Querying XML data : the DQL language

Emmanuel Bruno
Laboratoire SIS
Université de Toulon et due Var
BP 132, 83 957 La Garde, France
04 94 14 22 20
bruno@univ-tln.fr

Jacques Le Maitre
Laboratoire SIS
Université de Toulon et due Var
BP 132, 83 957 La Garde, France
04 94 14 20 06
lemaitre@univ-tln.fr

Elisabeth Murisasco
Laboratoire SIS
Université de Toulon et due Var
BP 132, 83 957 La Garde, France
04 94 14 26 21
murisasco@univ-tln.fr

ABSTRACT

This poster describes the DQL language : a new XML data manipulation language. DQL is an extension of OQL for the manipulation of tree-like and forest-like data. It integrates XPath location paths and provides us with the classical SQL operators and some specific tree transformation operators. The first version of DQL has been implemented in Java using the DOM and the SAX API's.

Keywords

Semi-structured data, XML, XPath, Query language.

1. INTRODUCTION

XML can serve as a simple exchange format, or as a data model. In both cases, we need to have a manipulation language at our disposal. Developing such languages is a hard task, due to the complexity of the structures which are involved (e.g. trees and graphs) and the impossibility of relying on a schema. Over the past years, numerous data manipulation languages have been developed for SGML, HTML, XML or semi-structured data ([1], [2], [3], [4], [5], [6], [7]). Recently, a W3C working group has started to specify a new XML data query language : the Xquery language [9]. XQuery is derived from Quilt [2], which in turn borrowed features from several other languages cited above. Despite their intrinsic qualities, none of the above languages has been unanimously accepted, since none of them has been recommended by the W3C.

In this poster, we propose a new XML data manipulation language: the DQL language which is an extension of OQL for the manipulation of tree-like and forest-like data. DQL integrates Xpath location paths [8] and provides us with the classical SQL operators (select…from…where, group…by, sort, etc) and some specific tree transformation operators which give to DQL the power of XSLT[10].

2. DQL OVERVIEW

DQL works with two types of value: trees and forests. The type of a tree can be text, number, boolean, attribute or element. A forest consists of a list of trees. A query is a functional expression which is composed of predefined operators and user-defined functions. DQL includes four kinds of operators :

1. extraction of fragments from documents. The location of each fragment is defined with an Xpath expression,
2. construction of elements,
3. select…from…where, group…by and sort…by operators,
4. transformation of document fragments by adding or removing sub-components.

Moreover, it is possible to define local and global variables. The use of XPath to extract fragments from documents and the transformation operators are the two original features of DQL. We give examples of DQL queries which are evaluated on the document of figure 1, which is bound to the $mybib variable.

```
<bib>
  <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher><price>65.95</price>
  </book>
  <book year="2000">
    <title>Data on the Web</title>
    <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price>39.95</price>
  </book>
  <book year="1999">
    <title>The Economics of Tech. and Content for Digital TV</title>
    <editor><last>Gerbarg</last><first>Darcy</first>
     <affiliation>CITI</affiliation></editor>
   <publisher>Kluwer  Academic Publishers</publisher>
   <price>129.95</price>
  </book>
</bib>
```

Figure 1 : A well-formed XML document

(Q1) Catalogue of book titles.

```
<catalog count = count($MyBib//book)>$MyBib//book/title</>
```

(Q2) Name and first name of authors that have published more than one book in year 2000.

```
select  [$a//last/*, $a//first/*]
from    $a in distinct($mybib//book[@year = 2000]/author)
where  count($mybib//book[author = $a]) > 1
```

(Q3) Title of books having an author with the name set after the firstname.

```
select  $b/title
from   $b in $mybib//book[author/last/preceding-sibling::first]
```

The predicate last/preceding-sibling::first is true if a first element precedes a last element and if both are siblings. This kind of query is useful to detect irregularities in XML documents when they are not valid.

(Q4) Concatenation of the name with the firstname of each author or editor.

```
replace //name as $n by <name>[$n/last/*, " ", $n/first/*]</>
from    $mybib
```

Without the replace operator, this query would need a complete reconstruction for each book. This would be a complex solution when books have irregular structures.

