XML Benchmarking: Limitations and Opportunities
(Technical Report)

Irena Mlýnková

Charles University
Faculty of Mathematics and Physics
Department of Software Engineering
Malostranske nam. 25
118 00 Prague 1, Czech Republic
Email: irena.mlynkova@mff.cuni.cz

Abstract. Since XML technologies have become a standard for data representation, a huge amount of methods for processing XML data occurs every day. Consequently, it is necessary to compare the newly proposed methods with the existing ones, as well as to analyze the behavior of a particular method on various types of data. In this paper we provide an overview of existing approaches to XML benchmarking from the point of view of various applications and we show that the problem has been highly marginalized so far. Therefore, in the second part of the paper we discuss persisting open issues and their possible solutions.

1 Introduction

Since XML [40] has become a de-facto standard for data representation and manipulation, there exists a huge amount of methods for efficient managing, processing, exchanging, querying, updating and compressing of XML documents. And new proposals occur every day. Naturally, each author performs various experimental tests of the newly proposed method and describes its advantages and disadvantages. But, if the eventual future user wants to decide which of the existing approaches is for his/hers particular requirements the most suitable, on the basis of the descriptions of methods it can be done very hardly. The problem is that various methods are usually tested on different data sets coming from diverse sources which either do not exist yet 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 source or executable files of the existing method or, at least, identical testing data sets are available. But, neither of the cases is always possible. In addition, in the latter case the performance evaluation is limited by the testing set whose characteristics are often unknown. Hence, a reader can hardly get a notion of the analyzed situation.

An analogous problem occurs if we want to test the behavior of a particular method on various types of data or 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 amount of errors [69] 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 [6]) or projects which enable to generate synthetic XML data on the basis of user-specified characteristics (e.g. project XMark [44]). 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, e.g., the amount and size of the data).

The first aim of this paper 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 especially issues related to their versatility. We will show that the problem of sophisticated XML benchmarking has been so far highly marginalized and the amount of possibilities how to acquire 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 occur also benchmarks determined for other purposes. Nevertheless, their number is surprisingly low or the existing representatives are already obsolete.

Next aim of the paper is to identify the most striking 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 behavior 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 of the 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 system in full generality 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.

The paper 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 Classification

Currently, there exists a number of approaches to experimental testing of XMLMSs and they can be classified variously. In general, a *benchmark* or a *test suite* is a set of

testing scenarios or test cases, i.e. data and related operations which enable to compare versatility, efficiency or behavior of *systems under test* (SUT). In our case the set of data involves XML documents, possibly with their XML schemes, whereas the set of operations can involve any kind of XML-related data operations.

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 reasonable application, the problem is that real-world data are quite simple [31, 33, 69] and do not contain most of the constructs allowed by W3C specifications, whereas benchmarks enabling to test 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 to create them 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 behavior of various types of applications. The most common ones are XML parsers and validators, XML management systems and query evaluators, XSL processors etc. And in particular areas we can establish also more finer classification, e.g., on the basis of exploited languages and constructs, such as, e.g., DTD [40] vs. XML Schema [34, 84] benchmarks, XQuery [35] vs. XPath [49] benchmarks, XPath 1.0 [49] vs. XPath 2.0 [32] benchmarks etc.

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 basic support of XML technologies, i.e. parsing, validating, storing, querying and updating. Naturally, there exist also 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 already out of the scope of this text.

2.1 Fixed Testing Sets of XML Data

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 rather interesting than useful examples of XML documents, such as, e.g., the Bible in XML [8], Shakespeare's plays [7], classic novels in XML [1] etc., the most common types of tested XML data are usually XML exports of various databases, such as, e.g., *IMDb* [5] database of movies and actors, *FreeDB* [3] database of musical CDs, *DBLP* [2] database of scientific papers, *Medical Subject Headings* [9] database of medical terms, *SIGMOD Record* in XML [11] etc., or repositories of real-world XML data provided from various resources, such as, e.g., project *INEX* [6], project *Ibiblio* [14], *Open Directory Project* [12] etc. There also exist examples of rather special XML data, such as, e.g., human genes [4], protein sequences [15], RNAs [10], NASA astronomical data [16], linguistic trees in XML [13] 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, e.g., [16].

