XML Benchmarking – the State of the Art and Possible Enhancements

Irena Mlynkova
Charles University, Czech Republic

ABSTRACT

Since XML technologies have become a standard for data representation, numerous methods for processing XML data emerge every day. Consequently, it is necessary to compare the newly proposed methods with the existing ones, as well as analyze the effect of a particular method when applied to various types of data. In this chapter, we provide an overview of existing approaches to XML benchmarking from the perspective of various applications and we show that to date the problem has been highly marginalized. Therefore, in the second part of the chapter we discuss persisting open issues and their possible solutions.

1. INTRODUCTION

Since XML (Bray et al., 2006) became a de-facto standard for data representation and manipulation, numerous methods have been proposed for efficiently managing, processing, exchanging, querying, updating and compressing XML documents. And new proposals emerge every day. Naturally, each author performs various experimental tests using the newly proposed method and describes its advantages and disadvantages. But, it can be very difficult for a future user to decide which of the existing approaches is the most suitable for his/hers particular requirements on the basis of the descriptions of methods. The problem is that various methods are usually tested on different data sets derived from diverse sources which either do not yet exist or which were created only for the testing purposes, with special requirements of particular applications etc.

An author of a new method will encounter a similar problem whenever he/she wants to compare the new proposal with an existing one. This is possible only if the source or executable files of the existing method or, at least, identical testing data sets are available. But, too often it is impossible to have access to this information. In addition, in the latter case, the performance evaluation is limited by the testing set whose characteristics are often unknown. Hence, a reader finds it difficult to obtain a clear notion of the analyzed situation.

An analogous problem occurs if we want to test the behaviour of a particular method on various types of data, or determine the correlation between the efficiency of the method and changing complexity of the input data. Not even the process of gathering the testing data sets is simple. Firstly, the real-world XML data usually contain a huge number of errors (Mlynkova et al., 2006) which need to be corrected. And what is worse, the real-world data sets are usually surprisingly simple and do not cover all constructs allowed by XML specifications.

Currently, there exist several projects which provide a set of testing XML data collections (usually together with a set of testing XML operations) that are publicly available and well-described. We can find either fixed (or gradually extended) databases of real-world XML data (e.g. project INEX (INEX, 2007)) or projects which enable us to generate synthetic XML data on the basis of user-specified characteristics (e.g. project XMark (Busse, 2003)). But, in the former case, we are limited by the characteristics of the testing set; whereas, in the latter case, the characteristics of the generated data that can be specified are trivial (such as the amount and size of the data).

1.1. Goals of the Chapter

The first aim of this chapter is to provide an overview of existing XML benchmarking projects, i.e. projects which provide a set of testing XML data collections, XML operations/scenarios etc. We will discuss their main characteristics and in particular the issues related to their versatility. We will show that the problem of sophisticated XML benchmarking has been so far highly marginalized and the number of possibilities for acquiring at least a reasonable testing set of XML data is surprisingly low.

Since the key operations of XML processing are undoubtedly parsing, validating and querying, most of the existing benchmarking projects focus mainly on them. But, there are also other popular and useful XML technologies or operations with XML data and, hence, there also exist benchmarks determined for other purposes. Nevertheless, their number is surprisingly low or the existing representatives are already obsolete.

The next aim of the chapter is to identify the most significant related open issues and unsolved problems. In particular, we will deal with a system which is able to generate synthetic XML data on the basis of a wide range of user-specified characteristics. We will focus on three aspects of the problem – automatic generation of synthetic XML documents, automatic generation of their XML schema, and automatic generation of respective XML queries. The main idea is that the author of a new method will be able to test its behaviour on any kind of data that can be described using this set of characteristics. On the other hand, any other author can use the same setting-up of characteristics, repeat the generation of the testing data sets and compare a new method with the existing results.

In general, we will describe and discuss such a system in full and focus on the related open problems as well as possible solutions. The particular implementation can then focus only on selected aspects appropriate for concrete exploitation.

Note that there already exist several analytical surveys on XML benchmarking projects (such as Nambiar et al., 2001; Bohme et al., 2003; Manegold, 2008). But, in general, most of the surveys consider only a subset of query benchmarks, often written by authors of a particular benchmarking project and, hence, the results are slightly biased or they have already become obsolete. We will mention and briefly describe them in relevant parts of our text as well.

1.2. Roadmap

The rest of the text is structured as follows: Section 2 classifies and briefly describes the existing approaches to XML benchmarking. Section 3 provides a general summary of the findings. Section 4

describes and discusses the remaining open issues and possible solutions. And, finally, Section 5 provides conclusions.

2. EXISTING APPROACHES AND THEIR CLASSIFICATIONS

Various XML benchmarking approaches have been proposed and can be classified as follows. From the point of view of the type of data, we can distinguish benchmarks which involve real-world data and benchmarks involving synthetic data. Though the former type seems to have more practical application, the problem is that real-world data are quite simple (Bex et al., 2004; Barbosa et al., 2005; Mlynkova et al., 2006) and do not contain most of the constructs allowed by W3C specifications, whereas benchmarks enabling the testing of all the allowed constructs are quite natural.

A different type of classification of XML benchmarks distinguishes approaches which involve a fixed set of testing data sets (e.g. XML documents, XML queries, XSL transformation etc.) and approaches which enable them to be created dynamically on the basis of user-specified parameters. While in the former case the data sets can be both real-world and synthetic, naturally in the latter case the data are purely synthetic.

On the basis of the purpose of the XML benchmark, we can further distinguish benchmarks which analyze quality and behaviour of various types of applications. The most common ones are XML parsers and validators, XML management systems, XSL transformers etc. And in particular areas, we can also establish a finer classification on, for example, the basis of exploited languages and constructs, such as DTD (Bray et al., 2006) vs. XML Schema (Thompson et al., 2004; Biron et al., 2004) benchmarks, XQuery (Boag et al., 2007) vs. XPath (Clark et al., 1999) benchmarks, XPath 1.0 (Clark et al., 1999) vs. XPath 2.0 (Berglund et al., 2007) benchmarks to name a few..

In the following sections, we briefly describe the best known representatives of particular approaches and their advantages and disadvantages. We will focus mainly on benchmarks related to the basic support of XML technologies such as parsing, validating, storing, querying and updating. Naturally, there also exist advanced XML operations and technologies which can and need to be benchmarked, such as, e.g. XSL transformations or compressing XML data, but these technologies are mostly closely related to the basic ones we will deal with. On the other hand, they may require special treatment which is outside the scope of this text.

