XOO7: Applying OO7 Benchmark to XML Query Processing Tools

Stéphane Bressan,
Mong Li Lee,
Ying Guang Li
Bimlesh Wadhwa
National University of Singapore
School of Computing
10 Kent Ridge Cresent, Singapore.
65-874-2905.
{steph, leeml, liyinggu, bimlesh}@comp.nus.edu.sg

Gillian Dobbie
Department of Computer Science
The University of Auckland
Auckland, New Zealand.
64-9-373-7453
gill@cs.auckland.ac.nz

Zoé Lacroix,
Ullas Nambiar
Arizona State University
PO Box 876106
Tempe AZ 85287-6106
USA
1-480-727-6935
{zoe.lacroix, mallu}@asu.edu

ABSTRACT

If XML is to play the critical role of the lingua franca for Internet data interchange that many predict, it is necessary to start designing and adopting benchmarks allowing the comparative performance analysis of the tools being developed and proposed. The effectiveness of existing XML query languages has been studied by many who focused on the comparison of linguistic features, implicitly reflecting the fact that most XML tools exist only on paper. In this paper, with a focus on efficiency and concreteness, we propose a pragmatic first step toward the systematic benchmarking of XML query processing platforms with an initial focus on the data (versus document) point of view. We propose XOO7, an XML version of the OO7 benchmark. We discuss the applicability of XOO7, its strengths, limitations and the extensions we are considering. We illustrate its use by presenting and discussing the performance comparison against XOO7 of three different query processing platforms for XML.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness)

General Terms

Measurement, Performance, Experimentation, Standardization.

Keywords

XOO7, XML Management Systems, XML Benchmarks, Native-XML database, XML aware database.

1. INTRODUCTION

It is becoming increasingly important to effectively and efficiently manage XML data. In particular, we expect new Web based applications for e-commerce to require XML query processing facilities. Introduced as a schema-less, self-describing data representation language, XML quickly emerged as the standard for information interchange for the Web [30]. The development of XML was not furthered directly by the mainstream database community, yet database researchers actively participated in developing standards centered on XML, and particularly query languages for XML. Many XML query languages have been proposed but only few query-processing tools are available for use. The languages and tools can be classified into two groups – those designed with a document focus e.g. XQL [23], Quilt [20] and KWEELT [27], and those designed with a database focus e.g. LORE [5] and XML-QL [12]. Recently, XQuery [33] has been drafted as the query language for XML, combining both document and data centric orientation of XML. At this juncture a user intending to setup a XML based data interchange or storage system would be faced with the question of which XML query languages to base her system on. With so many proposals and tools, end-users need better insight as to which one is most suitable in terms of features and performance for their application requirements. Several papers have compared the features of these XML query languages [15,8] but none have provided a performance evaluation.

In this paper, we propose XOO7 – a benchmark to evaluate the performance of XML query processing tools. XOO7 is an adaptation of the OO7 benchmark [10]. OO7 provides a comprehensive evaluation of object-oriented database management system (OODBMS) performance. The main OODBMS's and storage managers have been benchmarked against OO7: E/Exodus, Objectivity/DB, and Ontos. The rationale underlying both the design of XML, XML query languages, and the object-oriented data model and query languages is the need for richer structure for the flexible modeling and querying of complex data. Although XML also attempts to provide a framework for handling semi-structured data, it encompasses most of the modeling features of complex object models [3, 4]. This observation motivated our study. There are straightforward correspondences between the object-oriented schemas and instances and XML DTDs and data. We mapped the OO7 schema and instances into a DTD and the corresponding XML data sets. Our purpose here is to evaluate the performance of query processing facilities, therefore we translated the eight OO7 queries into the respective languages of the query processing tools we tested: LORE, a special-purpose (or semi structured) system university prototype; KWEELT, an open source university prototype that works on ASCII XML data files; and a commercial object-relational database system (OR-DBMS[1]) that provides a simple but limited mapping of XML data into object-relational data. The characteristics we measure are response time for different queries and classes of queries, time to load the data, and space required to store the data.

The rest of the paper is organized as follows. Section 2 addresses the expected functionalities of XML query languages. The design of a benchmark for XML queries is addressed in Section 3. The XOO7 data model and queries are defined in Section 4. Section 5 presents the preliminary performance results. Section 6 summarizes other related work and we conclude in Section 7 by highlighting the possible extensions to this work.

2. XML QUERY FUNCTIONALITIES

The performance of the implementation of query languages for XML depends strongly on their expressive power: the functionalities they provide. Indeed, some of the expected functionalities may affect significantly the efficiency of the system. Many languages claim to be XML query languages, however their functionalities vary dramatically. Some languages such as LOREL [5,16] provide the functionalities offered by a traditional data oriented query language such as SQL. Others focus on XML integration and restructuring with additional data-oriented functionalities such as join, nesting and aggregation as in XML-QL [31], or partial or none of these data-oriented functionalities as in XSL [32] and XQL [23]. More recently, languages such as Quilt [11] and XQuery [33] extend the data-oriented approach to functionalities to handle XML documents.

