JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2004**

Vol. 3, No. 7, July-August 2004

On Supporting Structure-Agnostic Queries for XML

Won Kim, Cyber Database Solutions, Austin, Texas, USA
Wol Young Lee and **Hwan Seung Yong**, Department of Computer Science and Engineering, Ewha Women's University, Seoul, Korea

Abstract

XML has rapidly emerged as the standard for the interchange of data in numerous application areas. Many software vendors now offer XML servers to store and retrieve XML documents. To allow for content-based queries, query languages have also been designed. The query languages require the users to know the structure of the XML documents and specify search conditions on the structure. This navigational access to XML documents is a natural consequence of the hierarchical structure of XML. However, it is also desirable to allow the users to formulate structure-agnostic queries against XML documents, to complement the current navigation-based queries. We have developed a query processor to evaluate structure-agnostic XML queries. In this article, we motivate the need for structure-agnostic, that is, non-navigational, queries against XML, and offer it as a worthy subject of further research and development.

1 INTRODUCTION

XML [XML00] provides simple yet flexible ways to represent the structure and contents of arbitrary documents. An XML document is constructed based on a few simple rules. The core feature of an XML document is the "element". An element has a start tag (element name), data value, and an end tag. Further, an element can have attributes, can have children elements, and can be empty. The fact that an element can have children elements is of course what allows it to express hierarchical structures of arbitrary documents. The simplicity and flexibility of XML have made it possible for XML to be adopted as the basis of data interchange standards in a wide variety of application areas, including electronic business (ebXML, X12, Rosetta Net), financial services (FIXML, FPML), molecular biology (bioML, SBML, CellML, RNAML), chemistry (CML), multimedia (SMIL, SVG, MPEG-7), scientific research (RIXML), metadata management (XMI), web services (WSDL, UDDI), data mining (PMML), etc.

Many XML servers are currently on the market. Some store XML documents in XML-optimized file systems, while others use relational databases (by "shredding" XML documents into rows and columns in tables). XML database servers on the market include Oracle XML DB [Oracle]. DB2 XML Extender [IBM], SQL server 2000 [Hhowlett and Jennings], BEA Systems, Actuate/Nimble, Poet, BlueStone, eXcelon, IPEDO, XAware, etc. It is of course necessary for the users to be able to issue "content-based" queries against XML documents, that is, queries in which search conditions are specified as predicates (e.g., director = "Luc Besson", to retrieve movies directed by "Luc Besson" from a database of "movie" XML documents).

There have been many proposals for a query language for semi-structured data in general, and XML in particular. These include Lorel [Abiteboul, et al], XML-QL [Deutsch et al]. X-Path [Xpath], XSL [XSL], YATL, Quilt, and XQuery [XQuery]. These query languages require the users to know the structure of XML documents, including all the element and attribute names, data types of the data values, and the hierarchical structure of the elements. Since a search condition needs to involve an element/attribute name, which appears somewhere in the hierarchical structure of an XML document, these query languages force the users to specify the navigational access paths to the elements/attributes that are involved in search conditions. Insofar as XML gives rise to hierarchical structure, and navigational access to selected parts of an XML document is a natural access pattern, the navigational content-based query languages clearly make sense. However, we believe that structure-agnostic, that is, non-navigational, content-based query languages for XML, and general semi-structured data, are also desirable. In other words, we believe that both navigational and non-navigational query languages are necessary to support all query needs against XML databases.

2 RATIONALE

A brief review of the history of generational shifts in database systems can provide a helpful insight to understand the necessity of both types of query facility for XML. The transition from hierarchical and network database systems to relational database systems took place in the 1980s. The hierarchical and network models of data, in which a record contains physical pointers to other records, made sense for applications that modeled data that naturally forms hierarchical structure, such as the bill of materials. Applications had to access the records by navigation, going from one record to the next records by following the pointers embedded in the first record. However, the use of physical pointers was problematic, since the pointers had to be changed in all the records if the records to which they pointed changed their physical locations. (Further, if the data structures used to store the database changed, all the application logic that made explicit use of the knowledge of the data structures had to be changed.) The relational model of data explicitly excludes the use of pointers, both logical and physical, and instead has a record in one table to be "linked" dynamically to a record in another table if the two records have a common value in some of the user-specified fields. This dynamic "linking" of records in two or more tables is called join. The relational model of data offers a high degree of "data independence" and makes "ad hoc content-based" queries possible. "ad hoc content-based" queries mean queries in which the users only specify the search conditions on selected columns (and output columns), but not how the data is to be located and accessed (e.g., navigational paths) or how the query is to be processed (e.g., using specific access methods such as indexes or hashing or sorting). The onus of accessing the data and processing the queries falls entirely on the database system. To support ad hoc queries, relational database systems had to develop the query optimizer and query processor modules. The query optimizer computes the estimated costs of all feasible plans of evaluating a given query, examining different orderings of tables when joining them and all the access methods at its disposal. The least-cost plan (which typically consists of a sequence of query-processing steps) selected by the query optimzier is passed to the query processor, which then executes each step of the plan. However, relational database systems make modeling and querying of data that is naturally hierarchical in structure awkward and inefficient. The object-relational model of data, which combines the relational model of data and an object-oriented model of data, reintroduced the pointer concept of hierarchical and network database systems in the form of object identifiers. An object identifier is a logical pointer, not a physical pointer, and makes it possible to naturally model and access hierarchical structures of data. Object-relational database systems represent a combination of the best of both the hierarchical database systems and relational database systems by allowing both "ad hoc" content-based queries and navigational queries.