2.1. XML Datasets

Currently, one of the most typical approaches to XML testing is exploitation of fixed sets of XML data. These sets usually involve real-world XML data that represent a particular field of XML processing. Apart from interesting rather than useful examples of XML documents, such as the Bible in XML (Fields, 1996), Shakespeare's plays (Bosak, 1997), classic novels in XML (Wendover, 2001) etc., the most common types of tested XML data are usually XML exports of various databases, such as *IMDb* (IMDb, 2008) database of movies and actors, *FreeDB* (FreeDB, 2008) database of musical CDs, *DBLP* (DBLP, 2008) database of scientific papers, *Medical Subject Headings* (MeSH, 2008) database of medical terms, *SIGMOD Record* in XML (SIGMOD, 2007) etc. or repositories of real-world XML data provided from various resources, such as project *INEX* (INEX, 2007), project *Ibiblio* (Ibiblio, 2008), *Open Directory Project* (Open Directory, 2004) etc. There also exist examples of rather special XML data, such as human genes (H-invDB, 2007), protein sequences (UniProt, 2008), RNAs (RNAdb, 2005), astronomical NASA

data (XDR, 2002), linguistic trees in XML (Treebank, 1999) etc., having very uncommon structure and, hence, requiring special processing. Some of these collections were not originally created in XML format, but for the purpose of XML benchmarking they were later converted and stored in appropriate repositories, such as (XDR, 2002).

Since all these examples of XML data collections are provided without respective XML queries, XSL transformations or any other operations, they cannot be considered as true XML benchmarks.

2.2. Benchmarking Projects for XML Parsers and Validators

The first applications necessary for XML data processing are XML parsers and XML validators. Their key aim is to check the correctness of the input data, i.e. their conformance to either W3C recommendations or respective XML schemes. Hence, the benchmarks usually involve sets of both correct and incorrect data and the goal is to test whether the application being tested recognizes them correctly.

XML Conformance Test Suites

The W3C consortium has naturally provided so-called *XML Conformance Test Suites* (Martinez et al., 2008) – a set of metrics to determine how well a particular implementation conforms to *W3C XML 1.0 (Second Edition) Recommendation, Extensible Markup Language (XML) 1.0 (Third Edition), Extensible Markup Language (XML) 1.1 (First Edition)* and *Namespaces in XML 1.1*. It consists of a set of 2000 XML documents which can be divided into two basic types – *binary tests* and *output tests*.

Binary tests contain a set of documents from one of the following categories: valid documents, invalid documents, non-well-formed documents, well-formed errors tied to external entity and documents with optional errors. Depending on the category, the tested parser must either accept or reject the document correctly (therefore, the tests are called binary). The expected behaviour naturally differs depending on whether the tested parser is validating or non-validating.

On the other hand, the output tests determine whether the respective applications report information as required by the recommendation. Again, validating processors are required to report more information than non-validating ones.

Performance Evaluation of XML Parsers

With the arrival of various types of XML parsers as well as various implementations of parsers of the same type, it became necessary to evaluate their performance. Currently, we can distinguish so-called *event-driven parsers* and *object-model parsers*. The former ones read the document and, while reading, they return the respective structure; whereas, the latter parsers read the document and build it completely in memory. The former ones can be further divided into *push-parsers* and *pull-parsers* which differentiate in the ability to influence the reading process. In the case of push-parsers, the reading cannot be influenced; whereas pull-parsers read the next data only if they are “asked” to. Combinations of various parsers have also been considered.

Currently, there are numerous projects which evaluate efficiency of various subsets of known XML parsers (such as, e.g., VTD-XML, 2003; Cooper, 1999; Farwick et al., 2007; Marcus, 1999; Oren, 2002), comparing either the same types of parsers or different approaches. But, they all use either a selected set of real-world XML data or a set of synthetic documents created specifically for the purpose of the benchmark. Although the authors usually make these documents available, there seems to be no true

benchmarking project which enables the analysis of all the various aspects of the different types of XML parsers.

There are also various implementations of systems which enable to benchmark a selected subset of parsers (Chilingaryan, 2004; Kumar, 2002; Sosnoski, 2002). The sets of the supported applications can usually be extended; also, the data sets used for their comparison are available and often extensible. However, the problem is that these projects are not true benchmarking project which define a set of experiments testing various aspects and especially bottlenecks of XML parsing and validating.

2.3. Benchmarking projects for XML Data Management Systems and Query Engines

Probably the biggest set of benchmarks contains projects which focus on testing XML data management systems and query engines. The aim of the benchmarks is to analyze the versatility of these tools, i.e. the number of query constructs they are able to process successfully and how efficiently they are processed. These benchmarks can be further classified on the basis of various aspects, such as the type of query language, the number of users (i.e. single user vs. multiple users), the type of the benchmark (i.e. application-level or micro-benchmarks) etc.

The authors of paper (Schmidt et al., 2001) have discussed and specified the set of challenges that a comprehensive benchmark should cover. These involve bulk loading (since at the time the paper was published there were no recommended update operations), round-tripping (i.e. reconstruction of the original document and the price of loss-less storage), basic path traversals, casting, optional elements ordering, references, joins, construction of large results and full-text search. Most of these well-specified challenges are usually covered by the existing benchmarks.

Note that the W3C XML Query Working Group has proposed and maintains a set of so-called *XML Query Use Cases* (Chamberlin et al., 2007). But, the set of queries is not considered as a benchmark, but rather a set of examples illustrating important applications for an XML query language. On the other hand, the *XML Query Test Suite (XQTS 1.0.2)* (Rorke et al., 2007) contains over 15000 test cases, i.e. queries and expected results, which enable to test the interoperability of the W3C XML Query language. Hence, also in this case, the purpose is slightly different.

In the following text, we provide an overview of the eight best known representatives of true XML query benchmarking projects, i.e. XMark, XOO7, XMach-1, MBench, XBench, XPathMark, MemBeR and TPoX. In particular, we describe and compare their key characteristics, advantages and disadvantages.

XMark

The XML benchmarking project *XMark* (Busse, 2003) is currently one of the most popular and most commonly used XML benchmarks (Afanasiev et al., 2006). It involves a data generator called *xmlgen* which enables the creation of synthetic XML documents according to a fixed DTD of an Internet auction database. The key parameter of the required data is their size ranging from minimal document (having size of 1MB) to any arbitrary size limited only by the capacity of the particular system. The textual parts of the resulting XML documents are constructed from 17,000 most frequently occurring words in Shakespeare's plays.

The XMark project also involves 20 XQuery queries which focus on various aspects of the language, such as, e.g., array look-ups ordering, casting, wildcard expressions, aggregations, references, constructors, joins, optional elements, user-defined functions, sorting etc.

Probably for the first time, the XMark benchmark has been used by its authors for analyzing the behaviour and performance of *Monet* XML framework (Schmidt et al., 2001).

X007 Benchmark

XML benchmark *XOO7* (Bressan et al., 2003) is an XML version of the original OO7 (Carey et al., 1993) benchmark for object-oriented database management systems (DBMS). Firstly, the original relational schema of OO7 was translated into the corresponding DTD using several author-defined mapping rules. The benchmark involves a generator called *genxml* which enables the generation of respective data sets on the basis of user-provided parameters of elements of the DTD. They influence the depth of the document tree (specified by the number of inclusions of a recursive element), fan-out (specified by the number of repetitions of two elements with allowed repeatable occurrence) or the amount of textual data (specified by the size in bytes of content of a textual and a mixed-content element). The authors propose three pre-defined types of data sets (small, medium and large) with pre-defined values of the parameters.

The XOO7 benchmark involves 23 XQuery queries divided into three categories – *relational queries* (involving joins, aggregation, sorting etc.), *document queries* (focussing on ordering of elements) and *navigational queries* (exploiting references and links).