The design of a benchmark for XML query languages shall address the performance issues connected to the characteristics of XML query languages, thus their functionalities. XML query languages functionalities were addressed in a comparative analysis of XML query Languages [8] and listed as “must have” in the requirements [19] published by the W3C XML Query language working group. Table 1 enumerates all these requirements. An XML query language should support the manipulation and extraction of data from multiple documents (R1), by accessing and combining different parts within documents (R9), querying the DTD [XML:00], XML Schema [24, 25, 26] (R1) or along paths (R13), by using data types (R1) or evaluating conditions over textual elements (R5). XML queries should support implicit order (order of elements within the XML document) as well as explicit order (order defined in the schema) (R2). Complex Data models can be defined using the XML data model, in par with this, a XML query language should therefore be able to work with differing data models (R4) all of which would have a common origin. Since XML is a semi-structured language, NULL values may be present. A missing element may or may not be representable as NULL valued element but vice versa may be true, and hence NULL value manipulation will take on additional complexity (R7). Support for quantification and negation in queries (R6) is needed. XML can capture structured information and hence a XML query language should have the expressiveness of a structured query language like SQL for relational databases. Hence such a language should support various types of join operations (R9), aggregation (R10), sorting (R11). Unlike XML, relational model disregards the order. Hence sorting and aggregation increase in complexity when order and document structure need to be preserved in some form (R17). The language must be capable of generating new XML structures and transforming one XML structure to another (R18). Since queries can be along paths and paths can consist of recursive calls to themselves or sub paths, structural recursion should be supported (R20). A query on a database may change the underlying data. Hence the query language should provide methods for updating the underlying database (R15).

3. DESIGNING A BENCHMARK FOR XML QUERIES

The rationale underlying both the desing of XML, XML query languages and the object-oriented data model and query languages is the need for richer structure for the flexible modeling and querying of complex data. Although XML attempts to provide a framework for handling semi-structued data, it encompasses most of the modeling features of complex object models. There are straight forward correspondences between the object-oriented schemas and instances and XML DTDs and data. XOO7 was designed keeping in mind these similarities in data model of XML and object-oriented approach. XOO7 is an adaptation of OO7 Benchmark [10].

XML syntax is suited for semi-structured data. Yet XML and semistructured data have subtle differences [2]. A tree representation of XML and semi structured data is interchangeable but a graph structure of both models has differences. Semistructured data model is based on unordered collections, while XML is ordered. Unique identifiers can be associated with elements in XML. References to such elements can be made by other elements in the XML document. A close observation of XML model will show its similarity to the object-oriented data model. XML is probably most similar to object-oriented data model in as much as it also consists of nodes, and nodes can contain heterogeneous data. On the other hand, just how heterogeneous nodes are depends a lot on the particular

---

[1] We have chosen to withhold the name of the commercial system we have tested given the sensitivity of the results of the benchmark experiments.

DTDs or Schemas used to define the structure of an XML document. The object-oriented data model is similar to both XML and semi-structured data model with respect to representation of objects or entities using trees. Similar to XML we can assign object identities or 'oids' to objects if these have to be referenced by other objects. An object identifier can become part of a namespace and can reference other objects across the Web. This is similar to the notion of Namespaces in XML. In reality, XML is less natural in representing Relational databases (RDMBS). Individual tables can be directly represented literally, but with far more information about the data (i.e Metadata) than actual RDBMS's do. Similarly representing relational query results involving joins, grouping, sorting, etc. in XML is straightforward and is the most widely practiced use of XML in existing data management systems. But the core of an RDBMS is its relations. In particular, the set of constraints that exist between tables, and that are enforced by the RDBMS are what make RDBMS's so useful and powerful. It is surely possible to represent a constraint set in XML for purposes of communicating it, but XML has no inherent mechanism for enforcing constraints of this sort (DTDs and Schemas are constraints of a sort, but in a different and more limited way). A data model cannot be present without constraints or rather without the ability to enforce the constraints. Also characteristics of RDBMS like fixed record lengths, compact storage formats etc., designed to improve reliability and performance cannot be easily mimicked in XML. In fact XML can be viewed as an object model. The standard API for XML proposed by W3C called DOM uses the Document Object Model [13] for XML documents. The Resource Description Framework used for describing metadata for XML also has object-oriented flavour [21].

Table 1 Functionalities of XML Query Languages

Id	Description
R1	Query all data types and collections of possibly multiple XML documents.
R2	Allow data-oriented, document-oriented and mixed queries.
R3	Accept streaming data.
R4	Support operations on various data models.
R5	Allow conditions/constraints on text elements.
R6	Support for hierarchical and sequence queries.
R7	Manipulate NULL values.
R8	Support quantifiers (∃,∀, and ~) in queries.
R9	Allow queries that combine different parts of document(s).
R10	Support for aggregation.
R11	Able to generate sorted results.
R12	Support composition of operations.
R13	Allow navigation (reference traversals).
R14	Able to use environment information as part of queries e.g. current date, time etc.
R15	Able to support XML updates if data model allows.
R16	Support for type coercion.
R17	Preserve the structure of the documents.
R18	Transform and create XML structures.
R19	Support ID creation.
R20	Structural recursion.

