XML Query Processing: Storage and Query Model Interplay

Ioana Manolescu
INRIA-Futurs and LRI, Gemo group
4, rue Jacques Monod, 91893 Orsay Cedex, France
E-mail: Ioana.Manolescu@inria.fr
Web: http://www-rocq.inria.fr/ manolesc

May 31, 2004

Tutorial type Advanced.

Presentation language English.

Tutorial goal XML data management has received a lot of attention from various perspectives. In this tutorial, we attempt to systematize the existing techniques, from the *persistent database* perspective, in which documents are stored once in a persistent repository, and queried/updated many times in the sequel. We survey existing techniques for storing XML documents, in tight connection with the *execution frameworks* that such techniques enable; our purpose is to highlight the tight connection between storage and querying, and their mutual influences.

Special audio-visual needs n/a

Other venues of the tutorial The tutorial will also be presented at the EDBT summer school to take place in Sardinia, Italy, in September 2004 (see http://edbtss04.dia.uniroma3.it/).

Author's bio Ioana Manolescu has earned her Ph.D. in 2001, at University of Versailles and INRIA (France), on "Optimization techniques for querying distributed heterogeneous data sources". She has been a post-doc in Politecnico di Milano, Italy, working in the integration of Web services and workflow in data-intensive Web applications. Since 2002, Ioana is a member of the Gemo group in INRIA Futurs, France. Her current research topics include XML storage and indexing, XQuery processing (she is co-chairing XIME-P, a new XQuery-centered workshop in cooperation with the ACM SIGMOD conference this year), peer-to-peer data management, and distributed process modeling for Web service integration.

Outline of the tutorial

XML data management has received a lot of attention from various perspectives; the problem has been studied for different query languages, such as Lorel [1], StruQL [13], XML-QL [11] and XQuery [31], and for various settings, including: the *persistent database scenario*, where documents are stored on disk and queried subsequently, using algebra-based techniques, e.g. as in [10, 14, 22, 26]; the *streaming scenario*, where documents are queried on the fly as they arrive, e.g. as in [18, 24]; and the *compiled scenario*, where documents are read in memory, while queries are evaluated by compiling them to some imperative or functional language code [3, 25]. Notice that some execution techniques devised for streaming data can also be applied on a persistent store.

This tutorial addresses the evaluation of XML queries expressed in XQuery, in the persistent database scenario, with a focus on the interplay between storage and query execution strategies. Each storage is best suited for some execution strategies; in turn, execution strategies that are established as clear winners push their matching storage schemes into the spotlight. Understanding this storage-processing interplay is important, as no XML query can be deemed "complex" or "easy" without referring to the storage and query model chosen.

The tutorial covers the fundamental elements of XML storage and execution models proposed in the last six years, and systematize them so as to encompass current and foreseeable future efforts.

1. Preliminaries: XML and XQuery

We briefly survey the particular aspects of XML that make the storage problem challenging, such as: heterogeneity, alternatives and repetition, need for identity, and support for order. We then introduce the most important features of the XQuery language, and the XQuery processing model [31]. We only introduce XQuery insofar as our topic requires, since it is not our main focus.

2. Requirements for XQuery processing

Building on the XQuery elements presented, we introduce here the requirements for efficient XQuery processing based on a persistent storage. The main elements covered here are:

- Efficient data access primitives.
- Efficient structural operations.
- Algebraic support for dealing with: order, duplicates, and nesting (nesting is frequent in XQuery).

Each of the three above can have a great impact on query performance. Together, they cover most (if not all) of the performance problems linked to XQuery evaluation.[1] The first is directly determined by the storage; the second also crucially depends on the storage, although in a more subtle manner. Finally, while a good algebra is crucial, at the basis of any query execution plan lie data access operators; they have to agree with the upper-level algebra.

3. Storage and query processing: an evolutionary tale

A variety of XML storage schemes have been proposed so far. We divide them in several broad classes, and show how each of them copes with the XQuery processing requirements we identified.

Relational storages A first large class of systems seeked to take advantage of existing storages, most notably, relational ones [4, 6, 12, 16, 27, 28, 29]. For such systems:

Data access is performed through the DBMS's modules (table scan or index access).

Structural operations are translated, at best, to structural joins [32], and at worst, to large join chains [16, 27]. Poor support for structural operations is the best-known drawback of relational-based storages.

