An Analysis of Relational Storage Strategies for Partially Structured XML

Yasser Abdel Kader[1], Barry Eaglestone[2], Siobhán North[1]
[1] Department of Computer Science
[2]Department of Information Studies
University of Sheffield, Sheffield, UK
Y.AbdelKader@dcs.shef.ac.uk, {B.Eaglestone, S.North}@sheffield.ac.uk

Keywords: XML databases, structure mapping, performance analysis

Abstract: This paper presents a performance analysis of strategies for storing XML data sets in relational databases, focusing on XML datasets that are a combination of structured and semi-structured data. The analysis demonstrates advantages of a hybrid approach combining structure mapping and XML data type instances. However problems remain with current technology with regards to scaling of the approach for large data sets. Also, anomalous results are identified and a threshold at which the cost of data shredding out weighs the advantages of structure mapping.

1 INTRODUCTION

The research reported is an exploration of performance implications of approaches to utilizing SQL/XML (ISO/IEC, 2006), focusing on datasets that comprise both structured and semi-structured data. Compromises between database complexity and increased size which are a consequence of structure mapping, and structural simplicity and compactness achieved by storing data as instances of XML types were investigated. Results show a hybrid approach that combines both to be advantageous, but reveal performance anomalies, problems of scaling, and thresholds at which cost of data shredding outweigh advantages of structure mapping.

2. EXPERIMENTAL DESIGN

Our hypothesis was - For the class of *partially-structured* XML documents, i.e. containing both a prescribed structured part and a semi-structured part, performance enhancement may be achieved over existing query processing techniques for semi-structured data by using relational database query processing and optimisation to exploit pre-knowledge of the prescribed structured part. Accordingly, experiments analyzed performance of a hybrid strorage model which, respectively, represents structured and semi-structured data using structure mapping and XML data types. Structure mapping (Shanmugasundaram et al, 1999; Yoshikawa & Amagasa, 2001; Balmin & Papakonstantinou, 2005; Pal et al, 2006) represents an XML document as a relational table. Other tables represent its nested tagged elements. Lowest-level tagged elements are stored as attributes. This suits highly structured document collections that conform to a limited number of document structures which are static over time (Lu et al, 2005; Yoshikawa & Amagasa, 2001).

We implmented structure mapping manually, in lieu of effective algorithms for stucture mapping partially-structured data. Also, a manual approach better suits partially-structured XML data by giving designers flexibility to apply structure mapping only for data deemed to be sufficiently well structured. This is not true of alternative approaches, i.e., data mining to extract structure from XML documents (Deutsch, Fernandez & Suciu, 1999), and DTD or XML Schema analysis techniques (Lv & Yan, 2006; Penna et al, 2006), such as inlining (Shanmugasundaram et al, 1999; Lu et al, 2003; Atay et al, 2007). In our manual structure mapping, in addition to devising a relational representation of the hierarchical organization of the structured component of the data, we ensured optimal querying of this part by creating sufficient indices.

We used XML data types (Krishnaprasad et al, 2005; Murthy et al, 2005; Rys, 2005; Pal et al, 2005, 2006; Ozcan et al, 2006; ISO/IEC, 2003, 2006; Eisenberg & Melton, 2004) within our hybrid model to represent the semi-structured part because structural irregularities can cause structure mapping to generate large complex schemas, and schema inflexibility becomes problematic for structurally volatile XML data. Other problems avoided by using XML data types are the need to resolve naming and type ambiguities and contentions, and, more generally, XML structural directives, such as Or (‘|’), cannot be mapped naturally into the relational model (Yoshikawa & Amagasa, 2001). XML data types support flexible querying of semi-structured data through path-based and regular expression-based querying facilities supported by SQL/XML. Also, in practice XML database designers will mainly utilize XML types within relational databases for storing XML data sets, given the current immaturity and relatively poor performance of native XML database systems (Lu et al, 2003, Grust at al, 2007).

