Beyond XML Query Languages

Peter Buneman Alin Deutsch Wenfei Fan
Hartmut Liefke Arnaud Sahuguet Wang-Chiew Tan

University of Pennsylvania
{peter,adeutsch,wfan,liefke,sahuguet,wctan}@saul.cis.upenn.edu

November 18, 1998

A query language is essential, if XML is to serve effectively as an exchange medium for large data sets. The design of query languages for XML is in its infancy, and the choice of a standard may be governed more by user acceptance than by any understanding of underlying principles. One would hope that expressive power, performance, and compatibility with other languages will be considered in choosing among alternatives, but it is likely that several contenders will co-exist for some time.

It is worth observing that, during the 20-year development of relational query languages, several competing languages were developed; and even today there are several relational query language standards. In spite of this, a great deal of technology was developed that was *independent* of the surface syntax of a query language. This included technology "below" the language such as efficient execution models and work "above" the level of language – such as techniques for view definition and maintenance, triggers, etc. At Penn we are working on some of these language-independent issues. We include a summary of them here. They include execution and data models to support XML and semistructured query languages; the use of schemas and constraints in optimizing XML query languages; and tools for extracting data form existing sources and presenting it as XML.

1 Challenges for Query Languages

Due to the success of the XML standard, we foresee the availability of large amounts of XML on the Web in the near future. This fact poses questions that the XML standard does not address. In particular,

- How will data be extracted from large XML documents?
- How will XML data be exchanged, e.g., by shipping XML documents or by shipping queries?
- How will XML data be exchanged between user communities using different but related ontologies (or DTD's)?
- How will XML data from multiple sources be integrated?

Data extraction, transformation, and integration are all well-understood database problems. Their solution relies on a query language, either relational (SQL) or object-oriented (OQL). These query languages do not apply immediately to XML, because the XML data differs from traditional relational or object-oriented data. XML data, however, is very similar to a data model recently studied in the research community: the semistructured data model. Several research query languages have been designed and implemented for semistructured data [6, 10, 2]. XML-QL [9] applies the experience accumulated in the design and implementation of these languages to query XML documents using pattern matching.

There are differences between the rather general model used in semi-structured data, and the XML model. While this may not affect the surface syntax of the query language, it will certainly affect the semantics of query languages and the optimizations that are possible. One of the research areas at Penn is the investigation of new models for semistructured data that may be more appropriate to XML. In particular, the principle that paths uniquely identify (nested) elements is not supported by current models for semi-structured data, and an alternative model is needed.

2 Constraints and Optimization

Among the various proposals for structuring or adding semantics to XML, a number advocate the need for constraints [5, 12, 11, 1] A class of integrity constraints, called *path constraints*, has been introduced and studied in [7]. These constraints are capable of expressing natural integrity constraints that are not only a fundamental part of the semantics of the data, but are also important in query optimization. In particular, they are useful for specifying and querying XML documents.

For example, let us assume that we have a set of projects (`Proj`) and a set of departments (`Dept`). Each department is involved in one or more projects (element `DProj`); each project is managed by one unique department (element `PDept`). Eventhough the cyclicity of element nesting requires the use of IDREF attributes, `DProj` and `PDept` will be treated as nested element tags in the following examples. The following are path constraints, which describe a typical inclusion constraint and an inverse relationship between projects `DProj` and departments `PDept`.

(INCL):
```
<constraint>
     <inclusion path = "Dept.DProj"  memberOf ="Proj" />
</constraint>
```

(INV):
```
<constraint>
     <prefix     path = "Dept" />
     <inverse    path = "DProj"  inverseOf = "PDept"/>
</constraint>
```

These constraints state that:
For each <Dept> element, and each of its <DProj> element dp, dp is also a <Proj> element.
For each <Dept> element d and each of its project elements <DProj> p, p is a <Proj> element such that its <PDept> element is d.

To see how these constraints help in query optimization, consider the XML-QL [9] queries in figure 1 (in XML-QL, IDREF attributes are dereferenced implicitely and treated as nested element tags).

