access, reference navigation, joins, construction of large results, containment and full-text searching.

Most of the existing XML benchmarks evaluate the above functionalities using an XML application scenario where a benchmark consists of one or more interrelated XML documents with a limited variation –in most cases- in terms of the database's dimensions including the depth, breadth and size. The query-set pays little or no attention to the impact of the document's nested structure on the XML querying/updating processes. This paper introduces a new XML test-model (called "The 3D XML Benchmark") that extends the existing benchmarks' design to include these features. Experiments, discussed in Section 5, show that the performance of an individual XML techniques is determined by two main factors; the nature of the XML database processed and the inclusion of these features (e.g. the database's three dimensions) in the XQuery syntax.

The rest of this paper is organized as follows. Section 2 reviews the XML benchmarking while the new benchmark is introduced and tested in Sections 3 and 5 respectively. Section 4 describes a node-based scaling algorithm used by the new benchmark to reduce the size of the XML databases; and Section 6 concludes the paper.

2 RELATED WORK

XML benchmarks can be divided into application benchmarks and micro benchmarks. This section reviews six most popular XML benchmarks from both categories, showing their strengths and weaknesses. The characteristics of these benchmarks are summarised in Table 1.

XMark: this benchmark (Schmidt et al. 2002) is widely used by the XML development community because it generates XML databases of any size and covers all XML query-able aspects identified in (Schmidt et al. 2001). The underlying dataset consists of a single, code-generated XML document of a size controlled by a positive floating-point

Tab 1: A Comparison between different XML benchmarks

Benchmark	Dataset					Query-set		
	Source	DB Environment (TC/DC)†	# of Docs	Size	Min/Max Depth	# of Search Queries	# of Update Queries	Depth Aware?
XMark	Synthetic	1 TC rest DC	1	Controlled by SF: tiny (KB) to huge (GB)	12	20	0	No
XOO7	Synthetic	Majority DC Few TC	1	Small Medium Large	5	23	0	No
XBench	Synthetic	Mixed	Mixed	Small (10MB) Normal(199MB) Large(1GB) Huge(10GB)	Limited variation	20	0	No
XMach~1	Synthetic	Mostly TC Few DC	Multi (10^4-10^7 docs)	2KB to 100KB per document	≤ 6 levels	8	3	No
Michigan (MBench~v1)	Synthetic	DC	1	Multiple of 728KB nodes Max. 100 times	5 to 16	28	3	Yes
TPoX	Synthetic	Mix of TC and DC	3.6×10^6 to 3.6×10^{11}	3KB-20KB each	controlled by template	7	10	No

† *TC→Document-centric DB, DC→Data-centric DB*

scaling factor (SF=1.0 produces 100MB), and with a depth which is always 12.

Although it simulates a real-life database scenario, elements in the corresponding XML tree tends to be evenly distributed at each level. This feature omits several irregular aspects of the underlying database such as the diversity in the node's fanouts. Using fixed-depth XML documents is also an issue in this benchmark.

XOO7: This benchmark (Li et al. 2001) is the XML version of the "OO7" (Carey et al. 1993), an evaluation technique used to benchmark object-oriented RDBMS. The benchmark's dataset contains a single document, translated from its base benchmark. The XML file can be produced in three versions: small, medium and large. Regardless its size, the depth of any generated XML file is always 5.

Using code-generated, fix-depth, and only 3-levels of document's size make the benchmark impractical for scalability tests and other irregularity evaluation.

XBench: XBench (Yao et al., 2004) is another XML benchmark which uses code-generated XML documents. In XBench, the underlying dataset can be one of four types: single-document/data-centric (SD/DC), single-document/text-centric (SD/TC), multiple-documents/data-centric (MD/DC) and multiple-documents/text-centric (MD/TC). The size of these documents varies from small (10MB), normal (100MB), large (1BG) and huge (10GB); but the depth ranges over a very limited domain.

Although it uses a template-based generation algorithm which simulates some real database scenarios, the features of the database produced are restricted by the features encoded in the generation templates. The benchmark also does not incorporate the document's depth-variation into the XML querying process.