Probably for the first time, the XOO7 benchmark has been used by its authors for analyzing and comparison of a semi-structured XML management system (XML MS) *Lore*, a native XML MS *Kweelt* and a commercial object-relational (OR) DBMS (see Li et al., 2001) and later for comparison of four XML processing tools – Lore, Kweelt, an XML-enabled DBMS *XENA* and a commercial XPath implementation (see Nambiar et al., 2002).

XML Data Management Benchmark (XMach-1)

XML benchmark *XMach-1* (Bohme et al., 2001) differs from the previously mentioned ones especially in the fact that it is a mult-iuser benchmark. In the previous cases, the authors assumed that the tested XML MSs were run on the same machine, so that network characteristics, communication costs, numbers of users were of no importance. In this case, the benchmark is based on the idea of a web application, i.e. a typical use case of XML MS. It consists of four parts – an XML database, application servers, loaders and browser clients. The application servers support processing of XML documents and interact with the backend XML database. The loaders load and delete various XML data into/from the database via the application servers. And it is assumed that browser clients will query and retrieve the stored XML data. Therefore, the tested systems are represented via the application servers and the database. The query and upload workload is generated by virtual browsers and loaders whose number is arbitrary.

Similarly to the previous cases, the benchmark involves a data generator and a set of XQuery queries. The data generator can prepare (and store into the database) either schema-less documents or documents conforming to a pre-defined DTD. However, the schema-less documents differ only in the fact that the DTD is not maintained in the database. Also, multiple data collections can be generated, but they differ only in the element/attributes names. Similarly to the previous cases, a user can specify various characteristics of the data, such as the number of documents per a DTD, number of occurrences of four elements of the DTD, probability of occurrence of phrases and links, number of words in a sentence etc.

The text values are generated from 10,000 most common English words distributed according to Zipf's law.

The benchmark queries involve 8 XQuery queries and, for the first time, also 3 data manipulation operations. The queries involve similar cases as in the previous cases, such as reconstruction of the whole document, text retrieval query, navigation through document tree, counting, sorting, joining etc. The data manipulation operations involve inserting a document into the database, deleting a document from the database and updating information in the directory entry of stored documents.

The authors' experience with the benchmark and performance results of comparison of two commercial native XML DBMSs and one relational DBMS are described in paper (Bohme et al., 2003). Later, the authors of paper (Lu et al., 2005) used both XMark and XMach-1 and analyzed the performance of nine different implementations of XML DBMS involving three native and six relational approaches.

Note that since the three benchmarks, i.e. XMark, XOO7 and XMach-1, appeared almost at the same time, naturally their properties were also soon compared and contrasted (Nambiar et al., 2001). The paper focuses mainly on comparison of similar and distinct types of queries of the benchmarks with regard to generally acknowledged desired properties of an XML query language.

The Michigan Benchmark (MBench)

Contrary to the previously described *application-level* benchmarks, the *Michigan Benchmark* (Runapongsa et al., 2006) (in literature often denoted as *MBench*) is a *micro-benchmark*. The basic ideas are very similar – both types of benchmarks consist of a data set and related queries – but an application benchmark is created to help users to compare and contrast various applications, whereas a micro-benchmark should be used to evaluate the performance of a single system in various situations.

Since the aim of the benchmark is different, the data set and the set of queries also strongly differ. The data set is generated according to a synthetic XSD (XML Schema definition) which consist of an element having 7 attributes (carrying information about its position in the document tree), a recursive sub-element with arbitrary occurrence and an optional element. Words of text are created synthetically and then distributed according to Zipf's law so that its characteristics are similar to a natural language and not biased by a particular language. Similarly to XMark, the generated data can be influenced by the scaling factor which expresses the size of the data.

The set of queries contains 46 queries and 7 update operations. They can be further divided into queries which reconstruct a selected structure, selection queries, join queries and aggregation queries, whereas within the groups they differ only slightly, for instance at the level of selectivity, the type of ordering, the complexity of returned data etc. The paper only describes the queries and, hence, they can be specified in any language. Nevertheless, the authors provide their SQL and XPath formulation.

Paper (Runapongsa et al., 2006) also describes performance results of the benchmark applied on two XML DBMSs and one commercial OR DBMS.

Similarly to the previous case, the four described benchmarks, i.e. XMark, XOO7, XMach-1 and MBench, were compared in paper (Bohme et al., 2003). It focuses mainly on the type of data the benchmarks include, number of involved users and servers, number of documents, schemes and element types, number of queries etc. The aim of the authors is to help users choose the most appropriate of the benchmarks, but the analysis is slightly, though naturally, biased by the fact that it is written by authors of XMach-1.

XBench

XML benchmark *XBench* (Yao et al., 2003) is denoted as a *family of benchmarks* since the authors distinguish four classes of XML applications with different requirements – text-centric/single document (TC/SD), text-centric/multiple documents (TC/MD), data-centric/single document (DC/SD) and data-centric/multiple documents (DC/MD).

For the purpose of generating XML data, the authors provide their own generator which is built on top of the ToXgene data generator (Barbosa et al., 2002) and enables the size of the generated documents to be modified – small (10MB), normal (100MB), large (1GB) and huge (10GB). The structure of the data in each of the four types of applications is based on analysis of several selected real-world XML data or XML database exports, their generalization and derivation of synthetic data on the basis of the results.

The set of XQuery queries covers functionality captured by W3C XML Query Use Cases. Similarly to the previous case the queries are specified abstractly and their XQuery specification is available. All together the authors provide 20 queries, but not all of them can be used in all the four applications. The queries involve similar cases and constructs as in the previous cases, such as exact matching ordering, function application, quantification, path expressions, joins, references, casting etc.

Using the benchmark, the authors have also performed corresponding experimental testing on three commercial DBMSs (Yao et al., 2004).

Similarly to the previous cases, paper (Manegold, 2008) provides an analysis of the five described benchmarks, that is: XMark, XOO7, XMach-1, MBench and XBench, applied on six XQuery processors.

On the other hand, paper (Afanasiev et al., 2006) analyzes the five benchmarks, but with a different aim – not to analyze the benchmarked systems, but the benchmarks themselves. Using four selected XQuery engines, the authors try to answer the following questions: How are the benchmarks currently used? What do the benchmarks measure? And what can we learn from these benchmarks? The key findings and conclusions are very interesting. In particular, the authors have discovered that only 1/3 of papers on XQuery processing use a kind of benchmark which is probably caused by the fact that 38% of benchmark queries are incorrect or outdated. In addition, 29% of the queries are XPath 1.0 queries, 61% are XPath 2.0 queries and only 10% cannot be expressed in XPath. The most popular benchmark seems to be the XMark benchmark.

It is important to note that the results of both of the papers were obtained using the project *XCheck* (Franceschet et al., 2006). It is a platform which enables the execution of multiple benchmarks on multiple query engines and helps to analyze and compare the results. The benchmarks are specified in input documents that describe the queries, the engines and the documents which should be used together. The engines can be easily added using wrapping adapters. Naturally, this is not the only representative of such an application. A very similar, but older platform facilitating XML benchmarking is system BumbleBee (BumbleBee, 2003).