Thus while developing the benchmark we based our decisions on two facts. First, the benchmark is for XML query systems using XML data and documents stored locally in files or database. Second, XML data model shows high degree of similarity to object-oriented model. Hence we decided to take OO7 – a benchmark designed to test performance of OOBDMBS and extend it to develop a benchmark for XML query processing systems. However, adaptations are needed if we want to use OO7 as a benchmark (refer to requirements of Table 1).

3.1 THE XOO7 BENCHMARK

XOO7 is an XML version of the OO7 Benchmark. Figure 1 shows the conceptual schema of the database modeled using the ER diagram given in the OO7 benchmark. We have translated this conceptual schema into the DTD shown in Figure 2. This translation involves some arbitrary choices, which are beyond the scope of this preliminary report. Nevertheless we outline our main decisions in the sequel of this section.

Table 2 XOO7 database parameters

Parameters	Small	Medium	Large
NumAtomicPerComp	20	200	200
NumConnPerAtomic	3, 6, 9	3, 6, 9	3, 6, 9
DocumentSize (bytes)	500	1000	1000
ManualSize (bytes)	2000	4000	4000
NumCompPerModule	50	50	50
NumAssmPerAssm	3	3	3
NumAssmLevels	5	5	5
NumComPerAssm	3	3	3
NumModules	1	1	10

Since XML does not cater for ISA relationships, we have pre-processed the inheritance of attributes and relationships. This transformation is common to many OO7 implementations. We choose the root of the XML document to be <Module>. There are three attributes in <Module>: MyID [2], type and buildDate. Each <Module> contains the elements <Manual> and <ComplexAssembly>. The element <ComplexAssembly> inherits the attributes of *Design Object*. Each assembly part has two integer attributes MyID and buildDate, and a string attribute type. Each <BaseAssembly> contains <CompositePart>. Each <CompositePart> has three attributes: MyID, type and buildDate, and three elements: <Document>, <AtomicPart> and <Connection>. The <Document> element has attributes MyID and title. Every <AtomicPart> has six attributes: MyID, type, buildDate, x, y and docId. Each <Connection> element has two attributes: type and length, and two sub-elements: <Part1> and <Part2>. Both <Part1> and <Part2> have an integer attribute IDREF. *Connection* is a recursive relationship. In XML, it can translate into an attribute of <AtomicPart>, or into an element at the same level as <AtomicPart> or at a level higher or lower than <AtomicPart>. We choose a lower level for our experiments on initial data sets. There are up-to seven levels of assemblies in the OO7 benchmark. We chose to use five levels in XOO7 because of the limitations of most existing XML tools in the volume of data they can manipulate. This is sometimes due to the naïve representation of tags (as ASCII) in many systems such as KWEELT.

---

[2] Since ID is a reserved word in XML, we have renamed it to `MyID`.

Similarly to OO7, XOO7 benchmark proposes three different databases of varying size: small, medium, and large. Table 2 summarizes the parameters and their corresponding values that are uses to control the size of the XML data. We have grouped the 8 OO7 queries, Q-1 to Q-8, into three groups as shown in Table 3. Group I involves lookups, Group II involves range queries, Group III is composed of join queries. To illustrate the concrete syntax of XML query languages, we give the code of Q-6 in KWEELT, Lorel for Lore, and SQL for the commercial OR-DBMS, respectively.

4. PERFORMANCE STUDY

We use XOO7 to evaluate three query processing platforms: Lore, KWEELT and OR-DBMS. The experiments are run on a SunOS 5.7 Unix system (333 MHz), with 256 MB RAM and 1.9 GB disk space. The C++ implementation of XOO7 is available at http://www.comp.nus.edu.sg/~ebh/XOO7.html.

Table 3 Queries in OO7

Group I	
Q-1	Exact match lookup. Generate 5 random numbers for AtomicPart's MyID. Return the AtomicPart's MyID according to the 5 numbers.
Q-4	Path lookup. Generate 5 random titles for Document. Return the Document's MyID according to the 5 titles.
Group II	
Q-2	Select 1% of AtomicPart (with a buildDate after 1990) and return their MyID.
Q-3	Select 10% of AtomicPart (with a buildDate after 1900) and return their MyID.
Q-7	Select all AtomicPart and return their MyID.
Group III	
Q-5	Single-level "make". Find the MyID of a CompositePart if it is more recent than the BaseAssembly it uses.
Q-6	Multi-level "make". Find the MyID of a CompositePart (recursively) if it is more recent than the BaseAssembly or the ComplexAssembly it uses.
Q-8	Ad hoc join. Join AtomicPart and Document on the docId of AtomicPart and the MyID of Document.

