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.

UvA-DARE is a service provided by the library of the University of Amsterdam (http://dare.uva.nl)
Chapter 1

Introduction

The Extensible Markup Language (XML) [World Wide Web Consortium 2008] is a textual format for representing data. Since its introduction in 1998 as a World Wide Web Consortium (W3C) recommendation, it has become a widely accepted standard for storage and exchange of data. XML was specifically designed to be able to handle semi-structured data: data that is not sufficiently structured, or whose structure is not sufficiently stable, for convenient representation in a traditional relational database.

The increasing amount of data available in the XML format raises a great demand to store, process, and query XML data in an effective and efficient manner. To address this demand, the W3C has developed various accompanying languages that facilitate, e.g., typing, querying, and transformation of XML data.

A large part of this thesis is devoted to XQuery, the XML Query language [World Wide Web Consortium 2007]. XQuery has been developed since 2000 and has been published as a W3C recommendation in 2007. Some of the main ingredients of the XQuery language are XML document navigation, data typing, filtering, joining, ordering, new XML document construction, and recursive user-defined functions. The fragment of XQuery that is concerned with XML document navigation is known as XPath. XPath is an important language that was developed independently of XQuery by the W3C and proposed as a recommendation in 1999 [World Wide Web Consortium 1999a]. It lies at the heart of XQuery as well as of other XML-related technologies such as Extensible Stylesheet Language Transformations (XSLT) and XML Schema.

In this thesis, we are concerned with the query processing problem for XQuery: given an XML document or collection of documents, and a query, to compute the answer to the query. This is a difficult problem. A long line of research addresses the problem of efficient query processing. One strategy is to employ query optimization. Query optimization refers to deciding which query processing

\[1\text{This problem is also known as query evaluation. In this thesis, though, we reserve the term “evaluation” for the empirical evaluation of systems or techniques.}\]
technique to use at run time, in order to gain efficiency.

Our research falls within two topics: (i) developing optimization techniques for XQuery, and (ii) performance evaluation for XQuery processing. These topics are tightly connected, since performance evaluation is aimed at measuring the success of optimization techniques. More specifically, we pursue optimization techniques for recursion in XQuery, on the one hand, and benchmarking as a performance evaluation technique, on the other hand.

Recursion in XQuery

The backbone of the XML data model, namely ordered, unranked trees, is inherently recursive and it is natural to equip the associated query languages with constructs that can query such recursive structures. XQuery provides two mechanisms for expressing recursive queries: recursive axes, such as descendant and ancestor [World Wide Web Consortium, 1999a] and recursive user-defined functions. The recursive axes of XPath provide a very restricted form of recursion, while recursive user-defined functions in XQuery allow for arbitrary types of recursion. Indeed, they are the key ingredient of the Turing completeness of the language [Kepser, 2004]. Thus, in terms of expressive power, the designers of the language took a giant leap from recursive axes to recursive user-defined functions.

Recursive user-defined functions are procedural in nature, they describe how to get the answer rather than what the answer is. Language constructs that describe the query answer are called declarative [Brundage, 2004]. Procedural constructs put the burden of optimization on the user’s shoulders. Recursive user-defined functions largely evade automatic optimization approaches beyond improvements like tail-recursion elimination or unfolding [Park et al., 2002, Grinev and Lizorkin, 2004].

In this thesis, we investigate the possibility of adding a declarative recursion operator to the XQuery language. While less expressive than recursive user-defined functions, the declarative operation puts the query processor in control of query optimizations.

Benchmarking

Performance is a key criterion in the design and use of XML processing systems. Performance evaluation helps to determine how well a system performs (possibly in comparison with alternative systems), whether any improvements need to be made and, if so, to which (bottleneck) components of the system, what the optimal values of the system’s parameters are, and how well the system will perform in the future, etc. This makes performance evaluation a key tool to achieving good performance.

Benchmarking is a performance measurement technique that has a well-established and acknowledged role in the development of DBMSs [Jain, 1991]. A
1.1 Research questions

Our research has two main goals: one is to develop benchmarking methodology and tools for performance evaluation of XML query engines, the other is to analyze and develop optimization techniques for declarative recursion operators for XML query languages. Correspondingly, the research questions that we address fall into two groups.

Developing benchmarking methodology and tools

The general research question of the first part of the thesis is: *How to evaluate the performance of XML query processing?* To answer this question, we first investigate the tools previously proposed in the literature, namely five benchmarks for performance evaluation of XQuery engines. The following questions are driving our investigation:

- **Question 3.1**: What do the benchmarks measure? or What conclusions can we draw about the performance of an engine from its benchmark results?
- **Question 3.2**: How are existing benchmarks used?
- **Question 3.3**: What can we learn from using the five benchmarks?

As a result of our investigation, we learn that there is a need for performance tools and methodology for precise and comprehensive experimental studies and that there are no tools to meet this need. Hence, the question we address next is:

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

Our answer to this question involves micro-benchmarks and a methodology for micro-benchmarking. To illustrate, and put to work, the methodology we propose, we undertake a concrete performance evaluation task driven by the following question:
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?

Besides building benchmarking tools and methodology, there are two other main issues that need to be addressed when pursuing performance evaluation. The following questions address these issues:

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

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 that need to be made?