(Q5) Removing editor affiliations.

```
remove //affiliation from $mybib
```

(Q6) Transformation of the “year” attribute into a year element.

```
replace //book as $b
by <book> <year>$b/@year</> content($b) </>
from    $mybib
```

(Q7) Simultaneous transformations (Q4 + Q5 + Q6).

```
replace
  //name as $n by <name>[$n/first, $n/last]</>
  //affiliation by void
  //book as $b by <book><year>$b/@year</> content($b)</>
from $mybib
```

(Q8) Statistics.

```
let $bks_MK = $mybib//book[publisher="Morgan Kaufmann"]
in   <stat count=count($bks_MK)
            average_price = avg($bks_MK//price) />
```

3. RELATED WORKS

Over the past years, numerous query languages for semi-structured or XML data have been proposed. Four of these languages have emerged : Lorel [1] ; XML-QL [5] ; YATL [4] and XQuery [9]. They are very complete and powerful. They have a functional semantics and they use tree or path pattern-matching to locate fragments from documents. In this poster, we compare DQL with these languages, highlighting its originality.

We consider, for this comparison, the following features : (1) the select…from…where operator, (2) the pattern-matching of fragments of documents, (3) the preservation of reading order and (4) the transformation operators.

1. Lorel and DQL have the select…from…where operator including path patterns in the from clause. The other three languages propose a specific construct which has nevertheless a similar semantics. In XML-QL, it is the where…construct operator; in XQuery, it is the for…let…where…return operator; in YATL, it is the make…match…where operator. The reason why they adopt a specific syntax is not very obvious. We think that it is better to respect the syntax of operators imported from SQL or OQL; many programmers are familiar with it.
2. XML-QL and YATL use tree patterns. XML-QL expresses them with the same syntax as XML elements. Moreover, XML-QL enables regular expressions in order to constrain the nesting of matched subtrees. Lorel uses path patterns, issued from the POQL language [3]. These patterns are first class citizen’s. Therefore it is possible to compare paths or to associate some constrains with them. However it is not very easy to express sibling relations. XQuery and DQL use XPath locations paths. While DQL implements them in their whole, XQuery implements only a subset of them. In particular, sibling relations, in XQuery, are expressed by two operators borrowed from XQL : before and after when preceding-sibling and following-sibling are axis of XPath.
3. Document querying is not only based on structural hierarchy but also on the reading order of elements. XQuery, YATL and DQL preserve reading order while Lorel and XML-QL do not preserve it (even if they propose an ad-hoc mechanism).
4. As far as we know, transformation operators like DQL replace or remove are not included in Lorel, XML-QL, YATL or XQuery. Consequently, for any transformation, the user has to explicitly define the structure of the result document and he needs to have a complete knowledge of the source structure. We think that these operators are one of the originalities and strength of DQL.

4. CONCLUSION

DQL is a SQL-like functional language which integrates (1) Xpath location paths, (2) traditional SQL operators and (3) powerful tree transformation operator. The first version of DQL has been implemented in Java using the DOM and the SAX API’s. In the future, we will focus on the two following points : (i) making DQL fully compliant with XML query requirements [8] and (ii) working on the query optimisation issue.

5. REFERENCES

[1] Abiteboul, S, Quass, D, McHugh J., Widow, A., Wiener J. (1997). The Lorel query language for semi-structured data. In Int. Journal on Digital Libraries, 1(1),pp. 68-88.

[2] D. Chamberlin, J. Robie, D. Florescu, Quiult : An XML query language for heterogenous Data sources, WebDB (Informal Proceedings) 2000, pp 53-62.

[3] V. Christophides, S. Abiteboul, S. Cluet, M. Scholl, “From structured Documents to Novel Query Facilities”, in Proc. of SIGMOD ' 94, Minneapolis, US, 1994, pp. 313-324.

[4] S. Cluet, C. Delobel, J. Siméon and K. Smaga, Your Mediators Need Data Conversion!, ACM SIGMOD' 98 International Conference on Management of Data, Seattle, Washington, June 1998.

[5] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu: XML-QL: A Query Language for XML. Proc. of the Eighth Int. World Wide Web Conference (WWW' 8), Toronto, Canada, 1999.

[6] J. Le Maitre, E. Murisasco, M. Rolbert, "SgmlQL, un langage d’interrogation de documents SGML ", Actes des Xièmes journées Bases de Données avancées (BDA ' 95), Nancy, 1995, pp. 431-446.

[7] J. Robie, J. Lapp, D. Schach, XML query language (XQL). Online : http://www.w3.org/TandS/QL/QL98/xql.html

[8] XPath XML Path Language (1999), W3C Rec. http://www.w3.org/TR/XPath

[9] XQuery, a Query language for XML (2001), W3C WD. http://www.w3.org/TR/xquery

[10] XSL Transformations (1999), W3C Rec. http://www.w3.org/TR/XSL