XMach-1: A Benchmark for XML Data Management

Timo Böhme Erhard Rahm

University of Leipzig, Germany
http://dbs.uni-leipzig.de

Abstract. We propose a scaleable multi-user benchmark called XMach-1 (*X*ML Data *Ma*nagement ben*ch*mark) for evaluating the performance of XML data management systems. It is based on a web application and considers different types of XML data, in particular text documents, schema-less data and structured data. We specify the structure of the benchmark database and the generation of its contents. Furthermore, we define a mix of XML queries and update operations for which system performance is determined. The primary performance metric, Xqps, measures the query throughput of a system under response time constraints. We will use XMach-1 to evaluate both native XML data management systems and XML-enabled relational DBMS.

1 Introduction

The need to efficiently store and manage large amounts of XML data is rapidly increasing due to the growing use of XML as an improved web format, as the native data format for a variety of applications and as a standard interchange format especially in the e-business domain. Two main types of systems are promoted to manage such XML data, namely native XML data stores [GMW99] [GMD99] [SAG00], and relational/object-relational DBMS augmented with an extension to store and manipulate XML data [DFS99] [FK99] [SKWW00] [STHZ99]. Native data stores are tailored to XML requirements and thus promise performance benefits and improved support for specific XML requirements (e.g., complex document structure, fast path navigation, text search). Relational and object-relational systems, on the other hand, typically provide good scalability and a large repertoire of performance-improving techniques, e.g. for query processing, that can be exploited for at least certain usage forms of XML data. Furthermore, they may avoid having separate data management systems for SQL and XML. An overview of currently available or announced XML data management systems can be found in [Bou00].

In this paper we propose a multi-user benchmark called XMach-1 (*X*ML Data *Ma*nagement ben*ch*mark) to realistically evaluate the performance of the different approaches to handle XML data. The benchmark should allow us to identify advantages and shortcomings of available XML data management approaches and to quantify their respective performance impact. Identifying major performance factors for XML storage and query processing also helps devising enhanced mechanisms for XML data management and is thus useful for both research and development.

In the next section we briefly discuss related benchmark work, in particular TPC-W. We then discuss major requirements for an XML data management benchmark. Next, we present the specification of XMach-1 which is based on a web-oriented usage scenario of XML data. We specify the structure and generation of the benchmark database as well as the workload mix consisting of XML queries and update operations. Furthermore, we present the XMach-1 performance metrics.

2 Related Work

So far a comprehensive benchmark for evaluating different approaches for XML data management has not been proposed. In [FK99] a benchmark is used to compare alternative mapping schemes for storing XML data in a relational database. The benchmark is not based on a real

world application and uses a single synthetically created XML document. Only single-user response times for various query types and update functions on a small database (80 MB) are determined. One observation was that short response times can be achieved for the considered queries while reconstruction of an XML document takes a very long time. The benchmark does not measure throughput performance which is of key importance for XML data management having to support many users. This can only be captured by a multi-user benchmark.

There are numerous domain-specific database benchmarks, e.g., for OLTP (TPC-C), decision support (TPC-H, TPC-R, APB-1), information retrieval, spatial data management (Sequoia) etc. [Gra93] [SFGM93] [ON97] [OLA98] [TPC00] [Cha00]. Furthermore, specific benchmarks have been proposed for object-oriented (OO1, OO7) and object-relational databases (Bucky) [CDN93] [CDN97] [CS92]. In early 2000, the e-commerce benchmark TPC-W was released by the leading database benchmark committee TPC [TPC00]. TPC-W represents a comprehensive multi-user web benchmark for evaluating sustained peak performance and cost-effectiveness for a complete e-commerce installation. It defines in detail various web interactions (browsing, ordering, …) resulting in the invocation of several transaction types on a backend database. Peak performance is measured in WIPS (web interactions per second) when at least 90% of the interactions meet a certain response time limit ranging from 3 to 20 s depending on the interaction type. Cost effectiveness is measured in $/WIPS taking into account the cost of the entire system (hardware, software, etc.) over 3 years. TPC-W is currently not tailored to XML but defined for a relational database (relational schema, relational data generator). The database structure and workload mix are largely OLTP-oriented and do thus not cover specific XML features such as support for complex document structures.