XMach~1: Among the benchmarks investigated in this paper, XMach~1 (Böhme and Rahm 2003) is a benchmark that targets multi-user environments using a Web-based application scenario. This section only discusses the structure of underlying dataset and query-set.

The benchmark's dataset can contain a huge number (10^4 to 10^7) of XML documents with file-size ranges from 2KB to 100KB. The interrelated XML documents are generated by a parameterised algorithm which controls the size of the document, the number of elements and attributes, the length of textual-contents, and the number of levels in each document. The number of levels is restricted to 6 levels in all documents, and the variation in the number of levels is not incorporated in the query-set design.

Besides the depth-restriction, XML documents generated by XMach-1 are very small, making the benchmark inappropriate for evaluating large scale implementations and/or scalability testing. Furthermore, the query-set does not cover all XML query-able functionalities identified in (Schmidt et al. 2001); examples include path traversal, joins, and aggregation.

MBench~v1: The Michigan Benchmark (Runapongsa et al. 2006) is a micro benchmark used to test individual system functionalities rather than the overall system performance. It is the only benchmark of this category; and it uses a single-document database scenario.

The size of *default* XML document used by XBench~v1 is 728×10^3 nodes, and the number of levels is 16. The size of the *default* XML database can be doubled up to 100 times by varying the node's fanouts (2 to 13) at levels 5, 6, 7 and 8. Also, the size and depth of the database can be controlled by rooting the corresponding XML tree at different levels between 5 and 16.

In the associated query-set, MBench~v1 supports all XML query-types identified by (Schmidt et al. 2001). Unlike other techniques, the benchmark incorporates a depth-dimension in the XML querying process. This is done by scaling the XPath expressions to match the database's maximum depth. MBench~v1 also provides three update queries: inserting a node, deleting a set of nodes and bulk-loading a new XML document into the underlying database.

TPoX: "Transaction Processing over XML" (Nicola et al., 2007) is another XML benchmark that targets multi-user environments using an application-oriented and domain-specific scenario.

The benchmark's dataset is generated by the ToXGene (Barbosa et al., 2002) XML generation tool which uses templates to determine the characteristics of the XML documents produced. Three XML Schemas are used to control TPoX's generation process, producing millions of tiny XML documents of size ranging from 3KB to 20KB depending on the XML Schema used. Additionally, an XML Schema controls the depth and breadth of the XML files generated while element types are preset to be, in general, data-centric with some having large text values.

Unlike other benchmarks, TPoX query set is concerned more with XML updates than search query evaluations. It contains two queries to insert/delete new/existing XML documents respectively, and six queries to alter the contents of existing documents. Seven search queries are used to retrieve information from the underlying dataset with no special attention paid to XML irregularities such as varying the length of XPaths used and testing missing element functionality.

Based on the above, the benchmark has no obvious advantage over its predecessor (XMach~1) in terms of dataset and query set specifications although it simulates more real life business transactions. This is also valid for a newer XML benchmark proposed in (Cohen, 2009) which uses the same sort of code-generated dataset as well as a code-generated XPath query set (Wu et al., 2009).

In summary, the above review has identified three main problems in existing XML benchmarks (Mlynkova, 2008). These are: the use of synthetic datasets which may exclude some real XML irregularities, the use of single-source XML databases which also restricts the diversity of XML structures and the use of fixed-depth datasets and/or query-set which prevents testing of the impact of the nested XML structure. The following section describes an XML test-model which aims to address the above limitations.

3 THE 3D BENCHMARK

3.1 Motivation

The proposed test-model aims to address the limitations identified above by using XML documents from different resources, including synthetic and real databases, so that a natural and logical diversity in the databases dimensions (i.e. the depth, breadth and size) is guaranteed. The proposed test-model must also reflect these features in the query-set design. So, the underlying dataset of the proposed test-model should contain –at least- three different XML databases simultaneously each of which incorporates certain XML features. The dataset and query-set designs are discussed in Section 3.3 and Section 3.4 respectively. The following section describes a framework for the technique's working-environment.