2.1 Experiment Setup

Experiments were run on a single machine environment: Intel ® Pentium ® CoreTM Duo processor T2250, 1.73 Ghz; 1536 MB Ram; 120 GB Hard Disk Drive; Microsoft Windows XP Professional 5.1.2600 service pack 2; Microsoft SQL Server 2005. Transact-SQL was used as the main language to access the SQL Server. The decision to use MS XML Server was pragmatic. It was available, widely used internationally, and is a representative example of SQL/XML technology.

SQL Server 2005 implements features of SQL/XML, including XML data types, indexing, full-text XML search, as well as XQuery and XPath. XML data can be stored as instances of un-typed or typed XML data fields (MS SQL Server, 2005; Pal et al, 2005, 2006; Rys, 2005; Lacoude, 2006). Using un-typed XML data types, there is no XML schema, so SQL Server only checks that XML data is well formed. Instances of a typed XML data type must conform to an XML schema, which is used to validate the data, perform type checks, and optimize storage and query processing. Results, not discussed here, showed a poor performance for un-typed XML representations. Therefore results presented here cover only experiments using typed XML data fields.

3.2 XML Data Set

The dataset used in the experiments was the Digital Bibliography & Library Project (DBLP) (http://dblp.uni-trier.de/xml/), which comprises computer science-related bibliographic information. This XML document references over 750,000 publications by 450,000 authors, stored in 335 Megabyte, as of September 2006 (Reuther et al, 2006; Ley & Reuther, 2006). Publications are mainly conference papers (60%) and journal articles (37%). This dataset is wide use in XML database research, thus allowing comparability of results. Also, the data conforms to a DTD from which it was possible to derive structure for all or part of the dataset, allowing flexible interpretations of the document to simulate varying ratios of semi-structured to highly-structured content. The use of “natural”, rather than synthesized data sets, adds to the validity of the study, but a limitation is our inability to vary the inherent structured-ness of the data set itself. This requires multiple data sets, possibly synthesized, which we advocate as further work.

2.3 Benchmark

The XBench benchmark (Yao, Ozsu & Keenleysidem, 2004) was used because of its comprehensive range of use cases for XQuery, and flexibility to deal with multiple XML scenarios, including data- and text-centric, single and multiple documents. This contrasts with single scenario benchmarks, such as TPC-C, TPC-H, TPC-R, TPC-W (Transaction Processing Performance Council (http://www.tpc.org)), Wisconsin benchmark (DeWitt, 1993), and XML application benchmarks, such as XMach-1 (Bohme & Rahm, 2002), XMark (Schmidt et al, 2002), X007 (Brassan et al, 2002), XPathMark (Franceschet, M. 2005) and TPoz (Nicola et al, 2007). We also rejected micro benchmarks, such as Michigan Benchmark (Runapongsa et al, 2006) and MemBer (Afanasiev et al, 2005), since they evaluate at too small a level of query granularity. The XBench query sets, as adapted for the DBLP data set, are listed in table 1.

2.4 Storage Strategies

The hybrid storage model and its two base models, i.e. structure mapping (S/M) and XML data types (XML), were evaluated. Five implementations of the hybrid model were created to evaluate the impact of varying the ratio of semi-structured to structured data with respect to two dimensions. The vertical dimension, so called because of the conventional tabular representation of data, is the ratio of semi-structured to structured components of the schema. The horizontal dimension is the ratio of semi-structured to structured data instances. The vertical dimension was varied in the first (D), second (DA) and third (DT) implementations, respectively, by structure mapping document key, document key and author(s), and document key and title. In the other implementations (60%X and 37%X), approximately 60% and 37% of data elements were represented as typed XML data fields and the rest represented using structure mapping, thus varying the horizontal ratios.

Experiments were conducted on different sized databases by selectively removing content: the whole DBLP data set (DB3/3); two-third of the data set (one third of in-proceedings references and article references deleted) (DB2/3); and one third of the data set (two third of the in-proceedings and articles referenced deleted) (DB1/3).