The algebra used in such systems is unordered, bag-oriented, and flat (no complex objects). While simplifying data access, this hinders the construction of efficient plans for complex XQuery queries.

[1] *Full-text* XML querying is likely to pose supplementary challenges. We do not address them here, as the field is still quite immature, and focus instead on the challenges posed by XQuery processing, where problems – and solutions !– are better understood.

Tree-based storages A second class of systems is oriented towards tree-based data storage; the most notable example here is Natix [22], but also the basic tree storage in [20].

Data access is performed through tree iterators.

Structural operations are translated in navigation. This entails high access locality requirements on the storage.

The algebra used here is also tree-based [21], and sometimes inspired by nested relational models [15].

Recent work [20, 9] has shown that navigation-based structural operations are outperformed by direct (index-based) data access. This result is consistent with similar observations made when optimizing object-oriented queries [19].

Emerging models Performance problems encountered for XQuery processing on relational or tree-oriented storage models have lead to the development of new types of storages [2, 5, 8, 23, 30]. These new models are quite heterogeneous, but they share the following features:

Data access in such systems is based on very low-level persistent storage modules (e.g. at the level of a persistent hash table rather than the level of a tuple or object manager).

Structural operations rely on *node labeling schemes* or other special index structure for efficiency. This is in recognition of the crucial performance impact of such operations.

The algebras used by such systems vary in their nature, but concur in the imperative for efficient XQuery processing - the XQuery requirement is injected in the very design.

4. Final considerations

After the euphoria of the early XML development years, complex query languages like XQuery and increasingly-complex XML applications show the limitations of simple storages and simplistic algebras. The field of XML storage and query processing is not yet exhausted, but the big challenges have been laid out: good storage support, and expressive algebras catering for the needs of algebraic reformulation, unnesting, duplicate, grouping, and order properties of a query execution plan.

As the state of the art advances, the convergence of views and systems requires good understanding and critical assessment of each technique. No particular conference paper will ever present "an XQuery processor", since such systems need to address a wide range of aspects, and cannot fit into the narrow conference format. But each new technique (whether storage or indexing model, physical or logical algebra) should be assessed in the light of its potential to contribute to a complete, viable framework. The purpose of the tutorial is to foster such critical reflection, by systematizing some questions to ask, and their existing and foreseeable answers.

Related issues and resources

For those interested in a more detailed introduction to XQuery, an excellent tutorial can be found at [7]. At a more advanced level, interesting issues in streaming XML query processing are described in [17].

An active field of research not covered by the tutorial is the one of XML updates. Suitability to updates, however, is a requirement for each storage format. Other interesting issues stem from the usage of schemas and schema constraints for query formulation, simplification, and optimization. Such issues can be seen as orthogonal to the storage and query model chosen, as they often apply at the query pre-processing level. To the extent, however, to which schema constraints can be injected in the storage design, the tutorial may briefly touch such issues.

References

[1] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.

[2] A. Arion, A. Bonifati, G. Costa, S. D'Aguanno, I. Manolescu, and A. Pugliese. Efficient query evaluation over compressed XML data. In *EDBT*, 2004.

[3] V. Benzaken, G. Castagna, and A. Frisch. CDuce: an XML-centric general-purpose language. In *ICFP*, pages 51–63, 2003.

[4] P. Bohannon, J. Freire, P. Roy, and J. Simeon. From XML schema to relations: A cost-based approach to XML storage. In *ICDE*, 2002.

[5] Jan-Marco Bremer and Michael Gertz. On distributing XML repositories. In *WebDB*, pages 73–78, 2003.

[6] M. Carey, D. Florescu, Z. Ives, Y. Lu, J. Shanmugasundaram, E. Shekita, and S. Subramanian. XPERANTO: Publishing object-relational data as XML. In *International Workshop on the Web and Databases*, 2000.

[7] Donald D. Chamberlin. XQuery: A query language for XML. In *SIGMOD*, page 682, 2003. Slides available at www.almaden.ibm.com/cs/people/chamberlin.

[8] Yi Chen, Susan Davidson, and Yifeng Zheng. BLAS: An efficient XPath processing system. In *SIGMOD*, 2004.