XPathMark

The XML benchmark *XPathMark* (Franceschet, 2005) was designed for XML documents generated using XMark benchmark, but having the queries expressed in XPath 1.0. The benchmark has two parts consisting of a default document and a set of related queries. The former contains an XML document generated using XMark and a set of 47 XPath queries that focus on axes, node tests, Boolean operators,

references and functions. The latter contains an XML document taken from a book on XML, whereas the related 12 queries focus on comments, processing instructions, namespaces and language attributes.

Paper (Franceschet, 2005) also involves results of experimental testing of two XML engines – *Saxon* and *Galax* – using XPathMark.

MemBeR:XQuery Micro-Benchmark Repository

From the above overview of the benchmarks and their various features, it is obvious that a single fixed set of queries is insufficient for the testing of various aspects of applications. Hence, the main aim of the *MemBeR repository* (Afanasiev et al., 2005) of micro-benchmarks is to allow users to add new data sets and/or queries for specific performance assessment tasks. The authors focus particularly on micro benchmarks, because of their scarcity (from the above-described best-known representatives only the MBench benchmark can be considered as a true micro-benchmark suite) and the huge amount of XML query features which need to be tested from various perspectives.

The repository has a predefined structure involving XML documents and their parameters, XML queries and their parameters, experiments and their parameters (i.e. related documents and/or queries), micro-benchmarks (i.e. sets of experiments) and micro-benchmark result sets. A new micro-benchmark or a new result set must be specified as an XML document conforming to a pre-defined DTD which describes all the related characteristics.

Currently, the repository contains three categories of benchmarks – XPath, query stability and XQuery. The benchmarks can be further classified (Afanasiev et al., 2005) into performance, consumption, correctness and completeness benchmarks on the basis of the resulting metric, type of scalability (data/query), usage of schema, query processing scenarios (e.g. persistent database, streaming etc.), query language and tested language feature.

One of the micro-benchmarks has been used in paper (Manolescu et al., 2006) for a very detailed analysis of four constructs of XQuery – XPath navigation, XPath predicates, XQuery FLWORs and XQuery node constructions – in six best-known freely available systems such as, *eXist*, Galax, MonetDB etc.

Transaction Processing over XML (TPoX)

Project *TPoX* (Nicola et al., 2007) seems to be the most recent XML query benchmark. It is an application-level benchmark simulating a financial multi-user application scenario based on the authors' real-world experience. In contrast to most of the previous cases, it does not focus on XQuery processing, but rather on other performance-relevant database features such as logging, indexing, schema validation, update operations, concurrency control, transaction processing etc. The main idea and architecture of the project are very similar to those of the XMach-1 project. The main differences are that the data set is data-centric (XMach-1 contains document-centric documents), the number of documents is several times higher than in XMach-1 and while XMach-1 enables to generate multiple synthetic DTDs, TPoX involves a single XSD consisting of multiple related subschemes all together describing the financial application.

The documents in the data set are again generated using the ToXgene data generator according to the XSD. The application can be scaled from extra small (XS) representing 3.6 millions of documents (approximately 10GB of data) and 10 users to extra-extra large representing 360 billions of documents (approximately 1PB of data) and 1 million users.

The operations over the database are divided into two stages. Stage 1 performs concurrent inserts, whereas stage 2 performs a multi-user read/write workload consisting of 70% of queries and 30% of

updates. The operations are divided into 17 real-world transactions which are randomly submitted by Java threads, each representing a single user. Since most of the features of the system can be controlled via parameters, it can be even set to query-only, single-user, single-document system and, hence, compared with the other benchmarks.

Paper (Nicola et al., 2007) describes not only the TPoX project itself, but also the authors' first experience with applying the benchmark on DB2 database and its XML support.

For better clarity, we conclude this section with an overview of the main characteristics of the existing XML query benchmarks as listed in Table 1. As we have mentioned, there are also papers which compare and contrast various subsets of the benchmarks in more detail. Hence, we do not repeat the information in this paper and refer an interested reader to them.

Table 1. Main characteristics of XML query benchmarks

	XMark	**XOO7**	**XMach-1**	**MBench**	**XBench**	**XPathMark**	**TPoX**
Type of benchmark	Application-level	Application-level	Application-level	Micro	Application-level	Application-level	Application-level
Number of users	Single	Single	Multiple	Single	Single	Single	Multiple
Number of applications	1	1	1	1	4	1	1 but complex
Documents in data set	Single	Single	Multiple	Single	Single/ multiple	Single	Multiple
Data generator	✓	✓	✓	✓	✓	✓	✓
Key parameters	Size	Depth, fan-out, size of textual data	Number of documents / elements / words in a sentence, probability of phrases / links	Size	Size	Size	Size + number of users
Default data set	Single 100MB document	3 documents (small, medium, large) with pre-defined parameters	4 data sets of 10000 / 100000 / 1000000 / 10000000 documents	Single document with 728000 nodes	Small (10MB) / normal (100MB) / large (1GB) / huge (10GB) document	1 XMark document and 1 sample document from a book	XS (3.6 millions of documents, 10 users), S, M, L, XL, XXL (360 billions of documents, 1 million users)
Schema of documents	DTD of an Internet auction database	DTD derived from OO7 relational schema	DTD of an document having chapters, paragraphs and sections	DTD / XSD of the recursive element	DTD / XSD	DTD	XSD
Number of schemes	1	1	Multiple	9	1	2	1 consisting of multiple
Number of queries	20	23	8	49	19, 17, 14, 16	47 + 12	7
Query language	XQuery	XQuery	XQuery	SQL, XPath	XQuery	XPath	XQuery
Number of update queries	0	0	3	7	0	0	10

And, finally, Table 2 provides an overview of papers which describe results of analyses of testing various systems using subsets of the benchmarks. The papers usually not only analyze the systems being tested, but they also compare and contrast features of the benchmarks.

Table 2. Subsets of benchmarks used for testing various systems

	XMark	XOO7	XMach-1	MBench	XBench	XPathMark	MemBeR	TPoX
Schmidt et al., 2001	✓	✗	✗	✗	✗	✗	✗	✗
Li et al., 2001; Nambiar et al., 2002	✗	✓	✗	✗	✗	✗	✗	✗
Bohme et al., 2003	✗	✗	✓	✗	✗	✗	✗	✗
Lu et al., 2005	✓	✗	✓	✗	✗	✗	✗	✗
Nambiar et al., 2001	✓	✓	✓	✗	✗	✗	✗	✗
Runapongsa et al., 2006	✗	✗	✗	✓	✗	✗	✗	✗
Bohme et al., 2003	✓	✓	✓	✓	✗	✗	✗	✗
Yao et al., 2004	✗	✗	✗	✗	✓	✗	✗	✗
Afanasiev et al., 2006; Manegold, 2008	✓	✓	✓	✓	✓	✗	✗	✗
Franceschet, 2005	✗	✗	✗	✗	✗	✓	✗	✗
Manolescu et al., 2006	✗	✗	✗	✗	✗	✗	✓	✗
Nicola et al., 2007	✗	✗	✗	✗	✗	✗	✗	✓

The table depicts a natural progress in papers dealing with exploitation and comparison of existing approaches. Firstly, there are papers involving tests of various selected implementations using a single, new benchmark. Later, papers emerge which perform the testing using multiple benchmarks and, hence, compare their features. As we can see, the biggest subset of compared benchmarks involves XMark, XOO7, XMach-1, MBench and XBench. For the three newest benchmarks, i.e. XPathMark, MemBerR micro benchmarks and TPoX, the respective comparison does not yet exist.