LORE, developed in Stanford University, is one of the earliest systems developed to store and query semi structured data. It has been extended at Stanford University to query XML data, and is implemented in C++. While LORE supports many needed features, it fails to support some important aggregate and update functions. KWEELT was designed and implemented at the University of Pennsylvania. It is written in Java and it is open-source. Its query language is based on Quilt, which in turn leverages the XPath standard.

KWEELT works from ASCII XML data files but can be interfaced to other storage back-ends. We have used it with ASCII XML data files. OR-DBMS is a commercial object-relational database management system. It is built on top of SQL and data in the object-relational database tables or views can be transformed into XML data. OR-DBMS provides a simple but limited mapping of XML data into object-relational data. We use XML-DBMS [6] to perform this mapping.

We record the space utilization for each of the systems for the various databases in the benchmark. The results are illustrated in Figure 3 for varying size of the input XML data. Each query is executed ten times and the average response time is recorded. The response time results are presented in Figure 4. Because of space limitations we present the results by groups of queries for the small and medium databases. The relatively bad performance of KWEELT can be explained by the fact that it accesses the ASCII XML data files. Regardless of the query, the performance degrades with the database (file) size. Group III involving path expressions and joins - Q-6 and Q-8, respectively - yield particularly bad performance. Lore is using a structured storage and implements access methods. The performance is consistent with the amount of data accessed by the query regardless of the overall database size. Only on path expression (Q-6) have we noticed a significant impact of the overall database size on the response time. We suspect that the path expression evaluation involves a systematic browsing of the data. The OR-DBMS leverages the query processing power of the relational database engine and yields the best response time. In Q-6, the path expression is implemented iteratively knowing there are exactly five levels. Notice finally that, in KWEELT, all the queries for a medium size database overflow the virtual memory and could not be executed. The storage requirements of KWEELT are equal to the size of the input ASCII XML data files. OR-DBMS takes advantage of the relational storage, economizing on the storage of the tags.

Table 4 Representation of Query 6 in 3 Systems

KWEELT	<result> FOR $ca IN Document("/home/hon/liyinggu/os/small91.xml")//ComplexAssembly, $ba IN $ca//BaseAssembly, $cp IN $ba/CompositePart [@buildDate.>.$ba/@buildDate OR @buildDate.>. $ca/@buildDate] RETURN $cp/@MyID </result>
Lorel for Lore	SELECT cp.MyID FROM Module(.ComplexAssembly)*ca, ca(.ComplexAssembly)*.BaseAssembly ba, ba.CompositePart cp WHERE ba.buildDate < cp.buildDate or ca.buildDate < cp.buildDate;
SQL for OR-DBMS	SELECT cp.MyID FROM COMPLEXASSEMBLY1 c1, COMPLESASSEMBLY2 c2, COMPLEXASSEMBLY3 c3, COMPLEXASSEMBLY4 c4, BASEASSEMBLY ba, COMPOSITEPART CP WHERE (cp.BUILDDATE > c1.BUILDATE and c1.MYID = c2.PARENTID and c2.MYID = c3.PARENTID and c3.MYID = c4.PARENTID and c4.MYID = ba.COMPLEXID and ba.MYID = cp.BASEID) or (cp.BUILDDATE > c2.BUILDDATE and c2.MYID = c3.PARENTID and c3.MYID = c4.PARENTID and c4.MYID = ba.COMPLEXID and ba.MYID = cp.BASEID) or (cp.BUILDDATE . c3.BUILDDATE and c3.MYID = c4.PARENTID and c4.MYID = ba.COMPLEXID and ba.MYID = cp.BASEID) or (cp.BUILDDATE > c4.BUILDDATE and c4.MYID = ba.COMPLEXID and ba.MYID = cp.BASEID) or (cp.BUILDDATE > ba.BUILDDATE and ba.MYID = cp.BASEID);

5. DISCUSSIONS AND RELATED WORK

Semistructured query languages and data models have been studied widely in [1][7]. In [14] several storage strategies and mapping schemes for XML data using a relational database are explored. Domain -specific database benchmarks for OLTP (TPC-C), decision support (TPC -H, TPC -R, APB -1), information retrieval, spatial data management (Sequoia) etc are available at [17], [29].

To our knowledge only two benchmarks, XMach -1 [9] and XMark [28], designed for XML, are publicly available. XMach-1 tests multi -user features. It evaluates standard and non -standard linguistic features such as insertion, deletion, querying URL, and aggregate operations. Although the proposed workload and queries are interesting, the benchmark has not been applied and no performance results exist. XMark is a very recent proposal to assess the performance of XML query processors. This benchmark consists of an application scenario which models an Internet auction site and 20 XQuery challenges designed to cover the essentials of XML query processing. These queries have been evaluated on an internal research prototype, Monet XML, to give a first baseline. Table 3 show the functionalities covered by queries given in XOO7. For queries of XMach -1 and XMark and functionalities they cover refer [34]. These benchmarks cover an average of 5 to 8 functionalities listed in Table 1. While the XMark benchmark 20 query challenges, both XOO7 and XMach -1 have 8 benchmarks queries. In additional, XMach -1 has 2 queries to test updates. We note that query Q8 in XMach -1 test several operations: count, sort, join and existential, making it hard to analyze the experiment result because it will not be clear which feature causes poor performance.