3 Requirements

In [Gra93], Gray argues for the use of domain-specific database benchmarks which should be relevant (measuring the performance of typical operations for the respective domain), portable to different platforms, scaleable (applicable to small and large computer systems) and simple. For XML data management this translates into the use of a web-based application domain, storage of XML documents and data, and measurement of throughput and response time performance for a variety of XML operations. To support scalability, different database sizes and load volumes should be supported.

This section analyses the XML-specific requirements w.r.t. data organization and operations in more detail to motivate the design of the XMach-1 benchmark. It seems tempting to require that the benchmark should capture most features of the released or soon-to-be-released XML-based specifications of the World Wide Web Consortium [WWW00][XQR00]. However, this would make it difficult to achieve a “simple” benchmark. Moreover, it would not be possible to run the benchmark with currently available systems which typically lack many features. In the first version of the benchmark we thus concentrate on basic XML features and rather simple operations that are supported by available systems. For instance, we do not require support for XML schema, XML namespaces, CDATA sections, entities and a particular XML query language. We expect that future versions of the benchmark will include additional features.

With respect to data storage the main usage forms of XML should be captured by the benchmark. In particular, text and non-text documents/data should be supported. As shown in Table 1, these document types differ w.r.t content type and preservation of element order. In both cases, the documents may or may not conform to a schema or DTD (document type definition). Schema-less XML data is feasible due to the self-describing character of XML.

Text documents contain natural language text marked up in its structure by XML tags. Obviously the order of elements in these documents is important (e.g. order of chapters, paragraphs). An important difference to other types of XML data is the possibility of mixed content [BPS98] meaning that there may be subelements or links embedded within text (character data), e.g. as commonly found in web pages. To support the hypertext character of XML web documents it is desirable that the benchmark includes link elements.

	schema (DTD)	Content type	order preservation
Text documents	yes / no	mixed content, natural language	yes
Non-text documents	yes / no	element content	yes / no

Table 1: Types of XML data

For data-centric (non-text) applications, we also may have schema-less or schema-based data. Here the schema-based case is referred to as *structured data.* Structured XML data is similar to conventional database data but often more complex in structure; interchange data in XML belongs to this category. In contrast to text documents, element content prevails where element types may either only contain child elements or only character data [BPS98]. While the XML specification requires element order be preserved, many applications do not depend on it. The benchmark should thus specify when order preservation is not required so that this relaxation may be used by the DBMS to improve performance.

The benchmark should define a set of "typical" update operations and queries on these types of XML data to determine overall performance of an XML data management system. In particular, new XML documents should be loaded into the database and existing documents should be deleted or replaced. The query mix should cover hierarchical, sequence (order) -based and set-oriented queries as well as information retrieval tasks such as text search for phrases. Also desirable are XML-specific queries such as search for metadata (children/parents of an element, elements having a specific attribute or name part, …). There should be different result sizes for queries ranging from single elements to complex element hierarchies. Also desirable are queries requiring the reconstruction of results in a new structure. Furthermore, typical database functionality such as join, sort and aggregation should be captured.

4 Benchmark description

The XMach-1 benchmark is based on a web application in order to model a typical use case of a XML data management system. The system architecture shown in Figure 1 consists of four parts: the XML database, application servers, loaders and browser clients. The database contains a directory structure and XML documents that are assumed to be loaded from various data sources in the internet by loader programs (e.g. robots crawling the web or a registration tool where web-site authors can add documents to the database). Every document has a unique URL which is maintained together with document metadata in the directory structure. The application servers run a web (HTTP) server and other middleware components to support processing of the XML documents and to interact with the backend database.

Figure 1 shows the System under Test (SUT) for which response time and throughput performance is determined. It includes both database and application servers similar to TPC-W. The XMach-1 benchmark is not meant to evaluate all kinds of XML-related processing but primarily XML

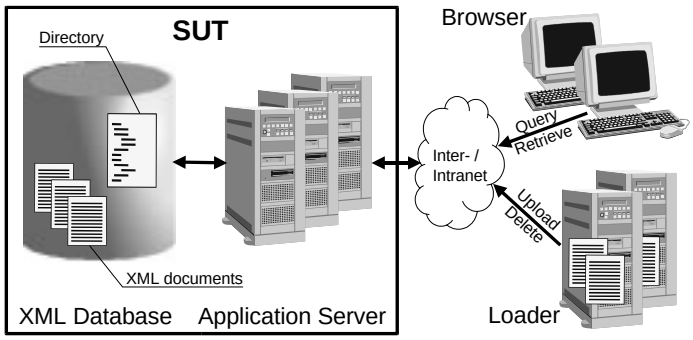

Figure 1: Components of benchmark architecture

data management. We feel that this does not allow restricting measurements to the database server because database processing is increasingly spread across the database backend and application servers in order to allow for improved throughput, scalability, load balancing and caching. For instance, mapping XML data to a relational format may be performed either at the database backend or on the application server.

