Processing XML in Database Systems
Schmidt, A.

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.
Database Management Systems are frameworks that traditionally have been used to facilitate the design, construction and manipulation of large sets of primarily tabular data. With the advent of XML, a new kind of data which is less structured and follows a less strict methodology has become ubiquitous. The goal of this thesis is to motivate the need for incorporating the new paradigm into the database methodology by extending it to accommodate data with a less rigid structure.

We set out by giving an overview of XML and related technologies and standards like schema definition and query languages. By just looking at the W3C standard recommendation, XML is simply a syntactical way of describing how to combine text and markup into structured documents. The many standardisation efforts, however, that have emerged around XML and build on it, make use of the markup in a way that closely resembles semantic modelling processes such as entity-relationship modelling. Thus, XML content models, i.e., the tag hierarchies used or permissible in a document, in many applications resemble real world entities, which, in turn, makes them an interesting target for query languages. Constraint definition languages like DTDs or XML Schema help to narrow down to range of permissible document structures and to get closer to the systematic rigidity of data models as deployed in relational database management systems. These models in turn equip a user with more control over the semantic integrity of the data used in an application, which then may even be enforced automatically or semi-automatically.

A most important reason why it is interesting to look at where and how XML and database technology can benefit from each other is not only that there are parallels between data-oriented XML documents and the table-based relational data model of DBMSs, which suggests that there are a number of areas where both worlds can benefit from each other, but also a software engineering aspect. One very attractive and widespread application of XML technologies is to serve as a kind of glue between systems to improve communication and reduce engineering effort since many standard data interchange tasks can be delegated to available software that complies with XML standards.

The high degree of freedom, however, that document designers enjoy when they conceive document structures also brings about the downside that only very general semantics can be assigned to syntactic elements. This becomes apparent when a semantically rich data model like the relational data model is to used to model and store XML data. The characterisation of physical data breakdowns that has been presented made clear that there are numerous possibilities each with its own advantages and con-
Conclusion

The meet operator, which was also presented in Chapter 4, is a tool to let users without domain knowledge query XML databases; this is done by exploiting the semantics that are encoded in the hierarchically nested tag structures that are the back-bone of XML's syntax. The key idea of the operator is to find the smallest enclosing context of certain document and query dependent groups of nodes; this concept is reduced to a lowest common ancestor search in the document tree. The experiments demonstrated that the meet can both return intuitive results and be implemented in an efficient manner.
and is therefore a useful addition to XML query engines.

An extension to relational engines concluded Chapter 4. By adding the CHOOSE operator to a relational query processor, we let users extend the query optimiser with rules that are not expressible in state-of-the-art commercially deployed constraint declaration languages. Although the operator is sensible not only in XML settings but in many application areas like Geographical Information Systems or On-Line Analytical Processing, XML can especially benefit from it. Each one in the plethora of ways to physically break down data comes with specific advantages, disadvantages and naturally constraints that may be hard to declare to a query processor. The CHOOSE operator can help here to enlarge the optimiser’s search space with promising alternatives. The focal feature of the operator is that it seamlessly integrates into the existing optimiser infrastructure.

Chapter 5 focused on the performance of XML processors and on the big picture. We started out by looking at what determines the performance of XML databases in general: the infrastructure around that database may be as important when an application is deployed as the database itself. A recurring point throughout this thesis was the observation that one of the primary reasons for introducing XML was to channel programmers’ workflow by making data exchange both machine and programming language independent and thus more robust. Due to its wide use, XML also fosters standardisation by encouraging programmers to make the data structures used in communication public. The task of an XML database in now to integrate smoothly into this all-XML framework; on the one hand, this is a software engineering challenge, on the other hand, since the database management system usually is the component to cope with large amounts of data, performance is often mission critical. That is where the XMark benchmark can help. It provides a document data base and a query set; together they model typical query challenges and help users analyse the performance characteristics of their XML processor. As well as challenging the query processor on which the benchmark is run with the XMark queries, users can at the same time do a requirements check on the set-up and the embedding infrastructure.

Concerning future work, there are still plenty of research opportunities since XML support in Database Management Systems is still in its infancy. XML data models so far have been concerned with storage and query processing issues; due to the lack of a standard data model, the impact of physical data breakdown, which may bring about intricate dependencies, on transaction processing, which is especially important in Internet settings, has so far received only little attention. Equally important and another natural research direction is to look at the exploration of opportunities of parallel processing and efficient handling of multi-query workloads as found in typical interactive Web-based information systems, a domain where XML is particularly strong. We have shown how to extend the relational algebra for XML processing; unfortunately, introducing joins and unions may enlarge the optimiser’s search space so much that the built-in heuristics do not work well anymore. It may be worthwhile to look at the interaction between algebraic primitives and the search space to find ways to enable even complex ad hoc queries, which may be problematic as our analysis of the benchmark results has shown. For even better support of novice users it may be fruitful to add document-dependent heuristics to the meet operator, possibly combining ontological information with hierarchical semantics and probability modes to improve ranking and result presentation. In terms of the CHOOSE operator, it would be natural to extend the concept of non-deterministic choice not only to cost estimation but also to availability of resources, which is an important problem in network information systems; note however that this would require support from a query execution engine since these
decisions would have to be made while the query is being executed. Then, however it would relieve the user of the burden of specifying which resource should be used during query execution and the query engine could choose from a set of different resources. As we have seen in some of our experiments, there is also the need for a general, standardised methodology that allows conclusive performance analyses and facilitates comparisons of different approaches.