Table 5 Current XOO7 Queries

ID	Description	Coverage
Q1	Randomly generate 5 numbers in the range of AtomicPart's MyID. Return the AtomicPart's MyIDs according to the 5 numbers.	R1, R2
Q4	Randomly generate 5 titles for Documents. Return Document's MyIDs by lookup on these titles.	R1, R2
Q2	Select 1% of the latest AtomicParts via buildDate. Return the MyIDs.	R4
Q3	Select 10% of the latest AtomicParts via buildDate. Return the MyIDs.	R4
Q7	Select all of the AtomicParts and return the MyIDs.	R4, R8
Q5	Find the MyID of a CompositePart if it is later than the BaseAssembly it is using.	R1, R2
Q6	Find the MyID of CompositePart (repeatedly) once there is a BaseAssembly or ComplexAssembly it is using with a buildDate more than it is using.	R1, R2
Q8	Join AtomicParts and Documents on AtomicParts docID and Documents MyID.	R9

6. CONCLUSION

XML is becoming ubiquitous. Numerous of -the-shelf XML processing systems are becoming available. To check whether these systems truly harness the power of XML, XML related technologies like XPath, XPointer etc., and the XML query languages, a benchmark becomes inevitable. In this paper we first identify the desirable XML query characteristics. Next we show similarities between object-oriented data model and XML and propose XOO7, an XML version of the OO7 benchmark. This benchmark is a pragmatic first step toward the systematic benchmarking of XML query processing platforms. We illustrated its use by presenting and discussing the performance comparison against XOO7 of three query processing platforms for XML: LORE, KWEELT, and OR -DBMS. Against this benchmark, LORE and OR -DBMS consistently outperformed KWEELT. However, OR -DBMS and KWEELT were more economical with space. We are heartened by these results and will extend the benchmark in a number of directions. Given that XOO7 is an XML version of OO7, there is a possibility that XOO7 is currently biased towards systems that perform database features well and against systems that are optimised for information retrieval. As an initial extension we provide a set of queries shown in Table 6 to capture document -centric query processing capabilities of XML systems. While designing these queries, we assume a document ordered representation of XOO7 data. The complexity involved in satisfying this assumption on existing XML management systems has to be empirically evaluated and forms part of our future work. At the moment we assume single user systems. On the other hand multi -user systems are highly prevalent and widely used. We plan to extend XOO7 to include multi -user querying capabilities, querying in presence of schema information and other aspects of XML data like Navigation queries.

Table 6 New Queries Added to XOO7

ID	Description	Coverage
Q9	Randomly generate two phrases among all phrases in Documents. Select these documents containing 2 phrases.	R5
Q10	Repeat query Q1 but replace duplicated elements using IDREF.	R13
Q11	Select all BaseAssemblies from one XML database where it has the same "MyID" and "type" attributes as the other BaseAssemblies but with later buildDate.	R9
Q12	Select all AtomicParts with corresponding CompositeParts as their sub-elements.	R1, R2
Q13	Select all ComplexAssemblies with type "type008".	R1, R2

7. ACKNOWLEDGEMENT

This work is funded by the National University of Singapore

Academic Research Fund RP082112.

8. REFERENCE

[1] S. Abiteboul. Querying semistructured data. *In Proc. of Int. Conf. On Database Theory (ICDT)* , pp 1 -18, 1997.

[2] S. Abiteboul, P. Buneman, D. Suciu. Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufman Publishers, 2000.

[3] S. Abiteboul, S. Grumbach. COL: A Logic-Based Language for Complex Objects. *In Proc. of Int. Conf. On Extending Database Technology (EDBT)*, pp 271-293, 1988.

[4] S. Abiteboul, M. Scholl. From Simple to Sophisticate Languages for Complex Objects. *Data Engineering Bulletin* 11(3), pp 15-22, 1988.

[5] S. Abiteboul, D. Quass, J. McHug, J. Widom, J. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68, April 1997.

[6] R. Bourret. Java Packages for Transferring Data between XML Documents and Relational Databases. http://www.rpbourret.com/xmldbms/readme.htm.

[7] P. Buneman. Semistructured Data. *In Proc. of Symposium on Principles of Database Systems (PODS)*, pp 117-121, 1997.

[8] A. Bonifati, S. Ceri. Comparative Analysis of Five XML Query Languages. *ACM SIGMOD Record*, 29(1), 2000.