Other XML Benchmarking Projects

Since the key aspects of XML processing are undoubtedly parsing, validating and querying, most of the existing benchmarking projects focus mainly on them. But, there are also other popular and useful XML technologies, such as, e.g. XSL transformations (Clark, 1999) and, hence, there occur also benchmarks determined for other purposes, though their number is surprisingly low or the only existing representatives are quite dated and, hence, obsolete (e.g. the XSLTMark (Kuznetsov et al., 2000) benchmark for XSLT).

From one point of view this may be caused by the fact that most of other technologies, such as, e.g., XSLT, XPointer (DeRose et al., 2002), XLink (DeRose et al., 2001) etc., are based on one of the basic ones, mostly XPath queries. Thus, an argument against new special benchmarking projects may be that projects for benchmarking XML queries in general are sufficient enough. But, on the other hand, the exploitation of, for instance, XPath in XSL can be very different from typical exploitation in XML DBMS. And, in addition, there are important aspects of XSL transformations other than the path queries which influence their correctness and efficiency. Furthermore, if we consider even more special operations on XML data, such as, e.g., XML compressing, the respective benchmark may deal with features which are for other types of XML processing marginal. Hence, the argument for special benchmarks seems to be much stronger. However, the number of these special purpose benchmarks is still low – it is difficult to find at least a single representative for each of the areas.

On the other hand, there are XML technologies that have become popular only recently and, consequently, their benchmarking projects are relatively rare. A representative of this situation is XML updating. As we can see in Table 1, some of the existing query benchmarks involve few update operations, but a true XML update benchmarking project has been proposed only recently (Phan et al., 2008).

3. SUMMARY

We can sum up the state of the art of existing XML benchmarking projects with the following natural, but important findings:

- Probably the most typical source of benchmarking XML data are repositories with fixed, usually real-world XML data. Their two main disadvantages are that the real-world XML data are usually too simple to cover all possible XML constructs, and they are not accompanied by respective operations, e.g., queries, updates, transformations etc.
- Since parsing and validating are two most important basic operations with XML data, the W3C consortium has defined appropriate conformance test suites which enable the testing of their correct behaviour. Hence, this area of benchmarks is well defined.
- While the conformance to W3C specifications is a natural and expected feature of XML parsers and validators, the key aspect of users' interest is their efficiency. Although there exist several papers and projects dealing with this topic, there seems to be no true benchmark involving testing data sets and queries that would cover all or, at least, the key influencing aspects.
- The second key operation on XML data is undoubtedly querying. Not only is it the way to access stored data using various approaches, but path queries are an important part of various other XML technologies, such as XSLT, XPointer, XLink etc. Hence, the related benchmarks form the most important subset of all related benchmarking projects.
- The authors of the existing query benchmarks tried to address as many aspects of the related language (e.g. XQuery, XPath etc.) as possible. But since most of the benchmarks originated at the time when specifications of XML query languages were as yet unfinished, most of them soon became obsolete. Either the syntax of queries was no longer correct, or the respective languages now support plenty of other, at that time unknown, constructs.
- Most of the query benchmarks naturally focus on the XQuery language which involves the XPath query language. But, probably none of the benchmarks is able to test all the respective aspects. Also, there seems to be no benchmark which focuses on differences of XPath 1.0 and XPath 2.0.
- Although all the benchmarking projects involve a kind of data generator, the most popular ones seem to be those which are of simple usage (e.g. XMark), i.e. having only few parameters to specify. On the other hand, these benchmarks usually provide only very simple data, of one special type and complexity.
- In addition, most of the benchmarks are for query-only, single-user and single-document. There is only one benchmark (XBench) which takes into account several possible scenarios of applications (single vs. multiple documents and data-centric vs. document-centric documents), but it is a single-user benchmark. There are two benchmarks (XMach-1 and TPoX) which are multi-user, but, at the same time, the number of related queries is low and the data sets are quite simple.In general, the area of query benchmarks is relatively wide and the projects usually try to cover the key query operations. But if we consider other XML technologies which involve path queries, the

typical usage can strongly differ. Hence, these technologies require special treatment and special benchmarking projects. Surprisingly, in these areas, the number of respective benchmarks is surprisingly low. Mostly, no appropriate benchmarking project exists.

4. OPEN ISSUES

Although each of the existing approaches contributes certain interesting ideas and optimizations, there is still room for possible future improvements. We describe and discuss them in this section.

General Requirements for Benchmarks

As mentioned in (Bohme et al., 2001), the recommended requirements for database benchmarks are that they should be domain-specific, relevant (measuring the performance of typical operations for the respective domain), portable to different platforms, scalable (applicable to small and large computer systems) and simple. But, for the purpose of XML technologies, not all of these requirements are necessary.

Portability and scalability are natural requirements which do not restrict the set of future users except for those using a selected hardware and/or operating system. Simplicity seems to be an important requirement too, although it may be sometimes be acquired only at the cost of restricted functionally. Nevertheless, as we have already mentioned, currently the most exploited benchmark seems to be XMark which involves only a fixed set of XML queries and the only parameter of the data is their size in Bytes. It confirms the importance of this requirement and indicates that since the researchers have already spent plenty of time proposing, improving and implementing their approach, they do not want to bother with a complicated benchmark system.

On the other hand, the question of domain-specificity and related relevancy is arguable. Since XML technologies have currently plenty of usages and almost every day new ones emerge, it is difficult if not impossible, to specify a benchmark which covers all of them. But, on the other hand, a benchmark which is restricted only to a single special use case cannot be very useful. We can also specify more general types of XML applications, such as the classical data-centric and document-centric, but their characteristics are still too general. Hence, a solution seems to be a versatile benchmarking project which can be highly parameterized and, at the same time, be extended to novel characteristics. On the other hand, it should involve an extensible set of pre-defined settings of the parameters which characterize particular applications.

More Sophisticated Data Generator

A natural first step towards obtaining the versatile XML benchmark is to exploit a more sophisticated data generator. The existing benchmarks use either a simple data generator, or rather, a modifier of the stored data that supports only a simple set of parameters. Sometimes they are built on top of a more complex data generator (the ToXgene generator seems to be the most popular one), but most of its characteristics are then fixed due to the fixed set of related XML queries. The generators usually deal with marginal problems such as where to get the textual data or element/attribute names to achieve as natural a result as possible; whereas, the set of characteristics which influence the structure or semantics of the data is usually trivial. For some applications (such as XML full-text operations or XML compression), it may be

that the content of textual data is important, but for most of the techniques related to XML querying, these aspects are of marginal importance, whereas the structure and semantics of the data are crucial.