2.5 Performance Metrics

Performance was measured in milliseconds, in terms of execution time, CPU busy time and IO busy time. Also, the number of IO reads and writes were recorded for each query. To establish stability, each query was run 20 times and performance was computed as the average (excluding maximum and minimum execution times). Standard deviations were calculated for each query and storage model, to identify any instability.

3. EXPERIMENT RESULTS AND ANALYSIS

Table 1 presents a ranking of the storage strategies for each category of query in the benchmark when executed with the largest of our datasets (DB3/3) (see (Abdel Kader, 2007) for full details). Though far from exhaustive, these suggest interesting relative characteristics of the storage strategies. None performed well for all query groups. For example, though *D* performed well accessing via the document key (Qs 1,9,12-16,19), it was relatively poor for querying author names (Qs 2-7,10-11). In the latter group, the *DA* model performed best, possibly demonstrating relative efficiency of querying conventional atomic attributes via indices, in contrast to using typed XML instances. However, relatively poor performance of *S/M* across these twenty queries illustrates advantages in storing data not directly utilized as query keywords as typed XML instances, i.e. the hybrid approach. Examples of the superiority of the hybrid model over pure structure mapping include quantification (Q6-7) and retrieval of individual documents (Q16). This is probably due to the fact that the hybrid model reduces data shreding, thus reducing the overhead of multiple join operations to re-assemble a document. However, as a counter example, though three of the queries in which article titles are queried (Qs 8,17) performed well with *DT*, this was not the case for the fourth (Q20), in which the length of the title is tested, where best performance was achieved using *S/M*. Also, storing the whole XML data as a typed XML instance (*XML*) always produced poorer performance than strategies which dividing data into smaller XML elements. In a number of queries, the XML model showed good performance, but in these cases so did the other hybrid models (e.g., quantification (Q6-7) and text search (Q17)).

Thus, messages are mixed. However, if we consider performance of specific strategies the results give a clearer view.

Pure structure mapping (*S/M*): In most instances this produced poor performance relative to hybrid approaches. Exceptions were Q2, Q8, Q18 and Q20. This was unexpected, since the BDLP dataset is relatively well structured and therefore well suited to structure mapping. Experiments with different dataset sizes demonstrated near linear deterioration in query time as size increased, with average deterioration of 59% to 125% respectively, as size doubled and tripled. This representation seems particularly ill-suited for irregular data (Q14), with a possibly exponential deteriorated from 300% to 1053% as size doubles and triples. Thus, dataset size and application characteristics are important considerations when contemplating this strategy.

Pure use of typed XML data types (*XML*): As with pure structure mapping, this approach produced poor performance relative to the hybrid approaches. Exceptions are Q7 (universal quantification) and Q20 (datatype casting). However, performance was poor for all conventional relational-style retrievals involving selection, projection and join. Deterioration in query performance was more extreme than for *S/M*. On average, deterioration was respectively 121% and 345% as the dataset size doubles and triples. The worst case was for the path expression query, Q8, where deterioration was from 700% to 1264%. Thus, average performance was poor and deterioration as size increases is worse than linear, suggesting viability only for small data sets.

Table 1: Summary of all the relative performance results ranging from 1 (best) to 6 (worst).