The number of database and application servers is not predetermined but can be chosen according to the performance goals. Caching of XML data at the application server is allowed but queries must see current data. Analogously to the TPC-W definition of "web page consistency" we require that query results only contain transaction consistent XML data and that an update must be reflected in query results within 30 s after the update is committed.

The query and update workload is generated by virtual (emulated) browsers and loaders. The number of these clients is not predetermined but can be chosen according to the throughput goals. Interaction with the application server is via a HTTP interface. The clients and the connection with the application servers are not part of the SUT so that the corresponding processing times and communication delays are not included in query response times. Response times are measured at the application server and cover the time duration between query arrival and sending the result in XML format to the client. We do thus not measure client-oriented transformations of XML data, e.g. between XML and HTML.

Our benchmark measures throughput performance for a mix of queries and update operations that have to be executed within specific response time limits. Two benchmark variants are distinguished depending on whether the XML documents are schema-less or conform to DTDs or schemas. This allows us to run the benchmark with systems only supporting one of the two cases. If both variants are possible, we can evaluate the performance impact of having schema support.

In the following we first describe the database structure and data generation in more detail. In 4.3 we present the workload model. Finally we discuss the XMach-1 performance metrics.

4.1 Database structure

The XML database contains both structured data and text documents. The directory contains metadata about all text documents. It represents structured data as it is schema-based and holds element content only. Such a structured representation of metadata is found in many XML applications (e.g., product catalogs for e-commerce). The text documents collected from the web are referred to as managed documents. Depending on whether we run the schema-based or schema-less variant of the benchmark they either all conform to a DTD (well-formed and valid documents) or they are assumed to be schema-less (well-formed but not valid documents).

4.1.1 Directory document

The DTD of the directory document is given in Figure 2. It maintains a unique URL for each managed document as well as other metadata such as document id, insert time, update time etc.

We do not restrict the directory to a flat relational structure but use a hierarchical XML representation for URLs. For this purpose, URL components are stored as separate elements (host, path). In Figure 3 the directory structure of two documents with URLs http://www.test-company.com/products/overview.xml and http://support.test-company.com/-help.xml are shown. The sequence of the

```
<!ELEMENT   directory     (host+) >
<!ELEMENT   host          (host+ | path+) >
<!ATTLIST   host
  name      CDATA         #REQUIRED >
<!ELEMENT   path          (path+ | doc_info) >
<!ATTLIST   path
  name      CDATA         #REQUIRED >
<!ELEMENT   doc_info      EMPTY >
<!ATTLIST   doc_info
  doc_id    ID            #REQUIRED
  loader    CDATA         #REQUIRED
  insert_time  NMTOKEN    #REQUIRED
  update_time  NMTOKEN    #IMPLIED >
```

Figure 2: Directory DTD

host address components is reversed to allow for a hierarchical representation of the URLs with little redundancy. The URLs in the example are for illustration purposes only. The benchmark generates URLs with other naming conventions as described in subsection 4.2.1.

Preservation of element order is not required since the order of elements per hierarchy level is immaterial.

```
<directory>
  <host name="com">
    <host name="test-company">
      <host name="www">
        <path name="products">
          <path name="overview.xml”>
            <doc_info  doc_id="2"
              loader="robot1"
              insert_time="20000110152530"/>
          </path>
        </path>
      </host>
      <host name="support">
        <path name="help.xml">
          <doc_info   doc_id="3"
            loader="robot1"
            insert_time="20000612094430"/>
        </path>
      </host>
    </host>
  </host>
</directory>
```

Figure 3: Example directory document

4.1.2 Managed documents

Most of the benchmark data consists of managed documents. These text documents are generated synthetically to support almost arbitrary numbers of documents with well-defined contents. Each document has a unique document id. It is added to the original XML document during insert. This id is also kept in the directory and can be used for join operations.