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 XML Data Generators

A natural solution to the previous problem is to generate the testing data sets synthetically. Currently, we can find several implementations of XML data generators which generate XML data on the basis of user-provided setting of parameters. Naturally, they can be classified on the basis of the input parameters. The most general classification differentiates so-called *schema-unaware* and *template-based* generators. The schema-unaware generators, such as, e.g., *NiagDataGen* [21], support general structural parameters (e.g. number of levels of the required XML trees, numbers of subelements at each level etc.) and exploit various strategies, such as, e.g., Zip's law, Markov chains, statistical distributions etc., to generate as realistic structure as possible randomly, but within the parameters. On the other hand, the template-based generators, such as, e.g., *ToXgene* [28–30], *VeXGene* [59], *MemBeR* [23], get on input a kind of annotated XML schema and generate XML documents valid against it. Consequently, the structure of the resulting data is specified more precisely, although the full generality of DTD or XML Schema languages is not usually supported. In addition, the annotations provide even more specific information, such as, e.g., probability distributions of occurrences of elements/attributes or lengths of string literals.

Apart from specification of structure of the required data, the generators also often deal with problems such as, e.g., where to get the textual data or element/attribute names to achieve as natural result as possible. For some applications, such as, e.g., XML full-text operations or XML compressing, may be the content of textual data important, but for techniques related to parsing, validating or querying the aspects are of marginal importance.

In general, the biggest advantage of the data generators is that they usually support a huge number of parameters a user can specify and, hence, provide quite a precise result. But, on the other hand, this is also a big disadvantage, because the user must know all these parameters of the required data. And this is of course realistic only in case of XML experts. Similarly to the case of real-world XML data, the synthetic XML data are not accompanied with respective operations as well. In fact, there seems to be no generator of, e.g., XPath queries over the given data having specified features.

2.3 Benchmark Projects for XML Parsers and Validators

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

XML Conformance Test Suites The W3C consortium has naturally provided so-called *XML Conformance Test Suites* [68] – 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 2 000 XML documents which can be divided into two basic types – *binary tests* and *output tests*.

Binary tests contain a set of documents of 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 behavior naturally differs if the tested parser is validating or non-validating.

On the other hand, the output tests enable to test 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, the requirement of their performance evaluation occurred too. 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 built 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 case of push-parsers the reading cannot be influenced, whereas pull-parsers read the next data only if they are "asked" to. And combinations of various parsers have been considered as well.

Currently, there are numerous projects which evaluate efficiency of various subsets of known XML parsers [19, 50, 54, 55, 67, 75] comparing either 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 particularly for the purpose of the benchmark. Although the authors usually make these documents available, there seems to be no true benchmarking project which would enable to analyze all 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 [47, 60, 83]. The sets of the supported applications can usually be extended and also the data sets used for their comparison are available and often extensible. But, the problem is that these projects are not true benchmarking project which would define a set of experiments testing various aspects and especially bottlenecks of XML parsing and validating.

2.4 Benchmark 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

versatility of these tools, i.e. the amount 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, e.g., the type of query language, the amount 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 [81] have discussed and specified the set of challenges 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 in the existing benchmarks.

Note that the W3C XML Query Working Group has proposed and maintains a set of so-called *XML Query Use Cases* [46]. 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. Nevertheless, this set of examples is usually considered as minimum set of operations a reasonable benchmark should cover. On the other hand, the *XML Query Test Suite* (XQTS 1.0.2) [77] contains over 15 000 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 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* [44, 82] is currently one of the most popular and most commonly used XML benchmarks [26]. It involves a data generator called *xmlgen* which enables to create 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 particular system. The textual parts of the resulting XML documents are constructed from 17 000 most frequently occurring words of Shakespeare's plays.

The XMark project involves also 20 XQuery queries [82] 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 behavior and performance of Monet XML framework (see [82]).

XOO7 Benchmark XML benchmark *XOO7* [41–43] is an XML version of the original *OO7* [45] 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 to generate 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 [63]) and later for comparison of four XML processing tools – Lore, Kweelt, an XML-enabled DBMS XENA and a commercial XPath implementation (see [72]).

XML Data Management Benchmark (XMach-1) XML benchmark *XMach-1* [36–38] differs from the previously mentioned ones especially in the fact that it is a multiuser benchmark. In the previous cases the authors assumed that the tested XML MSs are run on the same machine, hence, e.g., network characteristics, communication costs, numbers of users etc. are 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 browser clients are assumed to 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, e.g., number of documents per a DTD, numbers 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, e.g., reconstruction of the whole document, text retrieval query, navigation through document tree, counting, sorting, joining etc. The data manipulation operations involve inserting a document to the database, deleting a document from the database and update of 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 [39]. Later, the authors of paper [64] used both XMark and XMach-1 and analyzed 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 soon compared and contrasted too [70, 71]. 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* [78–80] (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 performance of a single system in various situations.

