Querying XML : benchmarks and recursion
Afanasiev, L.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Conclusions

In this chapter, we revisit the research questions that we pursed in this thesis and summarize our answers. We then wrap-up with a discussion of future directions.

10.1 Answering the research questions

In this thesis, we pursued two main research themes: How to evaluate the performance of XML query processing? and How to optimize recursion in XQuery? These general questions boiled down to more concrete questions that we addressed in the different chapters of the thesis. In this section, we summarize our answers to each of these questions.

Developing benchmarking methodology

While pursuing the first general research question, we focused our investigation on the XQuery language and started with an analysis of existing XQuery benchmarks: XMach-1, XMark, X007, MBench, and XBench. The main conclusion of this analysis is that the benchmarks are very useful for exploratory performance studies, but not adequate for rigorous performance evaluations of XML query processors. The three detailed questions that we addressed, Question 3.1, 3.2, and 3.3, and their answers follow.

Question 3.1 What do the benchmarks measure?

We analyzed and compared the workloads (i.e., the data and query sets) and measures of the five benchmarks. The benchmarks differ in their target and performance measure. Our comparison showed that XMach-1 and MBench have a distinct and clear focus, while X007, XMark, and XBench, have a more diffuse focus and are similar in many respects. XMach-1 is an application benchmark that tests the overall performance of an XML DBMS in a real application scenario; the benchmark measure is the query throughput. MBench a micro-benchmark that
Chapter 10. Conclusions

tests the performance of an XML query processor on five language features on an artificial document, where the benchmark measure is the query processing time. X007, XMark, and XBench are application benchmarks that test the performance of an XML query processor on a relatively small set of complex queries. The latter benchmarks differ from each other in the document scenario they test: X007, XMark, and XBench TC/SD and DC/SD test a single-document scenario, while XBench TC/MD and DC/MD test a multi-document scenario.

All benchmark queries have a common characteristic: they are designed to test important language features. However, we observed that a single query usually contains more than one language feature and an engine’s performance on that query cannot be attributed to only one of them. From this, we concluded that the benchmark queries have an exploratory nature rather than precise nature.

When considered together, as a family, the benchmarks have a good coverage of the main characteristics of XML documents and of the important XQuery language features. Nevertheless, they do not cover the whole space of XML query processing scenarios and parameters. Several advanced XML/XQuery features, such as typed data, namespaces, recursion, etc., are poorly covered. Also, 90% of all benchmark queries can already be expressed in XPath 1.0 or 2.0, provided that we consider only the element retrieval functionality and ignore the XML construction functionality of XQuery (the other 10% of the queries test two XQuery constructs: sorting and recursive user-defined functions).

Question 3.2: How are the benchmarks used?

We conducted a survey of scientific articles that contain experimental studies of XML processing and were reported in the 2004 and 2005 proceedings of the ICDE, SIGMOD and VLDB conferences, 41 papers in total. The survey showed that fewer than 1/3 of the articles on XML query processing that provide experimental results use benchmarks (11 papers use XMark and 2 papers use XBench). The remaining articles use ad-hoc experiments to evaluate their research results. The majority of these (73%) use benchmark data sets or real data and ad-hoc query sets. Thus, we concluded that with the exception of XMark and XBench, the benchmarks are not used. A reason for the limited usage of the benchmarks might be that many of the papers contain an in-depth analysis of a particular XPath/XQuery processing technique and the benchmarks are not suitable for this kind of analysis.

Question 3.3: What can we learn from using the benchmarks?

To answer Question 3.3, we ran the benchmarks on four XQuery engines: Galax, SaxonB, Qizx/Open, and MonetDB/XQuery and compared their performance. We observed that: (i) the engines produce errors and suffer from crashes (even on workloads of small size); (ii) the relative performance of the engines varies on different benchmarks; and (iii) the engines’ performance differs per benchmark
10.1. Answering the research questions

...even for queries intended to test the same language feature. We concluded that the tested engines (or at least the versions we tested) are still immature and that no engine can be crowned as a winner. The results further indicate that implicit benchmark parameters have a big impact on the performance. The benchmarks, even applied together, cannot be used to obtain a comprehensive understanding of the performance of an engine.