In the schema-based version of XMach-1 the documents conform to the generic DTD shown in Figure 4. Thus the text documents consist of chapters, sections and paragraphs for which order preservation is required. In order to enable specific queries some of the documents will contain an author tag or link tags to other documents. The link tag should be specified according to the XLink specification [DMT00]. Since this specification is not yet finalized and current XML databases may not support it, we allow to replace the xlink attributes by an attribute with name href. Since paragraphs contain both text and links we have mixed element content.

Having only one DTD for all documents is unlikely to be realistic for larger collections of documents coming from different sources. To simulate multiple DTDs we thus differentiate several variations of the generic DTD of Figure 4 by adding an integer number to all tag names but ‘author’ and ‘link’. For instance, the 17^{th} DTD will have elements named document17, chapter17, author, section17, etc.

In the schema-less variant of XMach-1 we use the same documents than for the schema-based case but no DTDs are kept by the data management system. This allows us to compare the performance for the two variants to quantify the impact of using DTDs.

```
<!ELEMENT  document  (title, chapter+)>
<!ATTLIST  document
    author     CDATA    #IMPLIED
    doc_id     ID       #IMPLIED>
<!ELEMENT  author    (#PCDATA)>
<!ELEMENT  title     (#PCDATA)>
<!ELEMENT  chapter   (author?, head, section+)>
<!ATTLIST  chapter
    id         ID       #REQUIRED>
<!ELEMENT  section   (head, paragraph+, section*)>
<!ATTLIST  section
    id         ID       #REQUIRED>
<!ELEMENT  head      (#PCDATA)>
<!ELEMENT  paragraph (#PCDATA | link)*>
<!ELEMENT  link      EMPTY>
<!ATTLIST  link
    xlink:type (simple)  #FIXED "simple"
    xlink:href CDATA     #REQUIRED>
```

Figure 4: DTD for managed documents

4.2 Database population and scaling

The XML database is populated with an initial number of managed documents before the benchmark is executed. To support scaling to different system configurations, four database sizes are possible with an initial document number of 10.000, 100.000, 1.000.000 or 10.000.000. Due to insert operations in the workload mix (see below) the number of documents will increase during benchmark execution.

Database population includes adding generated documents to the database and corresponding entries to the directory document. The generation process uses a random number generator (RNG) for uniformly distributed numbers. In the following we outline how to generate URLs, text, documents and search phrases for XMach-1.

4.2.1 URL generation for new managed documents

Every document added to the database gets a unique URL and a document id. The URLs do not use real world names such as com or edu. Instead they are made up of easy to generate strings following generic rules. This is necessary in order to predict result sets of operations on the directory structure.

A URL consists of three host elements and one to four path elements with the last path element termed document name. Generated URLs have to conform to the following regular expression syntax:

ahost**{**1-3**}.** bhost**{**1-(*N*/100)**}**.chost**{**1-5**}/[**apath**{**1-3**}/[**bpath{1-3}/**[**cpath**{**1-3**}]]]***NAME*

In this expression, curly braces contain ranges of numbers from which one has to be chosen by RNG, square brackets denote optional parts. The number of path elements is determined by RNG. N stands for the initial number of managed documents in the database. The document name is generated from the document and loader ids: NAME=DOC_ID+LOADER_ID+'.xml'.

4.2.2 Data generation for text (#PCDATA)

In order to get realistic performance results the synthetically generated text within the managed documents should have similar properties than natural language text (NLT). Words in NLT are commonly non-uniformly distributed according to Zipf's Law [Zip35]. We use a word list of the 10,000 most common English words and choose words randomly by applying Zipf's Law as described in [GSEB94].

4.2.3 Document generation

The structure of new documents follows the DTDs described in section 4.1.2. We use an algorithm generating document trees of variable (skewed) depth and size. The generation process is controlled by the parameters and their settings listed in Table 2. If a range is specified the actual value has to be chosen by a self similar distribution generator following the 80-20 rule as described in [GSEB94]. The tree generation algorithms first determines the total number of sections and uses probabilities at every node level to create a new child or to go back to the parent node. An additional rule ensures that every paragraph has at least one section element. For an average word length of 6 characters (including the space delimiter) the mean size of a document is about 10 kB corresponding to the typical size of current web documents.