Query Category	Query	*S/M*	*XML*	Hybrid				
				37%X	*60%X*	*D*	*DA*	*DT*
Exact match	Q1: Return titles of article with key value X.	4	5	1	3	2	-	-
	Q2: Return titles of articles authored by X.	2	6	3	4	5	1	-
Function application	Q3: Count articles authored by author X	5	6	3	2	4	1	-
Ordered access	Q4: Return titles of articles by author X., retaining their relative ordering in the BDPL data set.	6	5	4	2	3	1	-
	Q5: Return title of first article authored by X, listed within the BDPL data set.	5	6	2	2	4	1	-
Quantification	Q6: Return titles of articles authored either by author X and/or author Y	6	5	2	4	3	1	-
	Q7: Return titles of articles by both author X and author Y	6	2	5	4	3	1	-
Path expressions	Q8: Return article titles containing “XYZ”	1	5	3	4	6	-	2
	Q9: Return authors of articles that have a key value X. (The symbol ‘//’ to select subpaths in the document that match address an article key.)	4	5	1	3	2	-	-
Sorting	Q10: List the title, publication date and authors of all articles, sorted by title for specific authors.	5	6	3	4	2	1	-
	Q11: List the title, publication date and authors of all articles, sorted by publication date for specific author.	5	6	3	4	2	1	-
Document construction structure	Q12: List title, publication date and authors for a specific article, preserving original document structure.	4	5	2	3	1	-	-
	Q13: List title, publication date and authors for a specific article, transforming document structure.	4	5	2	3	1	-	-
Irregular data	Q14: List the title, publication date and authors of all articles in which the ‘ee’ element is missing.	4	5	3	2	1	-	-
	Q15: List title, publication date and authors of articles in which the ‘year’ element has a specific value.	5	4	3	2	1	-	-
Retrieval of individual documents	Q16: Retrieve article data that has key value X, keeping its original structure.	5	4	2	3	1	-	-
Text search	Q17: Search for the word XYZ in any field in the article data.	5	4	2	3	6	-	1
	Q18: Search for the phrase XX YY ZZ in any field in the article data.	2	5	3	4	6	-	1
References and Joins	Q19: Retrieve the first author of the article with key value X, and use the author’s name to return the tiles of all of their publication.	4	5	2	3	1	-	-
Datatype casting	Q20: Returns the titles of articles, where the word count is longer than a specific size.	1	2	5	6	4	-	3

Vertical hybrid approaches (*D, DA, DT*): Part of the XML schema common to all repeating instances is structure mapped and the rest is represented as typed XML instances. As expected, this outperformed the two pure base approaches where query keywords were within the structure mapped part. However, this superiority was surprising for text searches (Q17 and 18) where all fields were accessed. Also, we anticipated performance would continue to improve as the ratio of structured to semi-structured data increased, since conventional relation querying was likely to outperform the added XML facilities. Given that the BDLP dataset is mainly well structured, if this was true, the best performances would be for *S/M*. This was not the case, since there seems to be a threshold beyond which performance deteriorated. Mainly, query performance deterioration with increase in dataset size was near linear. For *D*, *DA* and *DT* deterioration was respectively from 355% to 769%, 25% to 63%, and 2801% to 4269% as size doubled and tripled. *DA*'s worst deterioration was for quantification (Q6), where increase in query time was from 126% to 290%, as the size doubled and tripled. *DT* deteriorated most for data type casting (Q20), from 10509% to 15880% as size doubled and tripled. *D* has two step changes with respect to query performance. Theses were also for data type casting (Q20), with a deterioration of 2204% when the size doubled, and for text searching (Q17), where there was a 3969% deterioration when size tripled. Thus, though our hypothesis is largely born out by the results, there are other factors which any decision model must take into account, including overheads incurred by data shredding, and the impact of dataset size.

Horizontal hybrid approaches (*37%X*, *60%X*): Some types of repeating instances are structure mapped while others are stored as typed XML instances. This produces mainly a middle ranking performance. Neither *37%X* nor *60%X* consistently outperforms the other, but, respectively, in 90% and 75% of cases both outperformed *S/M* and *XML*. Thus, there seems to be an advantage in horizontally partitioning data into structured and semi-structured representation, as well as more obvious benefits of the vertical approach. There is also a worrying possibly exponential deterioration, as dataset size increased. Both *37%X* and *60%X* exhibited similar average deterioration, from 27% to 395% and from 29% to 445%, as size doubled and tripled. However, for the path expression query, Q8, deterioration in performance was from 166% to 2205%. Thus, this approach mainly improves on pure structure mapping (*S/M*), but does not appear to scale to very large datasets.