Since the aim of the benchmark is different, also the data set and the set of queries 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 subelement with arbitrary occurrence and an optional element. Words of text are created syntheticly 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 in 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 [80] 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 [39]. 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 to choose between the benchmarks the most appropriate one, but the analysis is slightly, though naturally, biased by the fact that it is written by authors of XMach-1.

XBench XML benchmark *XBench* [86–88] is 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 of XML data the authors provide their own generator which is built on top of ToXgene data generator and enables to influence the size of the generated documents – 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, e.g., 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 [88].

Similarly to the previous cases, paper [65] provides an analysis of the five benchmarks, i.e. XMark, XOO7, XMach-1, MBench and XBench, applied on six XQuery processors.

On the other hand, paper [26] 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 found out 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 [26, 65] were obtained using the project *XCheck* [22, 58]. It is a platform which enables to execute multiple benchmarks on multiple query engines and helps to analyze and compare the results. The benchmarks are specified in input documents that describe which queries, which engines and which documents should be used together. The engines can be easily added using wrapping adapters. Naturally, this is not the only representative of such application. A very similar, but older platform easing XML benchmarking is system *BumbleBee* [18].

XPathMark The XML benchmark *XPathMark* [56, 57] 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 one 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 one 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 [57] 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 cannot allow testing of various aspects of applications. Hence, the main aim of the *MemBeR repository* [24, 25] 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 lack (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 points of view.

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 [25] 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 [66] 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, e.g., eXist, Galax, MonetDB etc.

Transaction Processing over XML (TPoX) Project *TPoX* [73, 74] seems to be the most recent XML query benchmark. It is an application-level benchmark simulating a financial multi-user application scenario based on authors' real-world experience. Contrary 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 is very similar to 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 [74] describes not only the TPoX project itself, but also authors' first experience with applying the benchmark on DB2 database and its XML support.

For better lucidity, 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.

And finally, Table 2 provides an overview of papers which describe results of analyses of testing various systems using subsets of the benchmarks. The table depicts a natural progress in papers dealing with exploitation and comparison of existing approaches. Firstly, there occur papers involving tests of various selected implementations using a single, new benchmark. Later, there occur also papers 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 exist yet.

2.5 Other XML Benchmark 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 [48] and, hence, there occur also benchmarks determined for other purposes. Surprisingly, the number of such special-purpose projects is low or the only existing representatives are quite old and, hence, obsolete. An example of the situation is benchmarking of XSL transformations. The only known benchmarking project is *XSLTMark* [62] which is not maintained anymore and supports only constructs of version 1.0 from 1999. Similarly, there exist several analytical papers which compare a subset of XSLT processors [17, 20, 61], nevertheless, most of them are based on the obsolete XSLTMark data set or distinct sets of real-world data.

From one point of view this may be caused by the fact that most of other technologies, such as, e.g., XSLT, XPointer [51], XLink [52] 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, e.g., XPath in XSL can be very different from typical exploitation in XML DBMS. And, in addition, there are other

Table 1. Main characteristics of XML DBMS 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
# of users	Single	Single	Multiple	Single	Single	Single	Multiple
# of applications	1	1	1	1	4 (TC/SD, TC/MD, DC/SD, DC/MD)	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	# of documents / elements / words in a sentence, probability of phrases and 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 10 000 / 100 000 / 1 000 000 / 10 000 000 documents	Single document with 728 000 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
# of schemes	1	1	Multiple	9	1	2	1 consisting of multiple
# of queries	20	23	8	49	19, 17, 14, 16	47 + 12	7
Query language	XQuery	XQuery	XQuery	SQL, XPath	XQuery	XPath	XQuery
# of updates	0	0	3	7	0	0	10

	[82]	[63, 72]	[39]	[64]	[70, 71]	[80]	[39]	[88]	[26, 65]	[57]	[66]	[74]
XMark	✓	×	×	✓	✓	×	✓	×	✓	×	×	×
XOO7	×	✓	×	×	✓	×	✓	×	✓	×	×	×
XMach-1	×	×	✓	✓	✓	×	✓	×	✓	×	×	×
MBench	×	×	×	×	×	✓	✓	×	✓	×	×	×
XBench	×	×	×	×	×	×	×	✓	✓	×	×	×
XPathMark	×	×	×	×	×	×	×	×	×	✓	×	×
MemBeR	×	×	×	×	×	×	×	×	×	×	✓	×
TPoX	×	×	×	×	×	×	×	×	×	×	×	✓

Table 2. Subsets of benchmarks exploited for testing various systems

important aspects of XSL transformations 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 amount of these special purpose benchmarks is still low – we can hardly 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 naturally 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 [76].