The probability of having an author attribute or element is 0.5. Author names are uniformly chosen from a list

Document generation parameter	Value
number of sections per document	5 – 150
number of paragraphs per section	1 – 15
number of sentences per paragraph	2 – 30
number of words per sentence	3 – 30
probability of having an author attribute/element	0.5
number of words per head or title element	2 – 12
probability of having a phrase within a sentence	0.01
probability of having a link element within a paragraph	0.05
number of documents per DTD	2 – 100

Table 2: Document generation parameters

of 20,000 last names and 8,000 first names. In order to ensure a reasonable number of documents per author we do not generate author names randomly from all possible combinations of first and last names. If N is the total number of managed documents, LN the total number of last names and FN the total number of first names, we choose the last name from range $[1 \ldots MIN(LN; 2 \cdot \sqrt{1/10} \cdot \sqrt{N})]$ and first name from $[1 \ldots MIN(FN; \cdot \sqrt{1/10} \cdot \sqrt{N})]$. This results in about N/5 different author names.

The contents of title and head elements as well as sentences making up the contents of a paragraph element are generated using the method described in section 4.2.2. The first word of a sentence starts with an uppercase character. A dot marks the end of a sentence. With a specified probability a phrase is inserted after the first word of a sentence from a phrase list (see next subsection). Furthermore, a link element is inserted within a paragraph after the first sentence according to the link probability. All links must point to a managed document.

For the schema-based version of XMach-1, each document is assigned to one of the DTD variants according to the specified range for the number of documents per DTD. It is up to the implementation whether a reference to an external DTD definition is provided or the complete DTD is included within the document.

4.2.4 Phrase list generation

The XMach-1 operation mix contains searches for phrases in documents. In order to get comparable results we have to know what phrases we can search for and how many hits we get on average. Therefore a list of search phrases is generated where each phrase is a word triple. The number of phrases is 1/10th of the initial number of documents. With the values defined in the previous subsection we will have about one phrase per document so that a specific phrase is contained in about 10 documents on average. Word triples are chosen from the word list starting with index position 1,000 to prevent having possible stop words in the phrase.

4.3 Operations

The XMach-1 workload mix consists of eight query types and three update types. We define the operations merely verbally since no XML query standard has been released so far. However, all operations can be carried out by most available XML data management systems either by using their respective query language or by an appropriate application program (executed on the application server that is included in the SUT).

4.3.1 Queries

The query types are listed in Table 3. The operations cover a wide range of processing features on both structured data and text documents considering the requirements of Section 3. Settings for query parameters such as URL, author and doc_id are randomly chosen from the values available in the database. To simulate broken links we require that 5% of the URL requests refer to undefined URLs.

To illustrate that the queries can be expressed by current language proposals we show the definitions of Q1 for URL /ahost1.bhost2.chost3/001_loader1.xml and Q4 for doc_id ’123’ in Quilt syntax [CRF00]:

```
Q1: FOR   $a  IN  /directory
          $b  IN  /*
    WHERE   $a/host[@name="chost3"]/host[@name="bhost2"]/host[@name="ahost1"]/
    (wrapped) path[@name="001_loader1.xml"]/doc_info/@doc_id = $b/@doc_id
    RETURN  $b

Q4: FOR   $a  IN  /*[@doc_id="123"]//*[starts-with(name(),"section")]/head
    RETURN $a
```

ID	Description	Comment
Q1	Get document with given URL	Return a complete document (complex hierarchy with ordering preserved)
Q2	Get doc_id and URL from documents containing a given phrase in paragraph elements	Text retrieval query. The phrase is chosen from the phrase list. Join needed to get URL for qualifying documents
Q3	Start with first element which name starts with 'chapter' and recursively follow first element which name starts with 'section'. Return each of the 'section' elements	Simulates navigating a document tree using sequential operators
Q4	Return flat list of head elements which are children of elements whose names start with 'section' from a document given by doc_id	Restructuring operation simulating creation of a table of contents
Q5	Get document name (last path element in directory structure) from all documents which are below a given URL fragment	Browse directory structure. Operation on structured unordered data
Q6	Get doc_id and id of parent element of author element with given content	Find chapters of a given author. This tests efficiency of index implementation
Q7	Get doc_id from documents which are referenced by at least four other documents	Get important documents. Needs some kind of group by and count operation
Q8	Get doc_id from the last 100 updated documents having an author attribute	Needs count, sort, join and existential operations and accesses metadata

Table 3: Query operations

4.3.2 Data manipulation operations

Three types of manipulation operations are executed during the XMach-1 benchmark to insert, delete and update XML documents (Table 4). All operation types change the directory document. They must be carried out within transactions to guarantee database consistency. Documents are generated as described in Section 4.2. For the schema-based variant of the benchmark new DTDs are also added to maintain the same average number of documents per DTD.