Since MBench was designed as a micro-benchmark, we tested whether MBench is suitable for a rigorous analysis of a language feature it targets, namely attribute-value joins. Based on the benchmarks results obtained on Qizx/Open, we concluded that the set of four queries designed for micro-benchmarking joins is insufficient for drawing sound conclusions about its performance. Thus, even though MBench provides a good starting point for micro-benchmarking, it is incomplete, which leads to inconclusive results.

Question 4.1: How to ensure the repeatability of experimental studies of database systems? This question incorporates two sub-questions: (i) What is a proper methodology for designing and reporting on experimental studies that facilitates their repeatability? and (ii) What is a proper mechanism for evaluating and improving the repeatability of experimental studies presented in scientific research?

At the SIGMOD 2008 conference, by way of an experiment, submitted papers were subjected to a repeatability reviewing process. Although repeatability reviewing was optional, most authors participated. In order to facilitate the task of the reviewers, the authors of the papers were requested to follow a specific methodology in describing and reporting on their experiments. All in all, this mechanism provides a solution for ensuring repeatability of experimental studies of database systems, and it shows that repeatability may be achieved and measured. Based on our experience as a member of the repeatability reviewing committee, we were able to address the above questions.

Concerning Question 4.1(i), the methodology used for describing and reporting experiments that the repeatability committee developed was enough to cover the 289 papers that were submitted for the repeatability review. Out of 64 papers that were assessed by the repeatability committee, 33 (52%) achieved the repeatability of all presented experiments and 20 (31%) achieved repeatability of some of the experiments. Considering that we strive for all experiments to be repeatable, 52% is a small number. Nevertheless, we consider these results to be a good start towards achieving the repeatability of experimental studies in the database research field.

Concerning Question 4.1(ii), the high percentage of participation in the optional review, 66% of the total submissions to the conference (289 out of 436), hints at the perceived usefulness of a peer reviewing process. The positive feedback from the authors of the papers recorded by the survey also confirms that...
Chapter 10. Conclusions

such a review is considered useful for the community: 80% of the surveyed authors found the process useful, while 84% would participate in such a process in the future.

There are still some problems that need to be addressed, though, if ensuring repeatability is to become a common practice in the database community. One of the problems is the amount of effort that the reviewing process takes.

Question 5.1: Is it possible to build a generic tool for automating the following three tasks: (i) running a performance benchmark, (ii) documenting the benchmark experiment, and (iii) analyzing the benchmark results? What are the design choices needed to be made?

We presented XCheck, a tool for running performance benchmarks that measure execution times on sets of XML documents and sets of queries, formulated in an XML query language, such as XPath and XQuery. Given a benchmark and a set of engines, XCheck runs the benchmark on these engines, collects performance times, query results, and the testing environment configuration. XCheck was used for running all experiments in this thesis, and, as of September 2009, it has been used in at least 9 scientific papers.

During the development of XCheck we had to address several issues. First, we had to decide how XCheck should communicate with the tested engines. The command line adapter design that XCheck implements is elegant and easily implementable—many of the XML query engines have a command line interface. Second, we had to decide what atomic metric XCheck should implement. Based on what the current XQuery benchmarks measure, namely performance times of a set of queries on a set of documents/collections, the atomic metric deployed by XCheck is the total execution time of processing one query on a document/collection. If the engines provide more detailed performance times, e.g., document processing, query compilation, etc., XCheck also collects these times. Finally, we had to decide how to store and present the performance results. XCheck uses XML to store the raw measurement data, and it uses HTML and plots to present it to the user in an easily readable format.

Question 6.1: What is a suitable methodology for precise and comprehensive performance evaluations of XML query processing techniques and systems?