The structure of the data is represented by the structure and complexity of trees of XML documents or graphs of XML schemes which consist of multiple types of nodes and edges representing the relationships among them. Then the W3C recommendations specify the allowed relationships, i.e. positions of the nodes within the tree. On the other hand, the semantics of the data is specified mostly by data types, unique/key/foreign key constraints and related functional dependencies. All of these characteristics (i.e. their amount, position and complexity) can be specified by a user and, hence, the respective system can generate any kind of data. The basic characteristics of XML documents can result from characteristics analyzed in existing statistical analyses of real-world XML data (Mlynkova et al., 2006) such as, e.g., the amount and size (in bytes) of XML documents, depth of XML documents, fan-out of elements (i.e. the number of subelements and attributes), percentage of various XML constructs (such as mixed-content elements, attributes etc.) etc. More complex characteristics can be specified, such as the statistical distribution a selected aspect should have (e.g. the depth of output documents).

But, as we have mentioned, this idea is at odds with the requirement of simplicity of benchmarks, because it requires plenty of user interaction. Nevertheless, this problem can easily be solved using predefined settings of parameters which specify various applications. Furthermore, such information can be extracted from statistical analyses of real world XML data. Assuming that the set is extensible and publicly available, each user can either exploit an existing data set or specify own types of data on which the particular system was tested. Moreover, according to the parameters, a data set with the same characteristics can be generated again and, hence, a new approach can easily be compared with existing ones.

Schema Generator

A natural requirement for a generator of XML documents is to provide also the respective XML schema of the resulting data or their selected subset. This problem can be viewed from two different perspectives depending on the order of generating the data.

If the generator first creates XML documents, we can exploit and/or utilize techniques for automatic inference of an XML schema from a given set of XML documents (e.g. Vosta et al., 2008). These approaches usually start with a schema that accepts exactly the given XML documents and they generalize it using various rules (such as, e.g., “if there are more than three occurrences of an element, it is probable that it can occur an arbitrary number of times”). Since there are multiple possibilities for how to define such rules, they can be restricted by user-specified parameters as well. Furthermore, if we consider the XML Schema language which involves plenty of “syntactic sugar”, i.e. sets of constructs which enable the specification of the same situation in various ways (such as, e.g., references vs. inheritance), we discover another large area of data characteristics that can be specified by a user.

On the other hand, if the generator first generates (or obtains an input) the XML schema, the characteristics of the respective instances (i.e. XML documents) are quite restricted. However, an XML schema naturally involves vague specification of the document structure – extensive examples can be * operator or recursion which allow infinitely wide or deep XML documents. Hence, a user can specify these characteristics more precisely. A similar approach has already been exploited in the ToXgene generator, where the input XSD together with a predefined set of annotations specifies the demanded XML data. On the other hand, the annotations either only express the data characteristics more precisely (e.g. maximum length of a text value of an element, minimum and maximum value of a numeric data type etc.) or they express data features which cannot be expressed in XML Schema language (e.g. the

probability distributions of various numeric values – numbers of occurrences, lengths etc.). Hence, in fact, the system simply enables the schema of the target documents to be specified more precisely. In some situations, this exact specification may be useful, but for the purpose of benchmarking, this system requires information that is too precise and, hence, not user-friendly.

Similarly to the previous case, since the amount of input parameters of the data may be quite high in both cases, there should exist respective pre-defined settings which characterize real-world XML data or various reasonable testing sets.

Query Generator

A natural third step of data generation is the generation of XML queries. All the described and probably all the existing works involve a fixed set of queries. The problem is that a fixed set of queries highly restricts the data sets, since, naturally, the queries are expected to query over the data we are provided with and return a reasonably complex result. But, similarly to the previous case, we may assume that a user knows what characteristics the queries over the tested system should have, but their manual creation is again quite a demanding task. Hence, a system, that is able to generate a set of queries with the respective characteristics would, undoubtedly, be useful.

We can again find plenty of characteristics that a query can have. Apart from the constructs that can be used in the query (e.g. axes, predicates, constructors, update operations etc.), we can specify the kind of data that the query should access (e.g. attributes, keys and foreign keys, mixed-content elements, recursive elements etc.), where the data are located (e.g. at what levels), the amount of data that is required (e.g. elements with specified structure) etc. In general, this problem seems to be the most complex, least explored and most challenging open issue of XML benchmarking.

Theoretic Study of Data Characteristics

All three types of the previously specified data generators have one thing in common. If we assume that our aim is to support as much data characteristics as possible, we can find out that various subsets of the data are correlated, i.e. influence each other and, hence, not all possible settings are available. Simple examples can include the length of attribute values and/or element contents vs. size of the document in Bytes or number of elements vs. size of the document in Bytes. More complex examples may include the depth of the document vs. element fan-out vs. size of the document in Bytes. A theoretic study of the data characteristics, their classification and, in particular, a discussion of how they mutually influence each other would be a very useful source of information.

For instance, the MemBeR XML generator (Afanasiev et al., 2005) solves this problem using brute force and does not allow the specifying of depth, fan-out and size simultaneously. But, naturally, this solution seems to be too restrictive.

Analysis and Visualization of the Resulting Data

An interesting part of a benchmarking project closely related to data generators may be a statistical analyzer of the resulting synthetic data. If we assume that a user specifies general characteristics of the data, he/she may be interested in the exact metrics of the result. And, on the other hand, sometimes it may be useful to include a subset of real-world XML data and, consequently, the analysis of their complexity becomes even more crucial. As we have outlined in the introduction, without knowing the structure of the data, it is difficult to conclude whether the tested system would be useful for the future user.

A related logical part of the analyzer may be also a data visualizer designed particularly for the purpose of XML data. Most of the existing implementations which involve a kind of XML visualization support only a simple tree structure. For simple XML data such as small XML documents and non-recursive XML schemes with a low number of shared elements, this may be sufficient. However, XML documents may be relatively large or they may be separated into a huge number of smaller documents, whereas XML schemes may involve a significant portion of recursion, complete subgraphs etc.(and statistical analyses show that in real-world data these situations are quite common (Mlynkova et al., 2006)). Hence, a sophisticated visualizer which is able to parse and display such complex kinds of data in the form of a graph may be a very useful tool. A similar problem has been solved in paper (Dokulil et al., 2008) which addresses the issue of visualisation of large RDF data.

Other Areas of Exploitation

The synthetic XML data (of all kinds) can be exploited for purposes other than benchmarking of different algorithms and their implementations. One of the most promising areas is undoubtedly that of e-learning. Automatically generated data can be used for the purpose of tests and quizzes, where a huge number of distinct examples with similar, pre-defined characteristics is necessary and their manual creation is a demanding process. This general idea can be easily exploited in XML technologies as well. A similar system is proposed in paper (Azalov et al., 2003) which enables the generation of synthetic source codes.

5. CONCLUSION

The main goal of this paper was to describe and discuss the current state of the art and open issues of XML benchmarking projects, i.e. projects focussing on benchmarking of various XML processing tools such as XML parsers and validators, XML data management systems etc. Firstly, we have provided several motivating examples justifying the importance of XML benchmarking as a topic. Then, we have provided an overview and classification of the existing approaches and their features and summed up the key findings. And finally, we have discussed the corresponding open issues and their possible solutions.