3 Summary

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

1. 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 that they are not accompanied with respective operations, e.g., queries, updates, transformations etc.
2. A solution to this problem can bring various generators of synthetic XML data. They enable to specify the precise structure of the target data and exploit various approaches to simulate real-world situations. Nevertheless, the problem is that such systems require a user well skilled in XML technologies and, especially, data characteristics. And, naturally, also these data are not accompanied with respective operations as well.
3. Since parsing and validating are two most important basic operations with XML data, the W3C consortium has defined appropriate conformance test suites which enable to test their correct behaviour. Hence, this area of benchmarks is well defined.

4. While the conformance to W3C specifications is a natural and expectable 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.
5. The second key operation on XML data is undoubtedly querying. It is not only the way how to access data stored using various approaches, but path queries are an important part of various other XML technologies, such as, e.g., XSLT, XPointer, XLink etc. Hence, the related benchmarks form the most important subset of all related benchmarking projects.
6. The authors of the existing query benchmarks tried to cover as much 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 not finished yet, most of them become obsolete soon. Either the syntax of queries is not correct anymore or the respective languages now support plenty of other, at that time unknown constructs.
7. 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 would focus on differences of XPath 1.0 and XPath 2.0.
8. 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.
9. In addition, most of the benchmarks are 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 amount of related queries is low and the data sets are quite simple.
10. 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 amount of respective benchmarks is surprisingly low. Mostly there exists no appropriate benchmarking project.

The general observation of our analysis is that the basic XML data operations, i.e. parsing, validating and querying, ale well covered with respective test suites and benchmarking projects. The situation of case of other technologies is much worse. Nevertheless, in these cases we can always exploit either real-world XML data or, if they do not cover our test cases, synthetically generated data sets.

On the other hand, this situation opens a wide research area of both proposing special-purpose benchmarking projects and test suites, as well as performing respective analyses of existing implementations.

4 Open Issues

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

4.1 General Requirements for Benchmarks

As mentioned in [36], 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 only to those using a selected hardware and/or operating system. Simplicity seems to be an important requirement too, although it may be sometimes 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 spent plenty of time proposing, improving and implementing their approach, they do not want to bother with 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 occur, it is hard, maybe even 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 have much usage. We can also specify more general types of XML applications, such as, e.g., the classical data-centric and document-centric, but their characteristics are still too general. Hence, a solution seems to a versatile benchmarking project which can be highly parameterized and, at the same time, extended to novel characteristics. On the other hand, it should involve an extensible set of predefined settings of the parameters which characterize particular applications.

4.2 More Sophisticated Data Generator

A natural first step towards 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, e.g., where to get the textual data or element/attribute names to achieve as natural 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, e.g., XML full-text operations or XML compression) may be the content of textual data 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 [69] 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, e.g., mixed-content elements, attributes etc.) etc. More complex characteristics can specify, e.g., the statistical distribution a selected aspect should have (e.g. the depth of output documents).

But, as we have mentioned, this idea collides 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. And such information can be extracted from statistical analyses of real world XML data as well. 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. What is more, 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.

4.3 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. [85]). 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 arbitrary times"). Since there are multiple possibilities 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 to specify 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 gets on 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 in-

finitely wide or deep XML documents. Hence, a user can specify these characteristics more precisely. A similar approach is already 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 to specify the schema of the target documents more precisely. In some situations this exact specification may be useful, but for the purpose of benchmarking this system requires too precise and, hence, user-unfriendly information.

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

4.4 Query Generator

A natural third step of data generation is generator 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 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 a quite demanding work. 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 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 what kind of data 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), what amount of data 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.

4.5 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 be, e.g., 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 are, e.g., 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 how they mutually influence each other would be a very useful source of information.

For instance, the MemBeR XML generator [23] solves this problem using a brute force and does not allow specifying depth, fan-out and size at the same time. But, naturally, this solution seems to be too restrictive.

4.6 Analysis and Visualization of the Resulting Data

An interesting part of a benchmarking project closely related to data generators may be 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 crucial. As we have outlined in the introduction, without the knowledge of the structure of the data, we can hardly make any conclusions on the tested system that 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, i.e. small XML documents and non-recursive XML schemes with 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 small 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 often [69]). Hence, a sophisticated visualizer which is able to parse and display such complex kind of data in the form of a graph may be a very useful tool. A similar problem has been solved in paper [53] which deals with the problem of visualization of large RDF data.