[9] T. Bohme, E. Rahm. XMach-1: A Benchmark for XML Data Management. Available at http://dbs.uni-leipzig.de/projekte/XML/XmlBenchmarking.html

[10] M. J. Carey, D. J. DeWitt, J. F. Naughton. The OO7 benchmark. *ACM SIGMOD Int. Conf. On Management of Data*, pp. 12-21, Washington, 1993.

[11] D. Chamberlin, J. Robie, D. Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. *ACM SIGMOD Workshop on Web and Databases (WebDB'00)*, Dallas, 2000.

[12] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu. XML-QL: A Query Language for XML. http://www.w3.org/TR/NOTE-xml-ql/.

[13] V. Apparao, S. Byrne. M Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, L. Wood. Document Object Model, 1998. http://www.w3.org/TR/REC-DOM-Level-1/.

[14] D. Florescu, D. Kossman. A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database, Report 3680 INRIA, France, May 1999.

[15] M. Fernandez, J. Simeon, P. Wadler. XML Query Languages: Experiences and Exemplars, 2000. http://www-db.research.bell-labs.com/user/simeon/xquery.html

[16] R. Goldman, J . McHugh, J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language, *ACM SIGMOD Workshop on Web and Databases (WebDB'99)*, 1999.

[17] J. Gray. The Benchmark Handbook: For Database and Transaction Processing Systems, 2nd Edition, Morgan Kaufmann Publishers, Inc., 1993.

[18] B. Chang, M. Scardina, K. Karun, S. Kiritzov, I. Macky, A. Novoselsky, N. Ramakrishnan. ORACLE XML Handbook (184-190), 2000.

[19] P. Frankhauser, M. Marchiori, J. Robie. XML Query Requirements, 2000. http://www.w3.org/TR/xmlquery-req/.

[20] J. Robie, D. Chamberlin, D. Florescu. Quilt: an XML query language, 2000. Available at http://www.gca.org/papers/xmleurope2000/papers/s08-01.html

[21] D. Brickley, R.V. Guha. Resource Description Framework (RDF) Schema Specification 1.0, 2000. http://www.w3.org/TR/rdf-schema/.

[22] XML Query Requirements. W3C Working Draft 15 August 2000. http://www.w3.org/TR/xmlquery-req

[23] J. Robie, J. Lapp, D. Schach. XML Query Language (XQL), 1998. http://www.w3.org/TandS/QL/QL98/pp/xql.html

[24] D. Fallside. XML Schema Part 0: Primer, 2001. http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/.

[25] H. Thompson, D. Beech, M. Maloney, N. Mendelsohn. XML Schema Part 1: Structures, 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[26] P. Biron, A. Malhotra. XML Schema Part 2: Datatypes, 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[27] A. Sahuguet, L. Dupont, T. L. Nguyen. Querying XML in the New Millennium. http://db.cis.upenn.edu/KWEELT/.

[28] 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.

[29] Transaction Processing Performance Council. http://www.tpc.org/.

[30] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler. Extensible Markup Language (XML) 1.0 (Second Edition), 2000. http://www.w3.org/TR/2000/REC-xml-20001006/.

[31] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu. XML-QL: A query language for XML, 1998. http://www.w3.org/TR/NOTE-xml-ql/.

[32] S. Adler, A. Berglund, J. Caruso, S. Deach, P. Grosso, E. Gutentag. Extensible Stylesheet Language (XSL),2000. http://www.w3.org/TR/xsl/.

[33] D. Chamberlin, D. Florescu, J. Robie, J. Sim. XQuery: A Query Language for XML, 2000. http://www.w3.org/TR/xmlquery/.

[34] S. Bressan, G. Dobbie, Z. Lacroix, M.L. Lee, U. Nambiar, Y.G. Li, B. Wadhwa. XOO7: Applying OO7 Benchmark to XML Query Processing Tools, NUS Technical Report TRB6/01, June 2001.