**Inflationary fixed points in XQuery**

Having focused on developing benchmarking methodology in the first part of the thesis, we pursue the following general question in the second part of the thesis: *How to obtain declarative means of recursion in XQuery?* This boils down to a more concrete question:

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 an answer to this question we consider an inflationary fixed point (IFP) operator added to XQuery and investigate an efficient evaluation technique for it. The next question addresses the theoretical properties of this operator in the setting of XQuery.

Question 9.1: How feasible it is 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?

The connection between our two main concerns in this thesis—performance evaluation and optimization—is established in Chapter 8 where we use the benchmarking tools developed in Part I to evaluate the performance of the optimizations proposed in Part II.
1.2 Main contributions

Following the research questions listed above, our contributions fall within the two topics of the thesis. The main contributions of Part I relate to developing tools for performance evaluation of XML query processors. The main contributions of Part II relate to analyzing and developing optimization techniques for a declarative recursion operator in XML query languages. A detailed list of contributions follows.

Developing benchmarking methodology and tools

1. A survey of experimental evaluations of XML query engines published by the database scientific community; a survey of XQuery benchmarks; an analysis of XQuery benchmarks with the focus on the benchmark queries and measures; a corrected and standardized list of benchmark queries.

2. A micro-benchmarking methodology for systematic, precise, and comprehensive performance analysis of XML query engines; a framework for collecting and storing micro-benchmarks. Both the methodology and the framework collectively go under the name of MemBeR.

3. A micro-benchmark for testing value-based joins expressed in XQuery; an analysis of four XQuery engines on value-based joins based on this micro-benchmark.

4. An attempt at defining a standard for reporting experimental studies with the goal of assuring their repeatability; a report on the review of repeatability of experimental studies conducted for the research articles submitted to SIGMOD 2008.

5. A tool for automatic execution of benchmarks on several XML query engines, called XCheck. XCheck includes statistical and visualization tools that help the performance analysis.

Inflationary fixed points in XQuery

1. An analysis of an inflationary fixed point operator for expressing recursion in XQuery; a distributivity property of IFP expressions that allows for efficient query processing; implementation of the IFP operator into an XQuery engine (MonetDB/XQuery); a performance evaluation of our approach.

2. An analysis of theoretical properties of the logical core of the XPath language (Core XPath) enhanced with an inflationary fixed point operator; a proof of the undecidability of this language; a proof of the fact that this language is strictly more expressive than the language obtained by adding a Transitive Closure operator to Core XPath.
1.3 Organization of the thesis

This thesis is organized in two parts, each focused on a different general research question. Before these parts begin, in Chapter 2 we set the background and introduce the terminology used throughout the thesis. After these two parts end, we conclude the thesis in Chapter 10. Figure 1.1 suggests reading paths through the thesis.

Part I (Benchmarks) is composed of five chapters addressing the research questions dealing with performance evaluation of XML query processing. In Chapter 3 we analyze existing XQuery benchmarks and survey their usage. This chapter serves as a detail discussion of related work for the whole part. In Chapter 4 we discuss how to achieve the repeatability of experimental evaluations of computer systems in the database domain. As part of setting up a methodology for repeatability, we perform a review of articles submitted to the research conference SIGMOD 2008 and measure the repeatability of the presented experimental evaluations. In Chapter 5 we discuss the problems and challenges of automating the execution of performance evaluation benchmarks on many XML query engines and comparison of their performance. We present a software tool, XCheck, as a
solution to these problems. In Chapter 6 we present a methodology for micro-benchmarking XML query engines, which we refer to as MemBeR. MemBeR also comprises a framework for collecting and storing micro-benchmarks. Finally, in Chapter 7 we present a MemBeR-style micro-benchmark for testing performance of value-based joins expressed in XQuery. We validate the micro-benchmark by analyzing the performance of four XQuery engines.

Part II (Recursion) is composed of two chapters addressing the research questions referring to recursion in XQuery. In Chapter 8 we consider adding an inflationary fixed point operator to XQuery. We develop an optimization technique for processing this operator. Further, we implement this technique on top of MonetDB/XQuery, and evaluate its performance. In Chapter 9 we study the theoretical properties, decidability and expressivity, of this inflationary fixed point operator in the setting of Core XPath, the XML tree navigational core of XPath and XQuery.

The work presented in Chapter 8 relies on a logical fragment of XQuery called LiXQuery. For a quick reference to the language constructs, we present the definition of the syntax of LiXQuery in Appendix A.

1.4 Publications underlying the thesis

The material in this thesis builds on a number of previously published papers that we list below. Full details of these publications can be found in the Bibliography.

Part I builds on work presented in:
1. “An analysis of XQuery benchmarks” [Afanasiev and Marx, 2006] and [Afanasiev and Marx, 2008];
2. “The repeatability experiment of SIGMOD 2008” [Manolescu et al., 2008a];
3. “XCheck: a Platform for Benchmarking XQuery Engines” [Afanasiev et al., 2006]; and

Part II builds on work presented in:
1. “On Core XPath with Inflationary Fixed Points” [Afanasiev and ten Cate, 2009]; a journal version of this article is currently in preparation;
2. “Recursion in XQuery: Put Your Distributivity Safety Belt On” [Afanasiev et al., 2009];
3. “An Inflationary Fixed Point Operator in XQuery” [Afanasiev et al., 2008];
4. “Lekker bomen” [Afanasiev et al., 2007]; and
5. “PDL for Ordered Trees” [Afanasiev et al., 2005b].