*Executive Summary of Dagstuhl Seminar*

# XQuery Implementation Paradigms (06472)

Nov 19 – 22, 2006


| | | |
|---|---|---|
| *Organizers:* | Peter A. Boncz | (CWI Amsterdam, NL) |
| | Torsten Grust | (TU München, DE) |
| | Jérôme Siméon | (IBM TJ Watson Research Center, USA) |
| | Maurice van Keulen | (U Twente, NL) |


## Motivation and Seminar Topic

Only a couple of weeks after the participants of seminar No. 06472 met in Dagstuhl, the W3C published the Final Recommendation documents that fix the XQuery 1.0 syntax, data model, formal semantics, built-in function library and the interaction with the XML Schema Recommendations (see W3C's XQuery web site at `http://www.w3.org/XML/Query/`). With the language's standardization nearing its end and now finally in place, the many efforts to construct correct, complete, and efficient implementations of XQuery finally got rid of the hindering "moving target" syndrome. This Dagstuhl seminar on the different XQuery implementation paradigms that have emerged in the recent past, thus was as timely as it could have possibly been.

From the beginning, XQuery has been designed as a declarative language in the style of modern functional programming languages. For the query author, declarativity means that the formulation of queries solely depends on the desired input and output—efficiency concerns should not have any impact at all. For XQuery implementations, declarativity provides a sheer endless pool of alternative strategies to consume and represent data model instances as well as to compile, optimize, and execute queries. In principle, all of these strategies are acceptable as long as they respect the language's formal semantics.

This freedom has led to a plethora of, sometimes radically different, approaches to the implementation of XQuery. It is characteristic for most of the implementation projects in this "zoo", that a specific set of XQuery features drove their initial development, *e.g.*, the evaluation of XPath location steps or the efficient implementation of nested *FLWOR* expressions and the derivation of equivalent database-style join strategies. To this end, our colleagues out in the field applied existing techniques and devised new approaches rooted in the programming language and database query language domains. Still, XQuery implementations which excel in both, completeness *and* efficiency, are rare (if available at all) today.


Dagstuhl Seminar Proceedings 06472
XQuery Implementation Paradigms
http://drops.dagstuhl.de/opus/volltexte/2007/1022

It was the foremost goal of this seminar to bring together a vivid group of academic and industrial researchers who are representatives of the distinct implementation camps that can be currently found in the XQuery landscape. In particular, the organizers tried to make sure that the *native*, *relational*, and *streaming* implementation camps all had their fair share of participants. We are happy to report that a total of 31 colleagues found their way to Dagstuhl—in effect, for three days the castle saw a concentration of expertise in the XQuery language and its implementation that goes unmatched even when compared to the major global scientific conferences in the field.

## Organization of the Seminar and Activities

***Talks.*** The seminar featured an exciting program of presentations covering the latest research and industrial advances in the area of XQuery implementation.

The first set of presentations were focused on the foundations of XQuery compilation. They covered a wide range of techniques, starting with algebraic compilation [Jens Teubner (TU München, DE), *A Purely Relational Approach to XQuery*; Jérôme Siméon (IBM Watson, USA), *An Algebraic Compiler for An Expressive XQuery Extension*] which described XQuery engines developed using traditional database approaches. Those were followed by the presentation of complementary techniques targeted towards tree-pattern detection in algebraic plans [Philippe Michiels (U Antwerp, BE), *Put a Tree Pattern in Your Algebra*]. Moving further away from pure database techniques, the following talks discussed the impact of choosing push *vs.* pull models in XQuery compilers and run-times [Michael Kay, (Saxonica Ltd., UK) *Push or Pull (Does it Make a Difference)?*], and the use of functional programming optimizations for XQuery [Kristoffer Rose, (IBM Watson, USA), *Functional Optimizations of XQuery using Higher Order Rewriting*]. The set of presentations underlined the variety and the complementary of the various techniques, with an increasing number of system combining those techniques.