Query Q, returning the names of projects with budget over 10000 and their corresponding departments, performs two iterations over the database, but its evaluation can be optimized to execute just a single iteration over the database: query Q' (equivalent to Q due to INCL) iterates over `Dept` entries, while query Q'' (equivalent to Q' due to INV) iterates over `Proj` entries.

3 Updates and annotations to XML documents

Updating documents is currently done in a very different fashion to database updates. Structural editors guarantee some integrity of the document structure, but they do not ensure any kind of consistency with schemas or type descriptions [13]. Just as we need languages for querying XML documents, we need languages for updating them.

Typically, information sources on the Internet are shared with many other users. Often, users keep local copies of documents to make querying more efficient. In this case, changes to the original information source

```
where    <Dept>
           <DProj>$dp</>
           <DName>$dn</>
         </Dept> ← DB,
         <Proj>$dp</> ← DB,
         <Budget>$b</> ← $dp,
         <PName>$pn</> ← $dp,
         $b > 10000

collect  <PName>$pn</>
         <DName>$dn</>
```

(a) Original query Q

```
where    <Dept>
           <DName>$dn</>
           <DProj>
             <Budget>$b</>
             <PName>$pn</>
           </DProj>
         </Dept> ← DB,
         $b > 10000

collect  <PName>$pn</>
         <DName>$dn</>
```

(b) Q' (= Q with INCL)

```
where    <Proj>
           <Budget>$b</>
           <PName>$pn</>
           <PDept.DName>$dn</>
         </Proj> ← DB,
         $b > 10000

collect  <PName>$pn</>
         <DName>$dn</>
```

(c) Q" (= Q with INV)

Figure 1: Equivalent XML-QL Queries Under Inclusion and Inverse Constraints

needs to be propagated in an efficient way. Downloading the entire documents from the source – whether they were updated or not , is often quite inefficient. Large scientific information sources, such as genetic databases (e.g. Swissprot and GenBank [4, 8]), therefore provide information about what changes have been made to the database in a given time span. With availability of large data sets specified in XML, there will be a greater demand for a generic language for specifying updates for XML data.

Another important issue related to data evolution is to store information about the changes made in the information source. The time, date, the author, and the motivation of the update are often relevant information stored.

For example, in public genetic databases, such as Swissprot or GenBank [4, 8], DNA sequences or protein information are changed frequently, since more structural information becomes available over time. Furthermore, DNA sequences are annotated with other, additional information that is obtained by complex statistical data analysis.

XML already provides a syntax for annotating data. We are also interested in a solution for annotating changes to data. This is in the same spirit as "mend" annotation proposed for XML[15]. We seek a solution that is simple and sufficiently general to capture changes in semi-structured data as well as conventional databases so that one could answer questions such as: "*Where did this data item come from?*", "*How did it evolve to this state?*". This would require a system of annotations that is capable of describing data at different levels of granularity. One reason for maintaining such information is that corrections and errors may be traced back to the originator in an straight-forward fashion.

4 Data extraction: migration from HTML documents

Even if XML seems to be accepted with enthusiasm, we should not forget that the number of XML documents on the Web is still negligible. Even if data stored in information systems can be easily exported as XML documents (HTML and XML are just two output formats), a lot of valuable information will remain stored as HTML documents. The migration from HTML to XML is therefore a very serious issue.

A larger problem is to be able to extract structure from HTML documents and export it into other formats. This is extremely useful for data inter-operation (between Web sources and legacy databases or among Web sources themselves), automation of Web information processing and data migration. From a database perspective, it means to build wrappers for Web sources.

The WysiWyg Web Wrapper Factory developed at Penn is an attempt to offer a toolkit to build such Web wrappers. In our case, extracting information for HTML documents consists of: (1) building a parse tree (according to the Document Object Model like in [3, 16])) and (2) evaluating extraction rules. Extraction rules consist of path expressions [2] along the tree and are expressed using the HTML Extraction Language (HEL).