Our aim was to show that XML benchmarking is an up-to-date problem. From the overview of the state of the art, we can see that even though there are interesting and inspiring approaches, there is still a variety of open problems which can and need to be solved or improved to enable the development of more informative benchmarking of XML processing tools and newly proposed methods.

ACKNOWLEDGMENT

This work was supported in part by the Czech Science Foundation (GACR), grant number 201/06/0756.

REFERENCES

Aboulnaga, A., Naughton, J. F., & Zhang, C. (2001). Generating Synthetic Complex-Structured XML Data. In *WebDB'01: Proceedings of the 4th International Workshop on the Web and Databases*, pages 79 – 84, Washington, DC, USA.

Afanasiev, L., Manolescu, I., & Michiels, P. (2005). MemBeR: A Micro-Benchmark Repository for XQuery. In *XSym'05: Proceedings of 3rd International XML Database Symposium*, Lecture Notes in Computer Science. Springer-Verlag.

Afanasiev, L., & Marx, M. (2006). An Analysis of the Current XQuery Benchmarks. In *ExpDB'06: Proceedings of the 1st International Workshop on Performance and Evaluation of Data Management Systems*, pages 9 – 20, Chicago, Illinois, USA. ACM Press.

Azalov, P., & Zlatarova, F. (2003). SDG – A System for Synthetic Data Generation. In *ITCC'03: Proc of the International Conference on Information Technology: Computers and Communications*, pages 69 – 75, Washington, DC, USA. IEEE Computer Society.

Barbosa, D., Mendelzon, A. O., Keenleyside, J., & Lyons, K. A. (2002). ToXgene: A Template-Based Data Generator for XML. In *SIGMOD'02: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, page 616, Madison, Wisconsin, USA. ACM Press.

Barbosa, D., Mignet, L., & P. Veltri (2005). Studying the XML Web: Gathering Statistics from an XML Sample. *World Wide Web*, 8(4), pages 413 – 438.

Berglund, A., Boag, S., Chamberlin, D., Fernandez, M. F., Kay, M., Robie, J., & Simeon, J. (2007). *XML Path Language (XPath) 2.0*. W3C. http://www.w3.org/TR/xpath20/.

Bex, G. J., Neven, F., & Van den Bussche, J. (2004). DTDs versus XML Schema: a Practical Study. In *WebDB'04: Proceedings of the 7th International Workshop on the Web and Databases*, pages 79 – 84, New York, NY, USA. ACM Press.

Biron, P. V., & Malhotra, A. (2004). *XML Schema Part 2: Datatypes (Second Edition)*. W3C. www.w3.org/TR/xmlschema-2/.

Boag, S., Chamberlin, D., Fernandez, M. F., Florescu, D., Robie, J., & Simeon, J. (2007). *XQuery 1.0: An XML Query Language*. W3C. http://www.w3.org/TR/xquery/.

Bohme, T., & Rahm, E. (2001). Benchmarking XML Database Systems – First Experiences. In *HPTS'01: Proceedings of 9th International Workshop on High Performance Transaction Systems*, Pacific Grove, California.

Bohme, T., & Rahm, E. (2001). XMach-1: A Benchmark for XML Data Management. In *BTW'01: Datenbanksysteme in Buro, Technik und Wissenschaft*, 9. GI-Fachtagung, pages 264 – 273, London, UK. Springer-Verlag.

Bohme, T. & Rahm, E. (2003). Multi-User Evaluation of XML Data Management Systems with XMach-1. In *Proceedings of the VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb on Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web-Revised Papers*, pages 148 – 158, London, UK. Springer-Verlag.

Bosak, J. (2007). *Jon Bosak's XML examples*. http://www.ibiblio.org/bosak/.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (2006). *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. W3C. http://www.w3.org/TR/REC-xml/.

Bressan, S., Lee, M.-L., Li, Y. G., Lacroix, Z., & Nambiar, U. (2003). The XOO7 Benchmark. In *Proceedings of the VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb on Efficiency and*

Effectiveness of XML Tools and Techniques and Data Integration over the Web-Revised Papers, pages 146 – 147, London, UK. Springer-Verlag.

BumbleBee (2003). Clarkware Consulting, Inc. and Hunter Digital Ventures, LLC. http://www.x-query.com/bumblebee/.

Busse, R., Carey, M., Florescu, D., Kersten, M., Manolescu, I., Schmidt, A., & Waas, F. (2003). *XMark – An XML Benchmarking project*. Centrum voor Wiskunde en Informatica (CWI), Amsterdam. http://www.xml-benchmark.org/.

Carey, M. J., DeWitt, D. J., & Naughton, J. F. (1993). The OO7 Benchmark. *SIGMOD Record* (ACM Special Interest Group on Management of Data), 22(2), pages 12 – 21.

Chamberlin, D., Fankhauser, P., Florescu, D., Marchiori, M., & Robie, J. (2007). *XML Query Use Cases*. W3C. http://www.w3.org/TR/xquery-use-cases/.

Chilingaryan, S. A. (2004). *XML Benchmark*. http://xmlbench.sourceforge.net/.

Clark, J. (1999). *XSL Transformations (XSLT) Version 1.0*. W3C. http://www.w3.org/TR/xslt.

Clark, J., & DeRose, S. (1999). *XML Path Language (XPath) Version 1.0*. W3C. http://www.w3.org/TR/xpath/.

Cooper, C. (1999). *Benchmarking XML Parsers*. XML.com. http://www.xml.com/pub/a/Benchmark/article.html.

DBLP (2008). *Digital Bibliography & Library Project*. http://dblp.uni-trier.de/.

DeRose, S., Daniel, R., Grosso, P., Maler, E., Marsh, J., & Walsh, N. (2002). *XML Pointer Language (XPointer)*. W3C. http://www.w3.org/TR/xptr/.

DeRose, S., Maler, E., & Orchard, D. (2001). *XML Linking Language (XLink) Version 1.0*. W3C. http://www.w3.org/TR/xlink/.

Dokulil, J., & Katreniakova, J. (2008). Visual Exploration of RDF Data. In *SOFSEM'08: Proceedings of the 34th International Conference on Current Trends in Theory and Practice of Computer Science*, pages 672 – 683, Novy Smokovec, Slovakia. Lecture Notes in Computer Science, volume 4910. Springer-Verlag.

Farwick, M. & Hafner, M. (2007). *XML Parser Benchmarks: Part 1 & 2*. XML.com. http://www.xml.com/pub/a/2007/05/09/xml-parser-benchmarks-part-1.html. http://www.xml.com/pub/a/2007/05/16/xml-parser-benchmarks-part-2.html.

Fields, M. (1996). *Mark Fields's Ebooks*. http://www.assortedthoughts.com/downloads.php.

Franceschet, M. (2005). XPathMark – An XPath Benchmark for XMark Generated Data. In *XSym'05: Proceedings of 3rd International XML Database Symposium*, Lecture Notes in Computer Science. Springer-Verlag.

Franceschet, M., Zimuel, E., Afanasiev, L., & Marx, M. (2006). *XCheck*. Informatics Institute, University of Amsterdam, The Netherlands. http://ilps.science.uva.nl/Resources/XCheck/.