4.7 Special-Purpose Benchmarks

As we have mentioned, most of the existing benchmarking projects cover basic XML operations, whereas for the advanced ones there can hardly be found a single representative. Hence, an obvious open problem to be solved is a proposal of respective benchmarks for operations such as, e.g., XML transformations, updates, compression etc.

4.8 Other Areas of Exploitation

The synthetic XML data (of all kinds) can have other exploitations than only benchmarking of different algorithms and their implementations. One of the most promising areas is undoubtedly the area of e-learning. Automatically generated data can be used for the purpose of tests and quizzes, where a huge amount 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, e.g., in paper [27] which enables to generate 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, e.g., XML parsers and validators, XML data management systems, XML processors, XML transactions etc. Firstly, we have provided several motivating examples serving as reasons why XML benchmarking is an important topic. Then, we have provided an overview and classification of the existing approaches and their features and summed up the key findings. 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 need to be solved or improved to enable more informative benchmarking of XML processing tools and new methods. Our future work will naturally follow the open issues stated at the end of this paper and especially survey into the possible solutions we have mentioned.

Acknowledgement

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

References

1. *Arthur's Classic Novels*. http://arthursclassicnovels.com/.
2. *DBLP – Digital Bibliography & Library Project*. http://dblp.uni-trier.de/.
3. *FreeDB*. http://www.freedb.org/.
4. *H-InvDB – Annotated Human Genes Database*. http://www.jbirc.aist.go.jp/hinv/.
5. *IMDb – the Internet Movie Database*. http://www.imdb.com/.
6. *Inex – INitiative for the Evaluation of XML Retrieval*. http://inex.is.informatik.uni-duisburg.de/.
7. *Jon Bosak's XML examples*. http://www.ibiblio.org/bosak/.
8. *Mark Fields's ebooks*. http://www.assortedthoughts.com/downloads.php.
9. *Medical Subject Headings*. http://www.nlm.nih.gov/mesh/meshhome.html.
10. *RNAdb*. http://research.imb.uq.edu.au/rnadb/.
11. *SIGMOD Record in XML*. http://www.sigmod.org/record/xml/.
12. *The Open Directory Project*. http://rdf.dmoz.org/.
13. *The Penn Treebank Project*. http://www.cis.upenn.edu/~treebank/.
14. *The Public's Library and Digital Archive*. http://www.ibiblio.org/.
15. *UniProt (Universal Protein Resource)*. http://www.ebi.uniprot.org/index.shtml.
16. *XML Data Repository*. www.cs.washington.edu/research/xmldatasets/www/repository.html.
17. *XSLT Benchmark*. Caucho Technology, Inc., 2001. http://www.caucho.com/articles/xslt-benchmark.xtp.