3.2 The Benchmark Architecture

The basic idea of the proposed test-model is to vary three XML aspects of the underlying XML dataset: the size of the database in terms of number of nodes, the depth (i.e. number of levels) and the breadth (i.e. average fanouts). Unlike most existing XML benchmarks (e.g. XMark (Schmidt et al. 2002), XOO7 (Li et al. 2001) and XBench (Böhme and Rahm 2003)), varying the database's depth in the new test-model is based on using distinct XML databases from different sources. This also allows natural diverse irregularity aspects in the underlying XML databases (see Section 3.3). By keeping the size of the dataset members used almost constant with a clear diversity in the number of levels, the

average-breadth degree of each XML database also becomes diverse (Lu et al. 2005).

The *3D XML Benchmark* adapts the XMark's (Schmidt et al. 2002) query-set which consists of twenty queries. Each query (over XMark database) is translated to match the schema of every XML database included in the new dataset. For example, the "Exact Matching" query of the XMark is translated over the other two databases (i.e. DBLP and TreeBank) to become three distinct queries (see Fig 3). Then, each of these queries is reproduced for the other two versions (i.e. reduced XML databases) of each database to add another six queries to the benchmark's query-set. The total number of queries in the entire query-set includes 20×3×3 queries.

The framework is illustrated in Fig 1. The illustration also shows that more XML databases can be added to the underlying XML data-set to represent other XML database scenarios.

The following two sections describe the design of the benchmark's dataset and query-set respectively.

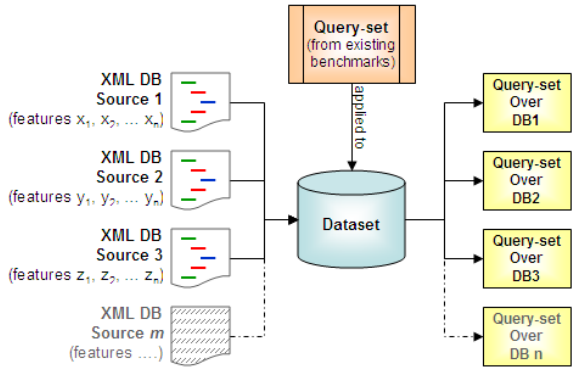

Fig 1: The working-environment for the 3D XML benchmark

3.3 Dataset Design

The new test-model (also called the "3D XML Benchmark") uses multiple XML databases taken from different sources to test the impact of three XML structures (i.e. the depth, the breadth, and the size in terms of the number of nodes). Of the XML *depth*, it is well known (Lu et al. 2005, Runapongsa et al. 2006) that the number of levels affects both the length of the query-evaluation process and the utilization of the underlying system resources (e.g. memory-stack management, CPU usage, I/O devices workload …etc). The memory-stack management is also affected by the breadth of the XML tree due to the nature of the pipelined querying process which needs to spread over more branches (i.e. more paths) at each navigational point (Beglund et al. 2007, Lu et al. 2005). Finally, varying the number of nodes during the XML benchmarking process is an essential tool to test the scalability of any XML implementation (Schmidt et al. 2002, Lu et al. 2005).

3.3.1 Dataset Selection

The base dataset of the 3D XML Benchmark includes two real, single-document XML databases and one synthetic, single-document database. The databases are selected carefully from the existing XML repositories (e.g. (Miklau 2009)) to reflect different categories of the three XML dimensions; that are the *depth*, *breadth* and *size*. At least one database among dataset members must represent the *low*, the *average*, and the *high* category for each dimension respectively. This structure is illustrated in Fig 2. The characteristics of the XML databases selected are as follows (also see Tab 2):

DBLP Database: DBLP stands for Digital Bibliography Library Project, and is a natural, large, data-centric XML document which contains bibliographic information about major computer science publications including journals and proceedings. The database is widely used in XML technology evaluation experiments and is freely downloadable at *http://dblp.uni-trier.de/*. The size of this database is 609MB as of 21st June 2009 but the version used in this study was 127MB and was obtained from *http://www.cs.washington.edu/* website.