The second set of presentations focused on recent developments in the area of XQuery benchmarking. There was a wide consensus among the seminar participants on the importance of developing benchmark suites that cover different classes of usage of the language and are representative of the processing performed in real applications. First, an overview of the most important efforts in that area were presented [Loredana Afanasiev (U Amsterdam, NL), *An Analysis of the Current XQuery Benchmarks*], including *XMach-1*, *XMark*, *X007*, the *Michigan benchmark*, and *XBench*. Special attention was given to the *MemBer* micro-benchmark which aims at evaluating the individual performance of XQuery implementations for fundamental operations, notably path navigation, *FLWOR* expressions, and element construction. Detailed overviews of two more recent efforts were also presented. The first one being *XPathMark* [Massimo Franceschet (U Udine, IT),

*XPathMark: functional and performance tests for XPath*], which focuses on evaluating the conformance, completeness, and performance for XPath support in existing implementations. The second, and maybe most ambitious, benchmark presented was *TPoX* [Matthias Nicola, (IBM San Jose, USA) *An Application-oriented XML Transaction Processing Benchmark*] which had it first public announcement at this seminar. *TPoX* is an application-level benchmark based on the FIXML dialect used in the financial industry. It includes query and update workloads as well as a complete transaction manager which can simulate access by simultaneous users over a large number of documents.

Finally, a third set of presentations explored more experimental development of the language, including support for XML updates [Ying Zhang (CWI Amsterdam, NL), *Loop-lifted XQuery RPC with Deterministic Updates*], support for text search [Paul Pedersen (FLWOR Foundation, USA), *Searching for XQuery*], support for distributed queries [Jérôme Siméon, *Distributed XQuery*], transaction management [Bettina Kemme (Mc Gill U, CA), *Snapshot-based Concurrency Control for XML*], and the possible use of XQuery over file systems [Marc H. Scholl (U Konstanz, DE), *File Systems are Obsolete (!?)*]. Those presentations are representative of some of the most advanced research in the area, and are indicative of the on-going interest in using XQuery for more advanced application development. Significant challenges are still to be tackled in order to develop mature solutions in each of those areas.

***Breakout Sessions.*** The above presentations often triggered lively technical discussions between seminar participants. Among the various aspects being discussed, a few specific topics that could benefit from further discussion were singled out as breakout sessions. In each case, those breakout sessions resulted in a clearer description of the most important problems, and some suggestions for possible solutions. We report briefly on the outcome of those breakout sessions.

*XQuery Benchmarking.* The first breakout session focused on the area of XQuery benchmarking. The *XMark* benchmark has been very popular for evaluating the effectiveness of new compilation techniques. The main reasons for this are its simplicity in generating data and the queries included in the benchmark. However, the process of running the benchmark is not well-defined. Thus the interpretation of benchmark results is difficult. Additionally, participants believed the set of queries included in *XMark* does not reflect the state of the XQuery specification and queries formulated by users. As a result, it was recommended that future benchmarks preserve the strengths of *XMark* while overcoming its shortcomings. To this end the need to structure the process of defining both application or micro-benchmarks should be enphasized, and the following bottom-up approach has been proposed: (1) define relevant data sets, (2) define statement sets (*i.e.*, queries and

updates), and (3) define the workload as a combination of statements. Finally, it has been suggested to organize this process by using the *MemBer* repository to manage datasets, queries, workloads, and discussions. Finally, in order to encourage a wide adoption of the resulting benchmark, the participants proposed to assign the "Dagstuhl label" to the new benchmark and, hence, call it *DMark*.

*Transactional Support for XQuery.* The second breakout sessions focused on the area of transactional support for XQuery. This session discussed the challenges XML poses to efficient transaction management. It considered both the programming interface and the internals of transaction management, and explored the following questions. What kind of transaction model is appropriate for XML applications? Where should transaction boundaries be set? What isolation levels are appropriate for XML applications? How can the ACID properties be implemented in an XML engine? Do the techniques used to guarantee transactional properties differ from those used in relational systems? How can distributed transactions be handled? The area of transactions management for XQuery is still largely unexplored and the participants felt further research in that important area should be encouraged.