18. *BumbleBee*. Clarkware Consulting, Inc. and Hunter Digital Ventures, LLC., 2003. http://www.x-query.com/bumblebee/.
19. *VTD-XML Benchmark Report for Version 2.0*. XimpleWare, 2003. http://www.ximpleware.com/2.0/benchmark_2.0_indexing.html.
20. *XSLT Benchmarks*. Ambrosoft, Inc., 2006. http://www.ambrosoft.com/benchmarks.htm.
21. A. Aboulnaga, J. F. Naughton, and C. Zhang. Generating Synthetic Complex-Structured XML Data. In *WebDB'01: Proc. of the 4th Int. Workshop on the Web and Databases*, pages 79–84, Washington, DC, USA, 2001.
22. L. Afanasiev, M. Franceschet, M. Marx, and E. Zimuel. XCheck: A Platform for Benchmarking XQuery Engines. In *VLDB'06: Proc. of the 32nd Int. Conf. on Very Large Data Bases*, pages 1247–1250. ACM, 2006.
23. L. Afanasiev, I. Manolescu, and P. Michiels. *MemBeR XML Generator*. http://ilps.science.uva.nl/Resources/MemBeR/member-generator.html.
24. L. Afanasiev, I. Manolescu, and P. Michiels. *MemBeR: XQuery Micro-Benchmark Repository*. http://ilps.science.uva.nl/Resources/MemBeR/index.html.
25. L. Afanasiev, I. Manolescu, and P. Michiels. MemBeR: A Micro-Benchmark Repository for XQuery. In *XSym'05: Proc. of 3rd Int. XML Database Symposium*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
26. L. Afanasiev and M. Marx. An Analysis of the Current XQuery Benchmarks. In *ExpDB'06: Proc. of the 1st Int. Workshop on Performance and Evaluation of Data Management Systems*, pages 9–20, Chicago, Illinois, USA, 2006. ACM.
27. P. Azalov and F. Zlatarova. SDG – A System for Synthetic Data Generation. In *ITCC'03: Proc of the Int. Conf. on Information Technology: Computers and Communications*, pages 69–75, Washington, DC, USA, 2003. IEEE Computer Society.
28. D. Barbosa, A. Mendelzon, and J. Keenleyside. *ToXgene*. IBM, 2005. http://www.alphaworks.ibm.com/tech/toxgene.
29. D. Barbosa, A. O. Mendelzon, J. Keenleyside, and K. A. Lyons. ToXgene: A Template-Based Data Generator for XML. In *SIGMOD'02: Proc. of the 2002 ACM SIGMOD Int. Conf. on Management of Data*, page 616, Madison, Wisconsin, USA, 2002. ACM.
30. D. Barbosa, A. O. Mendelzon, J. Keenleyside, and K. A. Lyons. ToXgene: An Extensible Template-Based Data Generator for XML. In *WebDB'02: Proc. of the 5th Int. Workshop on the Web and Databases*, pages 49–54, Madison, Wisconsin, USA, 2002.
31. D. Barbosa, L. Mignet, and P. Veltri. Studying the XML Web: Gathering Statistics from an XML Sample. *World Wide Web*, 8(4):413–438, 2005.
32. A. Berglund, S. Boag, D. Chamberlin, M. F. Fernndez, M. Kay, J. Robie, and J. Simeon. *XML Path Language (XPath) 2.0*. W3C, January 2007. http://www.w3.org/TR/xpath20/.
33. G. J. Bex, F. Neven, and J. Van den Bussche. DTDs versus XML Schema: a Practical Study. In *WebDB'04: Proc. of the 7th Int. Workshop on the Web and Databases*, pages 79–84, New York, NY, USA, 2004. ACM Press.
34. P. V. Biron and A. Malhotra. *XML Schema Part 2: Datatypes (Second Edition)*. W3C, October 2004. http://www.w3.org/TR/xmlschema-2/.
35. S. Boag, D. Chamberlin, M. F. Fernndez, D. Florescu, J. Robie, and J. Simeon. *XQuery 1.0: An XML Query Language*. W3C, January 2007. http://www.w3.org/TR/xquery/.
36. T. Bohme and E. Rahm. Benchmarking XML Database Systems – First Experiences. In *HPTS'01: Proc. of 9th Int. Workshop on High Performance Transaction Systems*, Pacific Grove, California, 2001.
37. T. Bohme and E. Rahm. 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, 2001. Springer-Verlag.