[9] Z. Chen, H.V. Jagadish, L. Lakshmanan, and S. Paparizos. From tree patterns to generalized tree patterns: On efficient evaluation of XQuery. In *VLDB*, 2003.

[10] Sophie Cluet, Pierangelo Veltri, and Dan Vodislav. Views in a large scale XML repository. In *VLDB*, pages 271–280, 2001.

[11] Alin Deutsch, Mary F. Fernandez, Daniela Florescu, Alon Y. Levy, and Dan Suciu. A query language for XML. In *Proc. of the Int. WWW Conf.*, volume 31(11-16), pages 1155–1169, 1999.

[12] M. Fernandez, W. Tan, and D. Suciu. SilkRoute: Trading between relational and XML. In *Proc. of the Int. WWW Conf.*, May 2000.

[13] Mary Fernandez, Daniela Florescu, Jaewoo Kang, Alon Levy, and Dan Suciu. Catching the boat with Strudel: Experiences with a web-site management system. In *SIGMOD*, Seattle, WA, 1998.

[14] Thorsten Fiebig, Sven Helmer, Carl-Christian Kanne, Guido Moerkotte, Julia Neumann, Robert Schiele, and Till Westmann. Anatomy of a native XML base management system. *VLDB Journal*, 11(4), 2002.

[15] Irna M. R. Evangelista Filha, Alberto H. F. Laender, and Altigran Soares da Silva. Querying semistructured data by example: The QSByE interface. In *Workshop on Information Integration on the Web*, 2001.

[16] D. Florescu and D. Kossmann. Storing and querying XML data using an RDMBS. In *IEEE Data Engineering Bulletin*, 1999.

[17] D. Florescu and D. Kossmann. XML query processing. In *ICDE*, 2004. Slides available at http://www-dbs.informatik.uni-heidelberg.de/publications/50.ppt.

[18] Daniela Florescu, Chris Hillery, Donald Kossmann, Paul Lucas, Fabio Riccardi, Till Westmann, Michael J. Carey, Arvind Sundararajan, and Geetika Agrawal. The BEA/XQRL streaming XQuery processor. In *VLDB*, pages 997–1008, 2003.

[19] Daniela D. Florescu. *Search Spaces for Object-Oriented Query Optimization.* PhD thesis, Univerisity of Paris VI, France, 1996.

[20] A. Halverson, J. Burger, L. Galanis, A. Kini, R. Krishnamurthy, A.N. Rao, F. Tian, S. Viglas, Y. Wang, J.F. Naughton, and D.J. DeWitt. Mixed mode XML query processing. In *VLDB*, 2003.

[21] H.V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava, and Keith Thompson. TAX: A tree algebra for XML. In *DBPL*, pages 149–164, 2001.

[22] C.C. Kanne and G. Moerkotte. Efficient storage of XML data. In *ICDE*, 2000.

[23] Christoph Koch. Efficient processing of expressive node-selecting queries on XML data in secondary storage: A tree automata-based approach. In *VLDB*, pages 249–260, 2003.

[24] B. Ludascher, P. Mukhopadhyay, and Y. Papakonstantinou. A transducer-based XML query processor. In *VLDB*, 2002.

[25] A. Marian and J. Simeon. Projecting XML documents. In *VLDB*, 2003.

[26] S. Paparizos, S. Al-Khalifa, A. Chapman, H.V. Jagadish, L. Lakshmanan, A. Nierman, J. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. TIMBER: A native system for querying XML. In *SIGMOD*, page 672, 2003.

[27] J. Shanmugasundaram, H. Gang, K. Tufte, C. Zhang, D. DeWitt, and J. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *VLDB*, 1999.

[28] J. Shanmugasundaram, J. Kiernan, E. Shekita, C. Fan, and J. Funderburk. Querying XML views of relational data. In *VLDB*, 2001.

[29] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, and B. Reinwald. Efficiently publishing relational data as XML documents. In *VLDB*, Cairo, Egypt, 2000.

[30] A. Vyas, M. Fernandez, and J. Simeon. The simplest XML storage manager ever. In *XIME-P workshop*, 2004. Informal proceedings.

[31] The XQuery language. www.w3.org/TR/xquery, 2003.

[32] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On supporting containment queries in RDBMSs. In *SIGMOD*, 2001.