As a result of investigating Questions 3.1–3.3, we identified a lack of suitable tools for precise and comprehensive performance evaluations. As a solution to this problem, we proposed MemBeR, an open-ended, community driven, repository of micro-benchmarks. We endowed the repository with micro-benchmarking design principles and methodology, with a fixed micro-benchmark structure, with suggestions for potentially interesting parameters, and tools for generating parametrized data sets. In Chapter 7, we presented a concrete micro-benchmark for evaluating value-based joins processing techniques that follows the MemBeR methodology.
10.1. Answering the research questions

With MemBeR we aim to consolidate the experience of individual researchers that spend time and effort in designing micro-benchmarks for performance evaluation of their query optimization and processing techniques. We hope MemBeR will provide the necessary performance evaluation tools and methodology and will be used in the XML data management community.

Question 7.1: How to measure the performance of value-based joins expressed in XQuery? What is a suitable measure and which parameters are important to consider?

We designed a micro-benchmark for value-based joins in XQuery, following the MemBeR methodology. The benchmark measures the impact of seven query and data parameters on the performance times of an engine. The benchmark query set is carefully designed to allow for testing the impact of every parameter value in isolation. We validated our benchmark by analyzing the performance of four XQuery engines. We obtained a comprehensive overview of the performance of each engine when it comes to evaluating joins, and we identified some shortcomings of the engines, as well as some missed opportunities for optimization. We concluded that the benchmark achieves its target and it is a useful tool for profiling the performance of XQuery engines on value-based joins with respect to the tested parameters.

Recursion in XQuery

Next, we answer the research questions pursued in the second part of the thesis, referring to recursion in XQuery.

Question 8.1: What is a suitable declarative recursive operator in XQuery that is rich enough to cover interesting cases of recursion query needs and that allows for (algebraic) automatic optimizations?

As a solution to this question we proposed an inflationary fixed point (IFP) operator for XQuery. This operator covers important use cases of recursion in XQuery, such as the transitive closure of path expressions, while also being susceptible to systematic optimizations. We also propose an optimization technique for this operator. This optimization relies on a distributivity property of XQuery expressions that can be effectively detected at the syntactic level. Furthermore, if we adopt a relational approach to XQuery evaluation, then distributivity can be detected more conveniently and effectively at the underlying algebraic level. We integrated the IFP operator into the MonetDB/XQuery system and assessed the practical gain of our approach on real-life use cases using the benchmarking tools developed in Part I.
Question 9.1: How feasible is it to do static analysis for recursive queries specified by means of the fixed point operator? Specifically, are there substantial fragments of XQuery with the fixed point operator for which static analysis tasks such as satisfiability are decidable?

We investigated the theoretical aspects of the IFP operator in the context of Core XPath (CXP) [Gottlob and Koch, 2002], the navigational core of XPath. We proved that the satisfiability problem of CXP extended with the IFP operator is undecidable. In fact, the fragment containing only the self and descendant axes is already undecidable. This means that a complete static analysis of recursive queries specified by means of the inflationary fixed point operator is not feasible. In other words, we cannot do better than implementing sound-but-not-complete query optimizations, such as the distributivity-based optimization presented in Chapter 8. As a by-product of the undecidability result, we also obtained that CXP extended with IFP is strictly more expressive than CXP extended with the transitive closure (TC) operator, also known as Regular XPath [Marx, 2004].

10.2 Outlook and further directions

While answering the above research questions, we have raised new questions and identified unsolved problems. In this section, we list these questions and problems for each research theme separately.

Performance evaluation of XQuery engines

We have analyzed existing benchmarks for performance evaluation of XQuery benchmarks and arrived at the conclusion that the XQuery community will benefit from new benchmarks—both application benchmarks and micro-benchmarks—that have a good coverage of XQuery features. Indeed, at the time of writing this thesis, another XQuery application benchmark has been proposed, TPox [Nicola et al., 2007], while we developed and presented micro-benchmarks and related methodology. A new question arises: do these new benchmarks and benchmarking methodology fulfill the need for benchmarking XQuery processing? In either case, our analysis showed that a serious investment should be made for maintaining the benchmarks at the same pace as the development of the XQuery engines (and language) themselves, otherwise the benchmarks quickly become obsolete.