The language comes with some powerful operators (navigation operators, regular expressions, conditions) to reach information at the tag level and deeper, and permits to return some nested structures in order to truly capture the information expressed in the document. As an illustration, should we want to return the list of pairs (url , name) for hyperlinks pointing to a .edu domain in a Web document, we could write the following rule (see [14] for details):

```
links = html->a[i:*] ( .txt # .getAttr(href) )
where   html->a[i:*].getAttr(href) =~ "http://[^/]+.edu";
```

The extracted structure here is a list of pairs that can be mapped into any user-defined data-structure: relational database, XML, etc. The major difference between HEL and [3, 16] is that we can extract arbitrarily complex structures from the Web document, and not just strings.

To facilitate the writing of extraction rules we also offer a Wysiwyg interface that takes a Web document and add some invisible annotations: for each piece of text, the corresponding path is encoded in a new tag. The document is returned to the user who can click from his browser on any piece of information and get magically the corresponding extraction rule.
The HEL language and this visual approach can also be applied to XML documents.

We have been using successfully the toolkit to build a large range of wrappers for diverse Web resources like knowledge repositories, Web catalogues and databases. Most of the time, we came up with a robust up-and-running wrapper within minutes thanks to our wizards. The maintenance has also been quite easy. We have been using the toolkit together with XML-QL (by exporting XML structures) to have a powerful query language for Web sources.

5 Conclusions

There are a number of issues surrounding query languages for XML that are just as important as the query language itself. We have described some of them: the semantics of the underlying semistructured data model, the basic operations on data, the interaction of these operations with contraints, the nature of updates, and the problems of generating XML efficiently from existing sources.

We would like to thank our colleagues in the database group at Penn for their input.

References

[1] A. Layman and E. Jung and E. Maler and H. S. Thompson and J. Paoli and J. Tigue and N. H. Mikula and S. De Rose. XML-Data. Technical report, World Wide Web Consortium, 1998. `http://www.w3.org/TR/1998-NOTE-XML-data`.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L. Wiener. The Lorel Query Language for Semistructured Data. *Journal on Digital Libraries*, 1997.

[3] Charles Allen. WIDL: Application Integration with XML. *World Wide Web Journal*, 2(4), November 1997.

[4] Apweiler R. Bairoch A. The SWISS-PROT protein sequence data bank and its supplement TrEMBL. *Nucleic Acids Res.*, 26, 1998.

[5] T. Bray, C. Frankston, and A. Malhotra. Document Content Description for XML, 1998. `http://www.w3.org/TR/NOTE-dcd`.

[6] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *ACM-SIGMOD*, pages 505–516, Montreal, Canada, June 1996.

[7] P. Buneman, W. Fan, and S. Weinstein. Path constraints on semistructured and structured data. In *PODS*, 1998. `http://www.cis.upenn.edu/db/langs/98papers.html`.

[8] Benson D.A., Boguski M.S., Lipman D.J., Ostell J., and Ouellette B.F. Genbank. *Nucleic Acids Res.*, 26(1), 1998.

[9] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A query language for xml. `http://www.w3.org/TR/NOTE-xml-ql`.

[10] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for a web-site management system. In *SIGMOD record*, volume 26, pages 4–11, September 1997.

[11] M. Fuchs, M. Maloney, and A. Milowski. Schema for object-oriented XML. Technical report, World Wide Web Consortium, 1998. `http://www.w3.org/TR/NOTE-SOX`.

[12] O. Lassila and R. R. Swick. Resource Description ramework (RDF) model and syntax specification. Technical report, World Wide Web Consortium, 1998. `http://www.w3.org/TR/WD-rdf-syntax`.

[13] Bit Rot. Brian hayes. `http://www.sigmaxi.org/amsci/issues/Comsci98/compsci1998-09.html`.

[14] Arnaud Sahuguet and Fabien Azavant. W4F: the WysiWyg Web Wrapper Factory. Technical report, University of Pennsylvania, Department of Computer and Information Science, 1998. To appear. `http://cheops.cis.upenn.edu/∼sahuguet/WAPI`.

[15] Henry S. Thompson and David McKelvie. Hyperlink semantics for standoff markup of read-only documents. `http://www.ltg.ed.ac.uk/ht/sgmleu97.html`.

[16] World Wide Web Consortium. XML Pointer Language, 1998. `http://www.w3.org/TR/1998/WD-xptr-19980303`.