XML has come about completely outside the database systems domain. The hierarchical structure of elements in XML certainly called for query languages that would support navigational access to different parts of XML documents at different levels of hierarchy. However, just as the users of database systems have discovered over the past 30 years, we believe that users of XML databases will benefit significantly from structure-agnostic, that is, non-navigational, content-based queries. Simply put, a good segment of XML users would want to specify in their queries only the search conditions (and output elements), without having to know and specify the detailed hierarchical structure of the XML documents or have to give hints for efficiently processing the queries.

3 ILLUSTRATIONS

To make the concept concrete, suppose that we have XML documents on movies, represented as in Figure 1a. If we are to search for movie titles whose genre is 'action', release year is '1994', and whose stars include 'Jean Reno', we would like to be able to state the search conditions simply as

genre = "action" and year = "1994" and actor = "Jean Reno"

regardless of the structure of the XML documents. Then a non-navigational content-based query (in Xquery) that includes the above search conditions would look something like

```
    for $t in doc()//title
    where genre = “action” and year = “1994” and actor = “Jean Reno”
    return $t
```

The same query, when expressed in navigational style, would look like

```
    for $m in doc ()//movie
    where $m/genre[text() = “action”] and $m/year[text() = “1994”] and $m/actor
          [text() = “Jean Reno”]
    return <title>$m/title/text()</title>
```

```
<movie>
  <year>1994</year>
  <country>America</country>
  <country>France</country>
  <genre>drama</genre>
  <genre>action</genre>
  <title>Leon</title>
  <director>Luc Besson</director>
  <actor>Jean Reno</actor>
  <actor>Natalie Portman</actor>
</movie>
```

Figure 1a. an XML document about a movie (using elements)

Of course, to be able to formulate non-navigational content-based queries against XML documents, the users would need to know at least the names of the elements/attributes and data types for the data values. Just as was the case with relational database systems, the onus of computing all possible paths along the XML documents’ hierarchical structure would fall on the query optimizer and query processor of the XML server.

There are a few reasons that we believe that it is imperative that XML users be able to formulate non-navigational content-based queries, besides the obvious observation that it can be tedious to have to specify the navigational paths in a query. First, the flexibility of XML allows users to specify the same contents in a number of different ways, and as such it becomes problematic for the users to have to content with all the disparate ways in which contents may be represented in XML. At the risk of belaboring the point, let us consider Figures 1b and 1c. These figures show how the same simple contents, represented in Figure 1a, can be represented in different ways, using element hierarchies, and attributes. (There are various additional ways.) The same structure-agnostic query shown previously would work for each of these alternate representations. The navigational query for Figure 1b would look like

```
    for $m in doc ()//movie
    where $m//genre[text() = “action”] and $m//year[text() = “1994”] and
              $m//actor [text() = “Jean Reno”]
    return <title>$m//title/text()</title>
```
(* We note that the above query would also work as navigational query for Figure 1a. *)

```
<movie>
  <general_info>
      <year>1994</year>
      <country>America</country>
      <country>France</country>
      <genre>drama</genre>
      <genre>action</genre>
  </general_info>
  <detail_info>
     <title>Leon</title>
     <people>
        <director>Luc Besson</director>
        <actors>
           <actor>Jean Reno</actor>
           <actor>Natalie Portman</actor>
        </actors>
     </people>
    </detail_info>
</movie>
```

Figure 1b. The same XML document about a movie (using an element hierarchy)

For Figure 1c, the navigational query would look like

```
    for $m in doc ()//movie
    where $m/*[@genre = “action”] and $m/*[@year = “1994”] and
            $m/*[actor = “Jean Reno”]
    return <title>$m/*/@title</title>
```

```
<movie>
  <production year=”1994”
                 country=”America France”>
  </production>
  <detail_info genre=”drama action”
                 title=”Leon”>
  </detail_info>
  <people actor=”Jean Reno”
           director=”Luc Besson”>
  </people>
</movie>
```

Figure 1c. the same XML document about a movie (using attributes)