XMark Database: This is a code-generated, single-document XML database produced by the XMark-generator (Schmidt et al. 2002) as the base XML database for the XMark benchmarking process. The database simulates an Internet auction application for some dummy products from around the world. The size of the XML documents generated, which range from few kilobytes to hundreds gigabytes, is controlled by a positive floating-point scaling-factor. The majority of this database's elements are data-centric with one descriptive element containing multiple long sentences about the items.

TreeBank Database: This is a single-document, project-based XML database which stores thousands of English sentences tagged to very deep levels. The database is a part of *The Penn Treebank Project* (PennProj 2009) which annotates naturally-occurring text for linguistic structures. The copyrighted text nodes have been encrypted without affecting the overall XML structure of the database. In addition, the deep recursive structure of this data

makes it an interesting case for many experiments (e.g. Liefke and Suciu 2000, Härder et al. 2007). The size of the database used is 82MB and it can be downloaded from *http://www.cs.washington.edu/* website.

Tab 2 provides statistical information about the base XML databases used in the *3D XML Benchmark*.

Tab 2: XML databases used by the 3D XML benchmark

	DBLP (dblp100)	**XMark** (xmark100)	**Treebank** (tree100)
Size (nodes)	2439294	2437669	2437667
#of Levels	6	10[††]	30[†††]
Min Breadth[†]	2	2	2
Max Breadth	222381	34041	56385
Avg Breadth[†]	11	6	3
#of Elems	2176587	1927185	2437666
#of Attrs	262707	510484	1

[†] *Calculations exclude leaf nodes*
[††] *#of level is reduced from 12 while eliminating the text-based elements*
[†††] *#of levels is reduced from 36 to enable the database's management using the available system resources without affecting the dataset specifications*

3.3.2 Varying XML Dimensions

Tab 2 shows that the diversity in the depth and breadth of the dataset is ensured by nature of the databases selected. Rationally, the depth-dimension and the breadth-dimension of an XML database are orthogonal given that the database's size (#of nodes) is constant. So, in a three-point scale, low, average and high, the DBLP database represents the low depth and high breadth categories while the TreeBank database does the opposite. Similarly, XMark is a representative of both the average depth and breadth.

In terms of the size-dimension variation (essential for the scalability testing), the *3D XML Benchmark* scales down each base XML database into two more versions: the first contains 50% of the nodes of the base database and the second contains 25%. The base XML databases are reduced using a node-based, structure-preserving scaling algorithm described in Section 4.

Conventionally, the node's reduction-percentage is attached to the name of base database to define the name of the reduced versions. For example, the 'DBLP100' is given to the base database of the DBLP while the 'DBLP050' defines the halved database and the 'DBLP025' defines the quarter-ized database. Fig 2 shows the location of the nine XML databases in the three-dimensional plane.

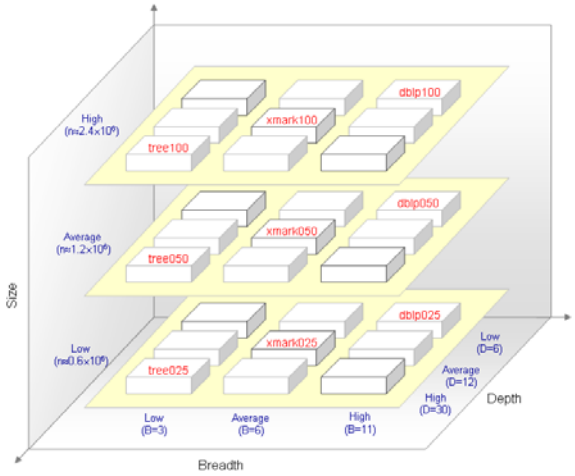

Fig 2: The benchmark's dataset members in the 3D-plane

3.4 Query-set Design

The *3D XML Benchmark* adapts the XMark's (Schmidt et al. 2002) query-set to produce nine different query-sets each of which is executed over a specific XML database of the benchmark's dataset. Using a set of twenty queries grouped into fourteen categories, XMark covers all query-able aspects of XML databases identified by (Schmidt et al. 2001). This section highlights only the query translation process while the complete query-set listing is provided in further publications.