Addressing the need for precise and comprehensive benchmarks, we developed a micro-benchmarking repository and related methodology, MemBeR. We have also developed a MemBeR micro-benchmark targeting the processing of value-based joins expressed in XQuery. Nevertheless, there is still a long way before the repository will contain micro-benchmarks covering many language features of XQuery and other XML query languages. We hope that more contributions will
be made to MemBeR and that the MemBeR micro-benchmarks will be used by the community.

With respect to the value-based join micro-benchmark that we developed, we have two questions left. One is whether the two benchmark parameters that did not show impact on the performance of the tested engines have impact on other engines and what impact. Another question is whether there are other parameters that might influence the impact of performance of this language features and, if so, can the micro-benchmark be extended to include them.

Benchmarking tools are only one aspect of achieving good performance evaluation. We have addressed the problem of ensuring the repeatability of experimental studies in the database community. Though, we made a first step and showed that repeatability may be indeed achieved and measured, there are still problems that need to be addressed, if ensuring repeatability is to become a common practice. One of the problems is the amount of effort that the reviewing process takes. Tools that automate the process of conducting and reporting on experiments, such as XCheck (Chapter 5), might be useful to reduce this effort.

Another aspect of repeatability that we did not discuss in this thesis is that of proper archiving mechanisms for ensuring the accessibility of experimental data and results. Long-term preservation and even curation of experimental results is another key factor of scientific proliferation. This question is actively being addressed in other database related fields, such as Information Retrieval [Agosti et al., 2007].

The realization of the need of serious empirical evaluation is gaining ground in the database community, as is witnessed by the experimental studies repeatability reviewing efforts at SIGMOD 2008 and 2009, by the establishment of “The Experiments and Analyses” track at VLDB 2008 and 2009, the organization of workshops like ExpDB 2006 and 2007, etc.. We hope that this thesis makes a contribution to this development. Nevertheless, as indicated by the above list of open questions, the biggest hurdles lie ahead.

Recursion in XQuery

We have proposed optimization techniques for recursion in XQuery by introducing a declarative recursive operator to XQuery. In spite of the fact that the inflationary fixed point operator covers a large class of recursive query needs in XQuery, some natural recursive operations cannot be expressed with it or it is cumbersome, e.g., recursive XML construction (XML transformations) and recursive aggregates. It remains an open question what set of declarative recursive operators would be most natural to implement in the XQuery settings. This set should: (i) cover the most useful, commonly used, recursive query needs, and (ii) be easily implementable and susceptible to automatic optimizations.

On a more theoretical level, we made a connection between Core XPath extended with the IFP operator (CXP+IFP) and Modal Logic extended with
the IFP operator (ML+IFP). We exploited this connection and established that CXP+IFP is highly undecidable. Several natural follow-up questions have arisen.

One follow-up question is whether CXP+IFP node expressions are strictly more expressive than ML+IFP formulas.

Another natural follow-up question is whether the undecidability result can be strengthened to smaller fragments of CXP+IFP. Recall that in CXP+IFP, the variables can be used both as atomic path expressions and as atomic node expressions. The former is the most natural, but the translation we gave from ML+IFP to CXP+IFP crucially uses the latter. Our conjecture is that the fragment of CXP+IFP in which variables are only allowed as atomic path expressions is already undecidable.

It is also natural to consider CXP+IFP expressions where the fixed point variables occur only under an even number of negations, so that the \texttt{WITH}-operator computes the least fixed point of a monotone operation. Note that this fragment is decidable, since it is contained in monadic second-order logic. Thus, further questions like the complexity of the static analysis problems and the expressive power of this language are open to investigation.

To wrap up, we believe that optimizing recursion in XQuery by exploring declarative recursive operators is worthwhile investigating, it might lead to further significant performance improvements and interesting theoretical questions.