38. T. Bohme and E. Rahm. *XMach-1: A Benchmark for XML Data Management*. Database Group Leipzig, 2002. http://dbs.uni-leipzig.de/en/projekte/XML/XmlBenchmarking.html.
39. T. Bohme and E. Rahm. Multi-User Evaluation of XML Data Management Systems with XMach-1. In *Proc. 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, 2003. Springer-Verlag.
40. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. W3C, September 2006. http://www.w3.org/TR/REC-xml/.
41. S. Bressan, M.-L. Lee, Y. G. Li, Z. Lacroix, and U. Nambiar. The XOO7 XML Management System Benchmark. Technical Report TR21/00, National University of Singapore, November 2001.
42. S. Bressan, M.-L. Lee, Y. G. Li, Z. Lacroix, and U. Nambiar. The XOO7 Benchmark. In *Proc. 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, 2003. Springer-Verlag.
43. S. Bressan, M. L. Lee, Y. G. Li, B. Wadhwa, Z. Lacroix, U. Nambiar, and G. Dobbie. *The XOO7 Benchmark*. 2002. http://www.comp.nus.edu.sg/~ebh/XOO7.html.
44. R. Busse, M. Carey, D. Florescu, M. Kersten, I. Manolescu, A. Schmidt, and F. Waas. *XMark – An XML Benchmark Project*. Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 2003. http://www.xml-benchmark.org/.
45. M. J. Carey, D. J. DeWitt, and J. F. Naughton. The OO7 Benchmark. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 22(2):12–21, 1993.
46. D. Chamberlin, P. Fankhauser, D. Florescu, M. Marchiori, and J. Robie. *XML Query Use Cases*. W3C, 2007. http://www.w3.org/TR/xquery-use-cases/.
47. S. A. Chilingaryan. *XML Benchmark*. 2004. http://xmlbench.sourceforge.net/.
48. J. Clark. *XSL Transformations (XSLT) Version 1.0*. W3C, November 1999. http://www.w3.org/TR/xslt.
49. J. Clark and S. DeRose. *XML Path Language (XPath) Version 1.0*. W3C, November 1999. http://www.w3.org/TR/xpath.
50. C. Cooper. *Benchmarking XML Parsers*. XML.com, 1999. http://www.xml.com/pub/a/Benchmark/article.html.
51. S. DeRose, R. Daniel, P. Grosso, E. Maler, J. Marsh, and N. Walsh. *XML Pointer Language (XPointer)*. W3C, August 2002. http://www.w3.org/TR/xptr/.
52. S. DeRose, E. Maler, and D. Orchard. *XML Linking Language (XLink) Version 1.0*. W3C, June 2001. http://www.w3.org/TR/xlink/.
53. J. Dokulil and J. Katreniakova. Visual Exploration of RDF Data. In *SOFSEM'08: Proc. of the 34th Int. Conf. on Current Trends in Theory and Practice of Computer Science (to appear)*, Lecture Notes in Computer Science. Springer-Verlag, 2008.
54. M. Farwick and M. Hafner. *XML Parser Benchmarks: Part 1*. XML.com, 2007. http://www.xml.com/pub/a/2007/05/09/xml-parser-benchmarks-part-1.html.
55. M. Farwick and M. Hafner. *XML Parser Benchmarks: Part 2*. XML.com, 2007. http://www.xml.com/pub/a/2007/05/16/xml-parser-benchmarks-part-2.html.
56. M. Franceschet. *XPathMark*. University of Udine, Italy, 2005. http://users.dimi.uniud.it/~massimo.franceschet/xpathmark/.

57. M. Franceschet. XPathMark – An XPath Benchmark for XMark Generated Data. In *XSym'05: Proc. of 3rd Int. XML Database Symposium*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
58. M. Franceschet, E. Zimuel, L. Afanasiev, and M. Marx. *XCheck*. Informatics Institute, University of Amsterdam, The Netherlands, 2006. `http://ilps.science.uva.nl/Resources/XCheck/`.
59. H. J. Jeong and S. H. Lee. A Versatile XML Data Generator. *International Journal of Software Effectiveness and Efficiency*, 1(1):21–24, 2006.
60. P. Kumar. *XPB4J – XML Processing Benchmark for Java*. 2002. `http://www.pankaj-k.net/xpb4j/`.
61. E. Kuznetsov and C. Dolph. *XSLT Benchmark Results*. XML.com, 2001. `http://www.xml.com/pub/a/2001/03/28/xsltmark/results.html`.
62. E. Kuznetsov and C. Dolph. *XSLT Processor Benchmarks*. XML.com, 2001. `http://www.xml.com/pub/a/2001/03/28/xsltmark/index.html`.
63. Y. G. Li, S. Bressan, G. Dobbie, Z. Lacroix, M. L. Lee, U. Nambiar, and B. Wadhwa. XOO7: Applying OO7 Benchmark to XML Query Processing Tool. In *CIKM'01: Proc. of the 10th Int. Conf. on Information and Knowledge Management*, pages 167–174, New York, NY, USA, 2001. ACM.
64. H. Lu, J. X. Yu, G. Wang, S. Zheng, H. Jiang, G. Yu, and A. Zhou. What Makes the Differences: Benchmarking XML Database Implementations. *ACM Trans. Inter. Tech.*, 5(1):154–194, 2005.
65. S. Manegold. An Empirical Evaluation of XQuery Processors. *Inf. Syst.*, 33(2):203–220, 2008.
66. I. Manolescu, C. Miachon, and P. Michiels. Towards Micro-Benchmarking XQuery. In *ExpDB'06: Proc. of the 1st Int. Workshop on Performance and Evaluation of Data Management Systems*, pages 28–39. ACM, 2006.
67. S. Marcus. *Benchmarking XML Parsers on Solaris*. XML.com, 1999. `http://www.xml.com/pub/a/1999/06/benchmark/solaris.html`.
68. S. I. Martinez, P. Grosso, and N. Walsh. *Extensible Markup Language (XML) Conformance Test Suites*. W3C. `http://www.w3.org/XML/Test/`.
69. I. Mlynkova, K. Toman, and J. Pokorny. Statistical Analysis of Real XML Data Collections. In *COMAD'06: Proc. of the 13th Int. Conf. on Management of Data*, pages 20–31, New Delhi, India, 2006. Tata McGraw-Hill Publishing Company Limited.
70. U. Nambiar, Z. Lacroix, S. Bressan, M. Lee, and Y. Li. Benchmarking XML Management Systems: The XOO7 Way. Technical Report TR-01-005, Dept of Computer Science, Arizona State University, November 2001.
71. U. Nambiar, Z. Lacroix, S. Bressan, M. Lee, and Y. Li. XML Benchmarks Put To The Test. In *IIWAS'01: Proc. of the 3rd Int. Conf. on Information Integration and Web-Based Applications and Services*, Linz, Austria, 2001.
72. U. Nambiar, Z. Lacroix, S. Bressan, M.-L. Lee, and Y. G. Li. Efficient XML Data Management: An Analysis. In *EC-WEB'02: Proc. of the 3rd Int. Conf. on E-Commerce and Web Technologies*, pages 87–98, London, UK, 2002. Springer-Verlag.
73. M. Nicola, I. Kogan, R. Raghu, A. Gonzalez, M. Liu, B. Schiefer, and G. Xie. *Transaction Processing over XML (TPoX)*. `http://tpox.sourceforge.net/`.
74. M. Nicola, I. Kogan, and B. Schiefer. An XML Transaction Processing Benchmark. In *SIGMOD'07: Proc. of the 2007 ACM SIGMOD Int. Conf. on Management of Data*, pages 937–948, New York, NY, USA, 2007. ACM.
75. Y. Oren. *SAX Parser Benchmarks*. SourceForge.net, 2002. `http://piccolo.sourceforge.net/bench.html`.