XML queries used by the XMark benchmarking project are expressed using FLWR XQuery (Boag et al. 2007) format. Each XQuery is composed of at least one XPath expression (Beglund et al. 2007) which navigates through the 'auction' XML database (File: 'auction.xml') to reach the desired XML nodes. In the XQuery example illustrated in Fig 3(c, d), the XPath expression seeks the name of an item labelled by 'item20748' which can be purchased from the North of America (namerica) region. So, the corresponding XQuery (shown in Fig 3(c, d)) tests two XPath expressions in order to return the name of target item. The first XPath expression "path 1" is used to locate the searched item while the second XPath expression "path 2" is used to return the name of the item that satisfies the searching criteria.

In the new query-set, the same XQuey syntax can be used over XMARK100, XMARK050 and XMARK025 databases, but with different search criteria in each case. The value of the item_id is changed for each database version to ensure that the item sought is found within approximately the same distance from the root node (e.g. at distance of 27% of the total number of nodes in the database). This is

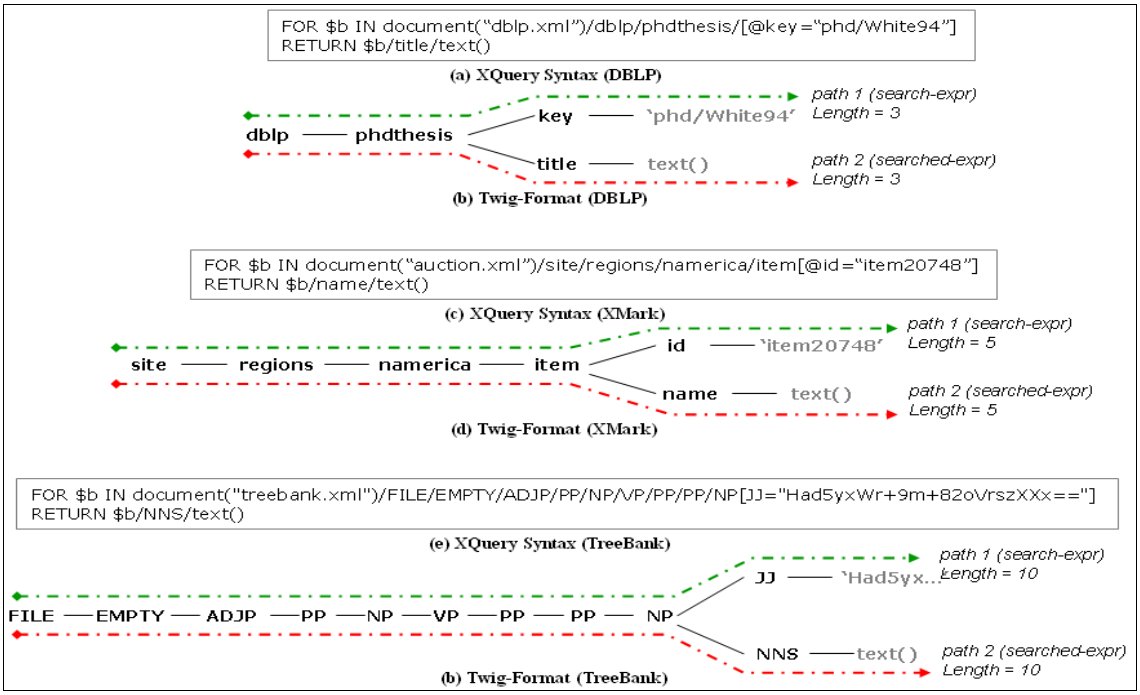

Fig 3: An Example of XQuery Translations (Exact-matching)