9. APPENDIX

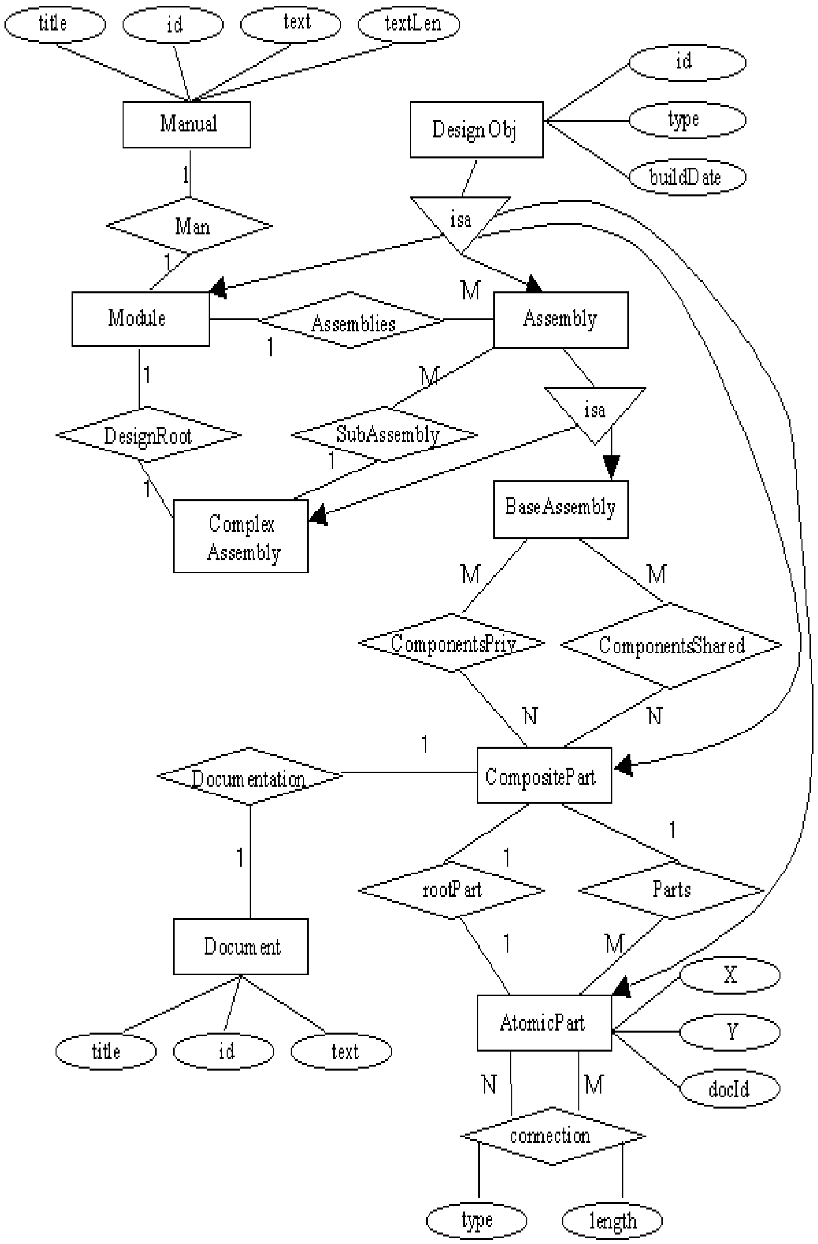

Figure 1: Entity-relationship diagram for the OO7 benchmark. [CDN94]

```
<!DOCTYPE Module>
<!ELEMENT Module (Manual, ComplexAssembly)>
<!ELEMENT Manual (#PCDATA)>
<!ELEMENT ComplexAssembly (ComplexAssembly+ | BaseAssembly+)>
<!ELEMENT BaseAssembly (CompositePart+)>
<!ELEMENT CompositePart (Document, AtomicPart+, connection+)>
<!ELEMENT Document (#PCDATA)>
<!ELEMENT Connection (Part1, Part2)>
<!ELEMENT AtomicPart EMPTY>
<!ELEMENT Part1 EMPTY>
<!ELEMENT Part2 EMPTY>
<!ATTLIST Module      MyID        NMTOKEN #REQUIRED
                      type        CDATA        #REQUIRED
                      buildDate   NMTOKEN      #REQUIRED>
<!ATTLIST Manual      MyID        NMTOKEN      #REQUIRED
                      title       CDATA        #REQUIRED
                      textlen     NMTOKEN      #REQUIRED>
<!ATTLIST ComplexAssembly
                      MyID        NMTOKEN      #REQUIRED
                      type        CDATA        #REQUIRED
                      buildDate   NMTOKEN      #REQUIRED>
<!ATTLIST BaseAssmebly
                      MyID        NMTOKEN      #REQUIRED
                      type        CDATA        #REQUIRED
                      buildDate   NMTOKEN      #REQUIRED>
<!ATTLIST CompositePart
                      MyID        NMTOKEN      #REQUIRED
                      type        CDATA        #REQUIRED
                      buildDate   NMTOKEN      #REQUIRED>
<!ATTLIST Document    MyID        NMTOKEN      #REQUIRED
                      title       CDATA        #REQUIRED
                      text        CDATA        #REQUIRED>
<!ATTLIST AtomicPart  MyID        NMTOKEN      #REQUIRED
                      type        CDATA        #REQUIRED
                      buildDate   NMTOKEN      #REQUIRED
                      x           NMTOKEN      #REQUIRED
                      y           NMTOKEN      #REQUIRED
                      docId       NMTOKEN      #REQUIRED>
<!ATTLIST Connection  type        CDATA        #REQUIRED
                      length      NMTOKEN      #REQUIRED>
<!ATTLIST Part1       IDREF       NMTOKEN      #REQUIRED>
<!ATTLIST Part2       IDREF       NMTOKEN      #REQUIRED>
```