Further, the same contents may be organized in a number of different hierarchical structures. Figures 2a and 2b represent the XML document of Figures 1 using hierarchical structures. Again the same structure-agnostic query shown previously would work for each of these alternate representations. However, the navigational query for Figure 2a would look like

```
for $y in doc ()//year
where $y[text() = “1994”] // genre[text() = “action”] // actor[text()= “Jean Reno”]
return $y//title/text()</title>
```

```
<year>1994
  <country>America
    <genre>action
      <movie>
        <title>Leon</title>
        <director>Luc Besson</director>
        <actor>Jean Reno</actor>
      </movie>
  …
  </country>
  …
</year>
…
```

Figure 2a. The same XML document about a movie (using an element hierarchy)

For Figure 2b, the navigational query would look like

```
for $g in doc ()//genre
where $g[@* = “action”] // year[*= “1994”] // actor[text() = “Jean Reno”]
return <title>$g//title/text()</title>
```

```
<genre type=”action”>
   <country name=”America”>
     <year><yyyy>1994</yyyy>
     <movie>
         <title>Leon</title>
         <people director=”Luc Besson”
                    actor=”Jean Reno”>
         </people>
     </movie>
…
   </country>
…
</genre>
```

Figure 2b. The same XML document about a movie (using an element hierarchy and attributes)

4 CONCLUDING REMARKS

In this article, we discussed why structure-agnostic or non-navigational content-based query facilities will complement the current navigation-based query languages for XML. To make the discussion concrete, we showed some examples of how many different ways in which the same contents may be represented in XML. It is obvious that once we consider “similar” contents, rather than the exact same contents, the problem becomes compounded. To retrieve “similar” XML documents that satisfy certain search conditions, one needs to consider non-identical XML documents that nonetheless contain some common elements and attributes, in a wide variety of hierarchical structures. The problem becomes even more difficult in a distributed, multiple-database environment, including the Web. Although XML is being touted as “the” standard for data interchange, it will be used as the basis of data interchange standards for a wide variety of different application areas, each with its own vocabulary and data semantics. In other words, there will not be a single universal XML data interchange standard. As such, even if different segments of the XML users adopt particular data interchange standards, the need to integrate XML documents across multiple segments will drive the need for non-navational content-based query facilities.

We have designed and implemented a non-navigational content-based query language and a query processor for the language. We will report details of our implementation and experiments in a forthcoming paper. Meanwhile, we believe that non-navigational query support for XML, in particular, query optimization and query processing, is an important area for further research, and would like to encourage the XML data management research community to follow up on our research.

REFERENCES

[XSL] S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, and S. Zilles: Extensible Stylesheet Language (XSL) Version 1.0, W3C Proposed Recommendation Aug. 2001

[Abit1997] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener, “The Lorel Query Language for Semistructured Data”, *International Journal on Digital Libraries*, Apr. 1997

[Oracle] S. Banerjee, Oracle XML DB, Oracle Corporation Technical White Paper. release 9.2, Jan. 2002

[XQuery] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, J. Siméon, XQuery 1.0: An XML Query Language, W3C Working Draft 16 Aug. 2002

[Deut1998] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu, XML-QL: A query language for XML, Submitted to the W3C 19 Aug. 1998

[Howl2002] S. Howlett and D. Jennings: “SQL Server 2000 and XML: Developing XML-enabled data solutions for the web”, *MSDN magazine*, Jan. 2002

[IBM] IBM Corporation, DB2 XML Extender, IBM Corporation, 2000

[XML] W3C Consortium, XML1.0 (Second Edition), W3C Recommendation 06 Oct. 2000, available at http://www.w3.org/TR/REC-xml

[XPath] W3C Consortium, XML Path Language (XPath) Version 1.0, W3C Recommendation 16 Nov. 1999

About the authors

Won Kim is President and CEO of Cyber Database Solutions (http://www.cyberdb.com/) and MaxScan (www.maxscan.com) in Austin, Texas, USA. He is also Dean of Ewha Institute of Science and Technology, Ewha Women's University, Seoul. Korea. He is Editor-in-Chief of ACM Transactions on Internet Technology (www.acm.org/toit), and Chair of ACM Special Interest Group on Knowledge Discovery and Data Mining (www.acm.org/sigkdd). He is the recipient of the ACM 2001 Distinguished Service Award.

Wol-Young Lee is a Ph.D. candidate in the Department of Computer Science and Engineering at Ewha Women's University in Seoul, Korea. Her research interests include programming languages, XML, and database systems). Her email is wylee at ewha.ac.kr.

Hwan-Seung Yong is associate professor with the Department of Computer Science and Engineering at Ewha Women's University in Seoul, Korea. His research interests include multimedia database systems, data mining and bioinformatics, Internet computing. He received a Ph.D. degree in computer engineering, Seoul National University. His email is hsyong at ewha.ac.kr.