ID	Description	Comment
M1	Insert document with given URL	The loader generates a document and URL and sends them to the HTTP server
M2	Delete a document with given doc_id	A robot requests deletion, e.g. because the corresponding original document no longer exists on the web
M3	Update URL and update_time for a given doc_id	Update directory entry

Table 4: Data manipulation operations

4.3.3 Operation mix

In order to get comparable performance results for different SUTs there must be a fixed composition of the workload w.r.t. the introduced operation types. Table 5 defines the shares for the 11 operation types. Q1 (get complete XML document for given URL) is the main operation type which will also be used for determining throughput. Its share should be 30% or less. For the other operation types, the percentages given in Table 5 indicate the minimal shares. They can be higher but would then reduce the Q1 share and thus throughput.

Only 2% of the operations are updates. Since we have three times as many insert (M1) than delete (M2) operations, the total number of documents increases during the benchmark proportionally to the throughput and execution time. This stresses transaction and cache consistency management as well as query processing, e.g., to efficiently process newly added documents.

Operation ID	Percentage
Q1	<= 30
Q2	>= 20
Q3, Q4, Q5, Q6	>= 10 each
Q7, Q8	>= 4 each
M1	>= 0,75
M2	>= 0,25
M3	>= 1

Table 5: Operation mix composition

4.4 Performance metrics and benchmark execution

XMach-1 is a multi-user benchmark so that throughput is our primary performance metric. For the schema-based version of XMach-1 throughput is measured in

Xqps (XML queries per second)

In the schema-less case we measure throughput in $Xqps_{sl}$ (Xqps schema less). In both cases we only consider the number of executed Q1 queries since the fixed workload composition guarantees a corresponding number of executions for the other operation types. Only Xqps values referring to the same database size (initial number of documents) can be compared with each other. The benchmark execution time has to be at least one hour.

The throughput must be obtained while guaranteeing short response times since it is rather easy to increase throughput without response time limits. Analogously to the TPC specifications we require that for all query types (Q1-Q8) and M3 90% of the operations be executed under 3 s. The insert and delete operations M1 and M2 are not time critical so that we set the 90^{th} percentile response time limit to 20 s. As discussed above, the response times include the execution times at the database and application server but not at the client.

To achieve a true multi-user environment with a realistic number of concurrent clients we require that each browser and loader runs at most one operation at a time. Furthermore, after completing an operation there is a think time between 1 and 10 s (chosen by RNG) before the next operation is started. The benchmark results to be reported include the throughput performance, for all operation types their execution frequency and 90% response times, the time to load the database as well as the sizes of the XML database and index structures.

Cost effectiveness in $/Xqps ($ / $Xqps_{sl.}$) can be measured by determining the $ cost of the SUT and dividing it by the throughput in Xqps ($Xqps_{sl.}$). For a specification of how to determine system cost we refer to the TPC-W specification [TPC00].

5 Summary and outlook

We have presented the design of XMach-1, a scaleable multi-user benchmark for evaluating the performance of XML data management systems. It measures the throughput of a generic XML-based web application consisting of a XML database and middleware components. The database contains both structured data and text documents. Two benchmark variants with separate throughput metrics, Xqps and $Xqps_{sl,}$ are distinguished for schema-based and schema-less XML documents. We have specified the structure of the benchmark database and its population. Furthermore, we defined a workload mix consisting of eight query and three update types. The proposal borrows several elements from proven TPC benchmarks.

We believe XMach-1 meets the general benchmark requirements of relevance, portability, scalability and simplicity as well as the XML-specific requirements discussed in section 3. In particular, by focussing on basic XML features it can be run for available systems, both XML-enabled relational DBMS and native XML data management systems, to compare their performance. Also, it will be easy to define XMach-1 in terms of standardized versions of XML schema and XML query once they are available.