Input: 's' is the scaling-percentage (1-100%); 'db1' is an XML database of 'n' nodes
Output: db2 : an XML database of 'm' nodes where m ≅ s×n/100
Parse 'db1' ⇒ construct the path-set and the individual path's node-set;
Consider 's%' of each path's node-set ⇒ an XML tree 'db2' with some *dangling* nodes;
Add all ancestors of *dangling* nodes ⇒ bigger XML tree 'db2' of 'm' nodes;
Repeat until m=(s×n/100) ± d, where 'd' is a very small integer comparing to 'n'
    Adjust 's' according to the value of 'm';
    Consider 's%' of each path's node-set ⇒ an XML tree 'db2' with some *dangling* nodes;
    Add all ancestors of *dangling* nodes ⇒ bigger XML 'db2' tree of 'm' nodes;
End;
Return db2;

Fig 4: The node-based scaling algorithm

important because the relative location of the target data impacts on the overall querying process. For other database categories (i.e. using DBLP or TreeBank instead of XMARK), the length of each XPath expression is scaled down/up to match the depth of the new XML database, and the location-steps are replaced with tag/attribute names from the new database's schema. The length of any XPath expression under the TreeBank (or DBLP) databases is kept (as much as possible) twice (or half) the length of the corresponding expression in the XMark database. Fig 3(a, b) and Fig 3(e, f) depict the translation of the XQuery example in Fig 3(c, d) for the DBLP and TreeBank databases respectively. In these examples, the query returns the title of a PhD thesis identified by the key 'phd/White94' in the first case while in the second case the query returns the textual contents of the element 'NNS' that is located under the path '/FILE/EMPTY/ADJP/PP/NP/VP/PP/PP/NP' where the value of the child element 'JJ' is "Had5yxWr+9m+82oVrszXXx==".

4 A NODE-BASED SCALING ALGORITHM (NBS)

The algorithm in Fig 4 was used by the 3D XML benchmark to reduce the size of the dataset members while keeping the underlying XML schema unchanged. The algorithm can also be used to add more XML databases to the base dataset. Because XML databases can be added from any source (synthetic or real), there will be conflicts in the size

in terms of number of nodes. Therefore, the algorithm is first used to bring the size of the new databases to the scale of the existing dataset members. Secondly, the algorithm is used to reproduce two more versions of each new database.

The algorithm works as follows. Based on the scaling-percentage "s", the algorithm selects "s%" of the node-sets of every path in the XML tree. The sub-tree selected may then contain some dangling-nodes (i.e. separated from the root node). The ancestors of these dangling-nodes are then added to the sub-tree. If the number of nodes in the new sub-tree is outside acceptable limits around "s%", the scaling-percentage "s" is tuned to increase or decrease the size of the sub-tree produced.

Fig 5 depicts the experimental results from a test of the node-based scaling algorithm. Firstly, five XML documents were produced by the XMark (Schmidt et al. 2002) generator at scaling-factor (SF) of 1.0, 0.8, 0.6, 0.4 and 0.2. The same number of documents were produced using the new algorithm which scales the base XML documents (produced by XMark at SF=1.0) at 80%, 60%, 40% and 20%. The other five XML documents were also produced by the new algorithm but using the tuning mechanism. The difference between the numbers of nodes for similar documents was calculated and is presented in the graph.

The experiment shows that the number of nodes in the XML documents generated using the un-tuned node-based scaling algorithms is usually less than the number of nodes in the XMark's documents. However, by tuning the scaling-percentage, the size of the documents is always within acceptable limits.

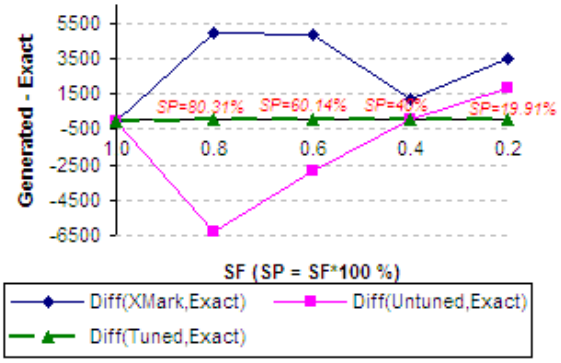

Fig 5: A Comparison between the document-size of XML documents generated by XMark and NBS algorithm