76. B. V. Phan and E. Pardede. 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, 2008. IEEE Computer Society.
77. M. Rorke, K. Muthiah, R. Chennoju, Y. Lu, A. Behm, C. Montanez, G. Sharma, and F. Englich. *XML Query Test Suite*. W3C, 2007. `http://www.w3.org/XML/Query/test-suite/`.
78. K. Runapongsa, J. M. Patel, H. V. Jagadish, Y. Chen, and S. Al-Khalifa. The Michigan benchmark: towards XML Query Performance Diagnostics (Extended Version). `http://www.eecs.umich.edu/db/mbench/mbench.pdf`.
79. K. Runapongsa, J. M. Patel, H. V. Jagadish, Y. Chen, and S. Al-Khalifa. *The Michigan Benchmark*. Department of Electrical Engineering and Computer Science, The University of Michigan, 2006. `http://www.eecs.umich.edu/db/mbench/`.
80. K. Runapongsa, J. M. Patel, H. V. Jagadish, Y. Chen, and S. Al-Khalifa. The Michigan benchmark: towards XML Query Performance Diagnostics. *Inf. Syst.*, 31(2):73–97, 2006.
81. A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, M. J. Carey, I. Manolescu, and R. Busse. Why and How to Benchmark XML Databases. *ACM SIGMOD Record*, 30(3):27–32, 2001.
82. A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, and R. Busse. The XML Benchmark Project. Technical Report INS-R0103, CWI, Amsterdam, The Netherlands, April 2001.
83. D. M. Sosnoski. *XMLBench Document Model Benchmark*. 2002. `http://www.sosnoski.com/opensrc/xmlbench/index.html`.
84. H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. *XML Schema Part 1: Structures (Second Edition)*. W3C, October 2004. `http://www.w3.org/TR/xmlschema-1/`.
85. O. Vosta, I. Mlynkova, and J. Pokorny. Even an Ant Can Create an XSD. In *DASFAA'08: Proc. of the 13th Int. Conf. on Database Systems for Advance Applications (to appear)*, Lecture Notes in Computer Science. Springer-Verlag, 2008.
86. B. B. Yao and M. T. Ozsu. *XBench – A Family of Benchmarks for XML DBMSs*. University of Waterloo, School of Computer Science, Database Research Group, 2003. `http://se.uwaterloo.ca/~ddbms/projects/xbench/`.
87. B. B. Yao, M. T. Ozsu, and J. Keenleyside. XBench – A Family of Benchmarks for XML DBMSs. In *Proc. 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, 2003. Springer-Verlag.
88. B. B. Yao, M. T. Ozsu, and N. Khandelwal. XBench Benchmark and Performance Testing of XML DBMSs. In *ICDE'04: Proc. of the 20th Int. Conf. on Data Engineering*, pages 621–632, Washington, DC, USA, 2004. IEEE Computer Society.