We have started to develop a XMach-1 benchmark environment for different XML data management systems and expect to have performance results soon available.

Acknowledgements: We would like to thank Jim Gray, Phil Bernstein, Ralph Busse and the reviewers for helpful comments.

References

[Bou00] R. Bourret: XML Database Products. Nov. 2000. http://www.rpbourret.com/xml/XMLDatabaseProds.htm

[BPS98] T. Bray, J. Paoli, C. M. Sperberg-McQueen: Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml, 1998

[CDN93] M. J. Carey, D. J. DeWitt, J. F. Naughton: The OO7 benchmark. In *ACM SIGMOD Conference*, pp. 12-21, Washington, 1993

[CDN97] M. Carey, D. DeWitt, J. Naughton, et al.: The Bucky Object-Relational Benchmark. In *Proc. ACM SIGMOD Conf.*, pp. 135-146, Tucson, AZ, 1997

[CRF00] D. D. Chamberlin, J. Robie, D. Florescu: Quilt: An XML Query Language for Heterogeneous Data Sources. In *Proc. 3rd ACM SIGMOD WebDB workshop*, 2000

[Cha00] A. B. Chaudhri: Benchmarks. 2000. http://www.soi.city.ac.uk/~akmal/html.dir/benchmarks.html

[CS92] R. Catell, J. Skeen: Object operations benchmark. In *ACM Transactions on Database Systems*, 17(1), pp. 1-31, 1992

[DFS99] A. Deutsch, M. Fernandez, D. Suciu: Storing semistructured data with STORED. In *Proc. ACM SIGMOD Conf.*, pp. 431-442, Philadephia, 1999

[DMT00] S. DeRose, E. Maler, D. Orchard, B. Trafford: XML Linking Language (XLink) Version 1.0. http://www.w3.org/TR/xlink/, 2000

[FK99] D. Florescu, D. Kossmann: Storing and Querying XML Data using RDBMS. In *IEEE Data Engin. Bulletin, Special Issue on XML*, 22(3), pp. 27-34, 1999

[GMD99] GMD: GMD-IPSI XQL Engine. http://xml.darmstadt.gmd.de/xql/, 1999

[GMW99] R. Goldman, J. McHugh, J. Widom: From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In *Proc. 2nd ACM SIGMOD WebDB workshop*, pp. 25-30, Philadelphia, 1999

[Gra93] J. Gray ed.: The Benchmark Handbook. Morgan Kaufmann, San Mateo, CA, 1993. http://www.benchmarkresources.com/handbook/

[GSEB94] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, P. Weinberger: Quickly generating billion-record synthetic databases. In *Proc. ACM SIGMOD Conf.*, 1994

[OLA98] OLAP Council: APB-1 OLAP Benchmark Release II. , 1998. http://www.olapcouncil.org/research/bmarkly.htm

[ON97] P. E. O'Neil: Database Performance Measurement. In *The Computer Science and Engineering Handbook*, CRC Press, 1997: 1078-1092

[SFGM93] M. Stonebraker, J. Frew, K. Gardels, J. Meredith: The Sequoia 2000 Benchmark. *SIGMOD Conference*, pp. 2-11, 1993

[SKWW00] A. Schmidt, M. L. Kersten, M. Windhouwer, F. Waas: Efficient Relational Storage and Retrieval of XML Documents. In *Proc. 3rd ACM SIGMOD WebDB workshop*, pp. 47-52, Dallas, 2000

[SAG00] Software AG: Tamino - The Information Server for Electronic Business. http://www.softwareag.com/tamino/, 2000

[STHZ99] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton: Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Proc. 25th VLDB conf.*, pp. 302-314, Edinburgh, 1999

[TPC00] Transaction Processing Performance Council. http://www.tpc.org, 2000

[WWW00] World Wide Web Consortium (W3C): Extensible Markup Language (XML). http://www.w3.org/XML, 2000.

[XQR00] D. Chamberlin, P. Fankhauser, M. Marchiori, J. Robie: XML Query Requirements. W3C Working Draft, http://www.w3.org/TR/xmlquery-req, 2000

[Zip35] G. K. Zipf: The Psychobiology of Language. Houghton Mifflin, Boston, 1935