Tab 3: Experiments' Query-set

Q1: Shallow Exact Matching
Q1B: Deep Exact Matching
Q2: Order Access
Q3: RPE using '*'
Q3B: RPE using '//'
Q4: Joins on Values
Q5: Path Traversal
Q6: Missing Elements

5 A PERFORMANCE STUDY

5.1 Experiment Setup

To measure the impact of varying the database's dimensions on the XML querying process, an experimental study was conducted to compare the relative performance of two well-known mapping techniques (Edge (Floescu and Kossmann 1999) and XParent (Jiang et al. 2002) using the 3D XML Benchmark. The experiment compares the execution-time of *eight* queries that were executed over the benchmark's base databases. The discussion of other experimental metrics (e.g. CPU usage) as well as the results over the reduced database versions is omitted from this paper due the space limitation. Finally, a short description of the queries used is given Tab 3 while results are discussed in the following section.

5.2 Experimental Results

The result presented in Fig 7 show that the performance of any XML technique was not always consistent over the three XML databases for some query-classes. In the "Shallow Exact Matching" for example, XParent technique performed better than Edge technique over the "DBLP100" and "XMark100" databases but not the "TREE100" database. The inconsistency was also found in the "RPE using * Operator" query, the "RPE using // Operator" query and the "Joins on Values" query.

6 CONCLUSION

This paper introduced a new test-model called the 3D XML benchmark which addresses several limitations found in the existing XML benchmarks by extending their datasets and query-sets. Rather than relying on synthetic datasets, the extended benchmark uses a combination of real and code-generated XML databases to encounter more XML structures that are inherited from both XML sources such as a clear variety in the depth and the breadth of the XML databases and wider range of irregular XML structures. In addition, the extended model

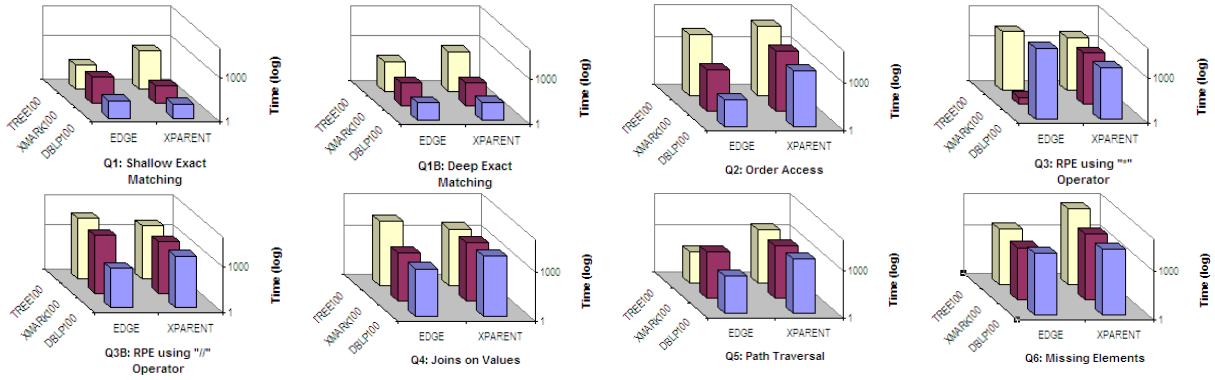

Fig 7: Experimental Results Representation

provides a more realistic environment for testing the scalability of XML implementations by preserving the XML schema of the databases' versions used. Unlike other XML scaling algorithms which, for example, discard some elements' classes or regenerate a fresh database in order to reduce its size, the 3D test-model uses a node-based scaling algorithm to proportionally reduce the fanouts at all tree levels.

The experimental results (discussed in this paper) have shown that some XML techniques, which were tested against the new benchmark's dataset, have performed inconsistently over different database's classes for some query-set members.