FreeDB (2008). http://www.freedb.org/.

H-InvDB (2007). *Annotated Human Genes Database*. http://www.jbirc.aist.go.jp/hinv/.

Ibiblio (2008). *The Public's Library and Digital Archive*. http://www.ibiblio.org/.

IMDb (2008). *The Internet Movie Database*. http://www.imdb.com/.

INEX (2007). *INitiative for the Evaluation of XML Retrieval*. http://inex.is.informatik.uni-duisburg.de/.

Kumar, P. (2002). *XPB4J – XML Processing Benchmark for Java*. http://www.pankaj-k.net/xpb4j/.

Kuznetsov, E., & Dolph, C. (2000). *XSLT Processor Benchmarks*. XML.com. http://www.xml.com/pub/a/2001/03/28/xsltmark/index.html.

Li, Y. G., Bressan, S., Dobbie, G., Lacroix, Z., Lee, M. L., Nambiar, U., & Wadhwa, B. (2001). XOO7: Applying OO7 Benchmark to XML Query Processing Tool. In *CIKM'01: Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 167 – 174, New York, NY, USA. ACM Press.

Lu, H., Yu, J. X., Wang, G., Zheng, S., Jiang, H., Yu, G., & Zhou, A. (2005). What Makes the Differences: Benchmarking XML Database Implementations. *ACM Trans. Inter. Tech.*, 5(1), pages 154 – 194.

Manegold, S. (2008). An Empirical Evaluation of XQuery Processors. *Inf. Syst.*, 33(2), pages 203 – 220.

Manolescu, I., Miachon, C., & Michiels, P. (2006). Towards Micro-Benchmarking XQuery. In *ExpDB'06: Proceedings of the 1st International Workshop on Performance and Evaluation of Data Management Systems*, pages 28 – 39. ACM Press.

Marcus, S. (1999). *Benchmarking XML Parsers on Solaris*. XML.com. http://www.xml.com/pub/a/1999/06/benchmark/solaris.html.

Martinez, S. I., Grosso, P., & Walsh, N. (2008). *Extensible Markup Language (XML) Conformance Test Suites*. W3C. http://www.w3.org/XML/Test/.

MeSH (2008). *Medical Subject Headings*. http://www.nlm.nih.gov/mesh/meshhome.html.

Mlynkova, I., Toman, K., & Pokorny, J. (2006). Statistical Analysis of Real XML Data Collections. In *COMAD'06: Proceedings of the 13th International Conference on Management of Data*, pages 20 – 31, New Delhi, India. Tata McGraw-Hill Publishing Company Limited.

Nambiar, U., Lacroix, Z., Bressan, S., Lee, M.-L., & Li, Y. G. (2001). XML Benchmarks Put To The Test. In *IIWAS'01: Proceedings of the 3rd International Conference on Information Integration and Web-Based Applications and Services*, Linz, Austria.

Nambiar, U., Lacroix, Z., Bressan, S., Lee, M.-L., & Li, Y. G. (2002). Efficient XML Data Management: An Analysis. In *EC-WEB'02: Proceedings of the 3rd International Conference on E-Commerce and Web Technologies*, pages 87 – 98, London, UK. Springer-Verlag.

Nicola, M., Kogan, I., & Schiefer, B. (2007). An XML Transaction Processing Benchmark. In *SIGMOD'07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 937 – 948, New York, NY, USA. ACM Press.

Open Directory Project (2004). http://rdf.dmoz.org/.

Oren, Y. (2002). *SAX Parser Benchmarks*. SourceForge.net. http://piccolo.sourceforge.net/bench.html.

Phan, B. V. & Pardede, E. (2008). Towards the Development of XML Benchmark for XML Updates. In *ITNG'08: Proceedings of the 5th International Conference on Information Technology: New Generations*, pages 500 – 505, Las Vegas, Nevada, USA. IEEE Computer Society.

RNAdb (2005). http://research.imb.uq.edu.au/rnadb/.

Rorke, M., Muthiah, K., Chennoju, R., Lu, Y., Behm, A., Montanez, C., Sharma, G., & Englich, F. (2007). *XML Query Test Suite*. W3C. http://www.w3.org/XML/Query/test-suite/.

Runapongsa, K., Patel, J. M., Jagadish, H. V., Chen, Y., & Al-Khalifa, S. (2006). The Michigan benchmark: towards XML Query Performance Diagnostics. *Inf. Syst.*, 31(2), pages 73 – 97.

Runapongsa, K., Patel, J. M., Jagadish, H. V., Chen, Y., & Al-Khalifa, S. (2006) *The Michigan benchmark: towards XML Query Performance Diagnostics (Extended Version)*. http://www.eecs.umich.edu/db/mbench/mbench.pdf.

Schmidt, A. R., Waas, F., Kersten, M. L., Florescu, D., Carey, M. J., Manolescu, I., & Busse, R. (2001). Why and How to Benchmark XML Databases. *ACM SIGMOD Record*, 30(3), pages 27 – 32.

Schmidt, A. R., Waas, F., Kersten, M. L., Florescu, D., Manolescu, I., Carey, M. J., & Busse, R. (2001). *The XML Benchmarking project*. Technical Report INS-R0103, CWI, Amsterdam, The Netherlands.

SIGMOD (2007). *SIGMOD Record in XML*. http://www.sigmod.org/record/xml/.

Sosnoski, D. M. (2002). *XMLBench Document Model Benchmark*. http://www.sosnoski.com/opensrc/xmlbench/index.html.

Thompson, H. S., Beech, D., Maloney, M., & Mendelsohn, N. (2004). *XML Schema Part 1: Structures (Second Edition)*. W3C. www.w3.org/TR/xmlschema-1/.

Treebank (1999). *The Penn Treebank Project*. http://www.cis.upenn.edu/~treebank/.

UniProt (2008). *Universal Protein Resource*. http://www.ebi.uniprot.org/index.shtml.

Vosta, O., Mlynkova, I., & Pokorny, J. (2008). Even an Ant Can Create an XSD. In *DASFAA '08: Proceedings of the 13th International Conference on Database Systems for Advance Applications*, pages 35 – 50, New Delhi, India. Lecture Notes in Computer Science, volume 4947, Springer-Verlag.

VTD-XML (2003). *Benchmark Report for Version 2.0*. XimpleWare. http://www.ximpleware.com/2.0/benchmark_2.0_indexing.html.

Wendover, A. (2001). *Arthur's Classic Novels*. http://arthursclassicnovels.com/.

XDR (2002). *XML Data Repository*. www.cs.washington.edu/research/xmldatasets/www/repository.html.

Yao, B. B., Ozsu, M. T., & Keenleyside, J. (2003). XBench – A Family of Benchmarks for XML DBMSs. In *Proceedings of the VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb on Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web-Revised Papers*, pages 162 – 164, London, UK. Springer-Verlag.

Yao, B. B., Ozsu, M. T., & Khandelwal, N. (2004). XBench Benchmark and Performance Testing of XML DBMSs. In *ICDE'04: Proceedings of the 20th International Conference on Data Engineering*, pages 621 – 632, Washington, DC, USA. IEEE Computer Society.