Finally, we note that the above results complement recent and more specific analyses of performance advantages of using "off-the-shelf" relational technology to store and query XML data sets. (Torsten et al, 2007) and (Gou &Chirkova, 2007) respectively analyse XPath processing and the related more general problem of twig pattern matching. As in our study, both explore the use of native relational facilities with XML datasets. Torsten *et al* describe and evaluate partitioned B-trees for non-recursive XPath axis evaluation; aggregation functions for pruning in tree join algorithms; and the effectiveness of relational query optimisation rewrite techniques for tree structured data. Gou and Chirkova survey relational and native XML techniques, concluding that a good trade-off can be achieved with a relational inverted list representation of XML data, complemented by efficient XML native join algorithms. Thus, their studies consider techniques that may be deployed within an XML type to improve querying performance. Consequently, we anticipate that the trade-offs we have explored between structure mapping and XML type instances will evolve dynamically with improvements in the relational/XML technology.

4. CONCLUSIONS

This paper has presented a performance analysis of relational storage strategies for partially-structured data. The hybrid approach, combining structure mapping and XML data types, was show to have query performance advantages over pure structure mapping and sole use of XML data types. However, results are inconclusive, since they identify a anomalies, problems of scaling, and the existence of thresholds where cost of data shredding appears to outweigh advantages of utilizing relation query processing. Each of these is a motivation for further research. Also, the experiments described are limited, as has been discussed in the body of this paper, and further experimental work is needed.

Our contribution is an analysis of relative performances within a specific configuration, rather than across systems, as in other performance studies. Also, we are not aware of other studies of partially-structured data and the impact of the horizontal and vertical dimensions of data structure-ness

Results presented here are part of a more intensive investigation (Abdel Kader, 2007). In particular, results have enabled us to devise a heuristics-based model to inform XML/relational design which we plan to validate and elaborate as further research.

REFERENCES

Abdel Kader, Y. 2007. A Performance Analysis of a Hybrid Relational-Xml Approach to store Partially-Structured Data. *PhD Thesis, University of Sheffield.*

Afanasiev, L. Manolescu, I., Michiels, P. 2005. *MemBer: A Micro-benchmark Repository for XQuery*. XML Symposium (XSym).

Atay, M., et al. 2007. Efficient schema-based XML-to-Relational data mapping. *Information Systems* 32(3), pp.458- 476.

Balmin, A., Papakonstantinou, Y 2005. Storing and Querying XML data using denormalized relational databases *The VLDB Journal*, 14, pp.30-49.

Böhme, T., Rahm, E. 2002. Multi-user Evaluation of XML Data Management Systems with XMach-1. *Efficiency and Effectiveness of XML Tools, and Techniques EEXTT*, pp.148-158.

Brassan, S., et al. 2002. The XOO7 benchmark. In *Proceedings of VLDB'02 Workshop Efficiency and Effectiveness of XML Tools, and Techniques EEXTT*, LNCS 2590, pp.146-147.

Deutsch, A., Fernandez, M., Suciu, D. 1999. Storing semistructured data with STORED. In *Proceedings of the 25th ACM SIGMOD International Conference on Management of Data*.

DeWitt, D. 1993. The Wisconsin Benchmark: Past, Present, and Future. *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann, ISBN 1-55860-292-5.

Eisenberg, A., Melton, J. 2004. Advancements in SQL/XML. SIGMOD Record 33(3), pp.79-86.

Franceschet, M. 2005. *XPathMark – An XPath benchmark for XMark Generated Data.* International XML Database Symposium (XSYM), pp. 129-143.

Gou, G., Chirkova, R. 2007. Efficiently Querying Large XML Data Repositories: A Survey. IEE Transactions on Knowledge and Data Engineering. 19(10), October, 2007, pp.1381-1403.

Krishnaprasad, et al. 2005. Towards an Industrial Strength SQL/XML Infrastructure. *Proceedings of the 21st International Conference on Data Engineering, ICDE* 2005. Tokyo, Japan, pp.991-1000.