REFERENCES

Berglund, A., Boag, S., Chamberlin, D., Fernández, M., Kay, M., Robie, J., and Siméon, J. (2007) XML Path Language (XPath) 2.0, [Online] Avail: http://www.w3.org/TR/xpath20/ [30/08/2009].

Boag, S., Chamberlin, D., Fernández, M., Florescu, D., Robie, J., and Siméon, J. (2007) XQuery 1.0: An XML Query Language [Online] Avail: http://www.w3.org/TR/xquery/ [30/08/2009].

Böhme, T. and Rahm, E., (2003) 'Multi-User Evalaution of XML Data Management Systems with XMach~1', *EEXTT'02*, pp 148-159.

Carey, M., DeWitt, D. and Naughton, J. (1993), 'The OO7 Benchmark', *ACM/SIGMOD Record*, 22(2), pp 12-21.

Cohen, S. (2008) 'Generating XML Structure Using Examples and Constraints', PVLDB'08, pp 490-501.

Florescu, D., and Kossmann, D. (1999) 'A Performance Evaluation of alternative Mapping Schemas for Storing XML Data in a Relational Database', *TR:3680, May 1999, INRIA, Rocquencourt, France.*

Harder, T., Haustein, M., Mathis, C., and Wagner, M. (2007) 'Node Labelling Schemes for Dynamic XML Documents Reconsidered'. *Int'l Journal of DKE, 60(1),* pp 126-149.

Jiang, H., Lu, H., Wang, W., and Yu, J. (2002) 'XParent: An Efficient RDBMS-Based XML Database System', *ICDE'02, CA, USA*, pp 1-2.

Li, Y., Bressan, S., Dobbie, G., Lacroix, Z., Lee, M., Nambiar, U. and Wadhwa, B. (2001) 'XOO7: Applying OO7 Benchmark to XML Query Processing Tool', *ACM/CIKM., Atlanta, USA*, pp 167-173.

Liefke, H., and Suciu, D. (2000) 'XMill: an efficient compressor for XML data', *ACM/SIGMOD'00,* pp 153-164.

Lu, H., Yu, J., Wang, G., Zheng, S., Jiang, H., Yu, G. and Zhou, A. (2005) 'What Makes the Differences: Benchmarking XML Database Implementations', *ACM/IT., 5(1), NY, USA*, pp 154-194.

Miklau, G. (2009) UW XMLData Repository [Online] Avail: http://www.cs.washington.edu/research/xmldatasets/ [15/06/2009].

Mlynkova, I. (2008) 'XML Benchmarking', IADIS'08, pp 59-66.

Nicola, M., Kogan, I.., and Schiefer, B. (2007) 'An XML Transaction Processing Benchmark', ACM/SIGMOD'07, pp 937-948.

PennProj. (1999) The Penn Treebank Project [Online] Avail: http://www.cis.upenn.edu/~treebank/ [21/06/2009].

Runapongsa, K., Patel, M., Jagadish, H., Chen, Y., and Al-Khalifa. S. (2006) 'The Michigan Benchmark: Towards XML Query Performance Diagnostics', *Int'l Journal of IS, 31(2),* pp 73-97.

Schmidt, A., Waas, F., Kersten, M., Carey, D., Manolescu, I., and Busse. R. (2002) 'XMark: A Benchmark for XML Data Management', *VLDB'02, Hong Kong, China*, pp 1-12.

Schmidt, A., Waas, F., Kersten, M., Florescu, D., Carey, M., Manolescu, J. and Busse, R. (2001), 'Why and How to Benchmark XML Databases', *ACM/SIGMOD'01*, CA, USA, pp 27-32.

Yao, B. Özeu, M., and Khandelwal, N., (2004) 'XBench Benchmark and Performance Testing of XML DBMSs', *ICDE'04*, pp 621-632.

Wu, Y., Lele, N., Aroskar, R., Chinnusamy, S., and Brenes, S. (2009) 'XQGen-An Algebra-based XPath Query Generator for Micro-Benchmarking', ACM/CIKM'09, pp 2109-2110.