Figure 2: DTD for XML data in XOO7 Benchmark

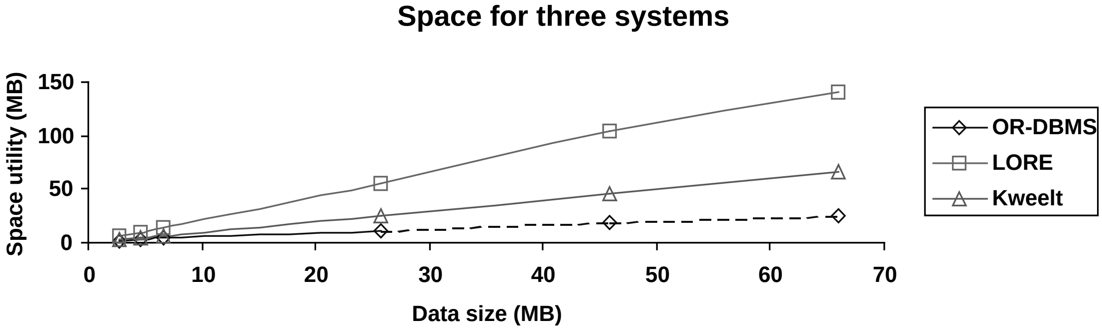

Figure 3: Space cost for three systems: LORE, Kweelt and OR-DBMS.

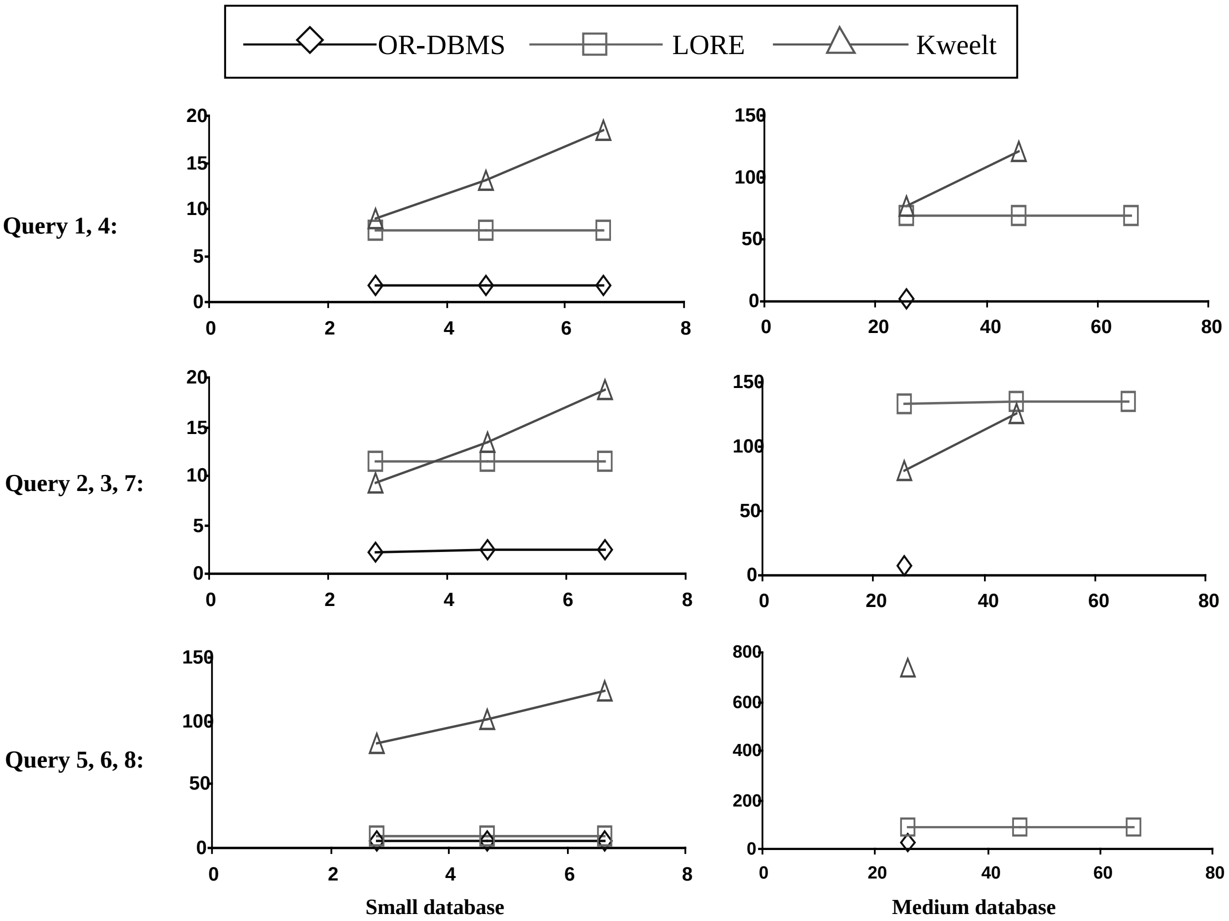

Figure 4: Response time result for the eight queries.