*XQuery Applications.* The third breakout session focused on the area of XQuery and its applications. This breakout disussion focused on the functionality of XQuery from two perspectives. (1) Application view: The functionality is very rich (maybe too rich), making it difficult to understand all subtleties. On the other hand, there may even be the need to add some selected additional functionality (such as some imperative language features). (2) Implementation view: The functionality is very rich and some parts of the semantics make an (efficient!) implementation difficult. Participant believed that in many cases, standards in the area of query languages resulted in powerful languages, but that in many cases typical application make use of only a small fraction of the overall functionality. As a result, they encouraged the identification of a small and simple subset of XQuery as a language kernel and package additional functionality as predefined sets of optional features. This could be proposed as an item on an XQuery 2.0 wish list. The discussion could not, obviously, identify this small "kernel" *vs.* the "extension packages." Two sample features played a role in the discussion: recursion and (imperative) variables with assignments (see XQueryP).

*XQuery Compiler Interoperability.* The fourth breakout session focused on the area of interoperability between XQuery compilers. This discussion resulted from the observation that several existing implementations shared common principles, and often had complementary strengths. An important observation resulting from that discussion is that greater interoperability between systems should result in the ability to share development more easily, to facilitate a meaningful comparison between systems, and to enable a greater reuse of research results between those

systems. Several research groups represented among the participants agreed to explore the possibilty to develop APIs and intermediate languages to improve interoperability between their implementations.

***XQuery Hard Nuts.*** Last, the seminar was concluded by a lively session on so-called XQuery *hard nuts*. The purpose of that session was to solicitates from the participant examples of queries over XML data which they believed are difficult to express, or to evaluate efficiently. In each case, an effort was made to identify research problems that could be inferred from those examples. We briefly give here only an overview of those examples. The corresponding XQueries in each case can be found on the Web page of the seminar.

Massimo Franceschet presented an example of a recursive functions in XQuery which implements a transitive closure of a location path over the XMark document. This example emphasized the importance and difficulty in evaluating highly recursive XQueries.

Torsten Grust (TU München, DE) presented an example that illustrated the need to use functional programming optimization techniques such as memoization. The example illustrated a common idiom where the same function is called many different times over the same parameters.

Jérôme Siméon presented three different examples. The first example illustrated the need to support join optimization combined with a recursive function. The second example showed a simple recursive transform for which streaming has been shown to provide considerable speedup compared to currently proposed XQuery compilation techniques. The third example emphasized the need for increasing the robustness of optimization in existing XQuery compilers. This last example featured a set of XQueries which are semantically equivalent but syntactically quite different and challenged the participant to build compilers which can equally optimize those different queries.

Finally, Maurice van Keulen (U Twente, NL) presented two examples. The first example again involved recursion. He argued that recursion usually structurally "follows" the XML hierarchy (by descending into an XML tree), the order in sequences, some diminishing computation, or a combination thereof. He also argued that such cases are currently very hard to support using some of the existing compilation approaches, notably based on algebras. The second example illustrated a common pattern in his experience where some part of the code is used to build intermediate data structure, while another part of the code is used to access those structure. He observed that the intermediary structures are not really necessary, and challenged the participant to develop techniques which could detect when such computation patterns could "cancel each other out".

### Concluding Remarks and Future Plans

The functional nature of the XQuery language makes it particularly amenable to implementation techniques developed in the functional programming languages domain (this point was made by Kristoffer Rose, Philippe Michiels, Jérôme Siméon, Maurice van Keulen, and Torsten Grust). It is indeed perceivable to define faithful reformulations of the XQuery semantics in terms of combinator languages or variants of monad comprehensions, two expressions forms from which efficient database-style algebraic plans can be derived. A group of seminar participants will engage in an effort to further develop and study a (unified) algebraic representations for XQuery (see fourth breakout session). Ideally, this will lead to interoperability between some of the many promising XQuery implementation efforts.

We hoped that the participants were prepared and willing to teach each other in a constructive fashion and we were lucky to find exactly that during the seminar days. Dagstuhl greatly helped to create an atmosphere in which the formerly separate camps collaboratively worked on the syntheses of proven XQuery compilation and evaluation techniques.

The organizers would like to sincerely thank the Dagstuhl Scientific Directorate of Dagstuhl castle and are looking forward to put forward a follow-up seminar proposal which will reflect the then current developments around the XQuery language. Quite possibly this will include *XQuery 1.1*, whose initial requirements analysis phase has started just as we write this, and the forthcoming *XQuery Scripting Extension* which will bring the worlds of functional XML querying and stateful programming even closer together.