Lacoude, P. 2006. *Pushing SQL Server 2005 Limits, Dealing with Oversized XML Documents* [online] Available from: http://www.lacoude.com/Docs/public /public.aspx?doc=SQL90XML.PDF [Accessed 17.10.2007]

Ley, M., Reuther, P. 2006. Maintaining an Online Bibliographical Database: The Problem of Data Quality. *EGC 2006*, Lille, France, pp.5-10.

Lu, H. et al., et al. 2005. What Makes the Differences: Benchmarking XML Database Implementations. *ACM Transactions on Internet Technology (ACM TOIT)*, 5 (1), pp.154-194.

Lu, S., et al. 2003. A new inlining algorithm for mapping XML DTDs to relational schemas. In *Proceedings of the 1st International Workshop on XML Schema and Data Management*. LNCS, Chicago, Illinois, USA.

Lv, T., Yan, P. 2006. Mapping DTDs to relational schemas with semantic constraints. *Information and Software Technology*, Volume 48 (4), pp. 245-252

Murthy, R. et al. 2005. Towards an enterprise XML architecture. In *Proceedings of the 2005 ACM SIGMOD international Conference on Management of Data*, Baltimore, Maryland, pp.14-16.

Nicola, M., Kogan, I., Schiefer, B. 2007. *An XML Transaction Processing Benchmark.* SIGMOD, Beijing, China.

Özcan, F., et al. 2006. Integration of SQL and XQuery in IBM DB2. *IBM System Journal*. 45 (2), pp.245-270.

Pal, S. et al. 2005. XQuery implementation in a relational database system. In *Proceedings of the 31st international Conference on Very Large Data Bases*. Trondheim, Norway, pp. 1175-1186.

Pal, S., Tomic, D., Berg, B., Xavier, J. 2006. Managing Collections of XML Schemas in Microsoft SQL Server 2005. *EDBT 2006*, pp. 1102-1105.

Penna, G. et al. 2006. Interoperability mapping from XML schemas to ER diagrams. *Data & Knowledge Engineering*, 59 (1), pp.166-188.

Reuther, P. et al. 2006. Managing the Quality of Person Names in DBLP. *Research and Advanced Technology for Digital Libraries, 10th European Conference, ECDL 2006*, Alicante, Spain, pp.508-511.

Runapongsa, K. et al. 2006. The Michigan benchmark: towards XML query performance diagnostics. *Information Systems* 31(2), pp.73-97.

Rys, M. 2005. XML and relational database management systems: inside Microsoft® SQL Server™ 2005. In Proceedings of the 2005 *ACM SIGMOD International Conference on Management of Data*, Baltimore, Maryland, pp.14-16.

Schmidt, A., et al. 2002. XMark: A benchmark for XML data management. In *Proceedings of the 28th International Conference on VLDB*. Hong Kong, China, pp.974-985.

Shanmugasundaram et al. 1999. Relational Databases for Querying XML Documents: Limitations and Opportunities. *Proceeding of the 25th VLDB Conference*, Edinburgh, Scotland.

MS SQL Server 2005 [online] Available from: http:// www.microsoft.com/sql/default.mspx [Accessed 17.10.2007].

SQL:2006. International Organization for Standardization (ISO). *Information Technology-Database Language SQL. Standard No. ISO/IEC 9075-14:2006. Part 14: XML-Related Specifications (SQL/XML)* (Available from American National Standards Institute, New York, NY 10036)

XQuery 1.0 and XPath 2.0 Data Model (XDM) W3C Candidate Recommendation 11 July 2006 [online]. Available from: http://www.w3.org/TR/xpath-datamodel/ [Accessed 07.11.2006]

Yao, B., Özsu, M. and Keenleysidem J. 2004. XBench Benchmark and Performance Testing of XML DBMSs. In *Proceedings of 20th International Conference on Data Engineering*, Boston, MA, United States of America, pp.621-632.

Yoshikawa, M. and Amagasa, T. 2001. Xrel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases. *ACM Transaction on Internet Technology* 1 (1), pp.110-141.