XML-BASED QUERY LANGUAGES USED IN MULTIMEDIA

Sabin BURAGA[1], Mihaela BRUT[2]
"Al. I. Cuza" University of Iaşi
Berthelot, 16, RO-700483 Iaşi
[1] *busaco@infoiasi.ro*
[2] *mihaela@infoiasi.ro*

Abstract. *In this paper, we present the general framework of the XML information retrieval on Web, with a special lookout to the XQuery language – proposed recommendation of the World-Wide Web Consortium. Also, we focus on the different possible techniques for querying the multimedia information annotated in SMIL.*

Keywords: *query languages, XML documents, multimedia retrieval.*

Introduction

The XML language (Extensible Markup Language) [9] as the standard meta-language applied to define markups for Web documents is a key technology in defining the structured and semi-structured collections of data. XML is already used to represent many different kinds of data: web pages, web messages, electronic books, e-business data and applications.

In addition, some systems offer XML views of non-XML data sources such as relational databases, allowing XML-based processing of data that is not physically represented as XML. Also, using the XML structure of the Web documents, the XML-based query languages provide different ways of information retrieval.

In this paper, we shall present the general framework of the querying XML documents problem, followed by a survey of query languages and techniques proposals, emphasizing the XQuery language [9] – the Web Consortium recommendation, currently in process of standardization.

We shall discuss the possibility of querying Web multimedia information by applying XQuery constructs to multimedia Web documents expressed by SMIL (Synchronized Multimedia Integration Language) [8] elements, in order to obtain the results in the same format.

Querying Issues in the Web Context

The World Wide Web space already became a universe of information, being a needful work instrument for all researchers. Many scientific data collections are now available on the Web. The problem of the efficient access and management of the growing information became more and more critically. The huge quantity of available data makes the results of searching a certain subject using a traditional search engine to include hundreds or even thousands Web pages, often provided in an irrelevant order of significance. The main reason is that the actual search engines (e.g. Altavista or Google) do not take into account the structure of the Web documents containing the requested keywords. The XML query languages and techniques tried to address this deficiency by extrapolating the techniques of querying databases and processing text documents for exploiting the XML documents structure.

As a standard recommended by the Web Consortium, XML is considered as the data format for information interchanging between the diverse Internet applications. The XML popularity is mainly due to its flexibility in the representation of many data types. The use of markups give to the XML language the possibility of self-description, and its extensible nature makes possible the definition of new document types, with a special destination.

Alongside XML, appeared a series of other Web standards which were adopted, as auxiliaries, in the query languages specifications. For example, XML Schema [9] is a formalism which can be used to define new data-types for querying results. XPath [9] language offers a notation for selecting elements from an XML document. XSLT (Extensible Stylesheet Language Transformations) [9] provides a language for transforming an XML document from a representation to another (for example, from XHTML to SMIL).

Because XML is the standard format for interchanging Web information, it is naturally that queries applied to different types of documents to be expressed as queries on XML data. For this reasons, the necessity of a standard specialized XML query language became stringent.

Many of the important requirements for a querying language – notably, XQuery [8] – follow:

- data extraction from huge XML documents, by homogeneously processing both the structured part (elements, attributes, values) and the properly text;
- syntax based on other XML standards, such as XPath, XPointer, XSLT;
- XML data transfer between documents having different ontologies (DTDs);
- querying the distributed data, in different XML formats, and the integration of XML results from multiple sources;
- support for standard querying operations: selection, extraction, reduction, reorganization, combination;
- verifying the rightness of query results.

Alongside the query languages, there where developed many software tools (notably COVA) which implement them together with other specific retrieval techniques.

In order to pass towards the Semantic Web, there where developed a series of XML-based languages specialized in the modelling of knowledge, in the same time appearing specific query languages for documents marked up according to these: RDQL (for RDF) or DQL (for DAML+OIL and OWL) [9].

XML Query Languages and Techniques

The query languages for XML documents are based on the following idea: they offer to the user the possibility to formulate a query in a specific manner and generate a new XML document as a pattern for this query. It shall be compared with the target XML document and shall be retained and returned only matched data, under the restructured form of a new XML document. The query languages themselves are implemented as XML documents.

We present below XML query languages and techniques foregoing the XQuery language [9].

XQL

The XML Query Language (XQL) [6, 9] offers the possibility of filtering and extracting information from XML documents using a pattern modelled after the directory notation. The relation between tags is referred as that between (parent) directories and their (child) sub-directories. Inspired by XPath language, XQL accepts the "/" character to be used to specify the sub-tags and "@" – for attributes. Complex queries could be constituted by using different logical and relational operators.

For example, “author[name = ‘Berners-Lee’ and @year gt 2000]” specify all books which author is Tim Berners-Lee, and where published after 2000 year.

XML-QL

XML-QL [9] is a query language which allows the extraction of information from XML documents by means of an implemented WHERE-CONSTRUCT command, analogous to the SELECT-WHERE construct from SQL or other query languages for semi-structured data. The "WHERE" part of the command defines the interrogation and the "CONSTRUCT" part specifies the modality of taking over the result, as in the following example:

```
WHERE <book>
          <domain>Computer Science </>
          <title> $t </>
          <author> $a </>
       </> IN www.abc/bibl.xml
CONSTRUCT <bibliography>
               <author> $a </>
               <title>  $t </>
           </>
```

Among the XML-QL facilities there could be mentioned the references to the derived or circular data structures (by means of regular expressions), derived queries, and integration of queries inside other documents.

XML-GL

The *XML-GL* language [6] is a graph-based query language with both its syntax and semantics defined in terms of graph structures and operations. Although the queries are formulated visually, the mechanism is too sophisticated for an ordinary user. An extension of XML-GL, named *XML-GLrec*, was developed. This approach allows moreover representing XML simple links and generic recursive queries. Derived from XML-GL, *visXcerpt* [1] is a visual querying language for XML data.

Web Query Graphical Language (WQGL)

To assist users to prepare queries, we designed an XML-based markup language, called *Web Query Formulating Language* (*WQFL*) [4, 5]. The main goal of WQFL is to permit obtaining Web matched-pages with complex and flexible queries. Each query shall be modeled by WQFL and a WQFL document will be created for each found page.

In the designing phase, we encode the Web pages structural information [5]. According to the given potential of WQFL language, some users would like the graphical content to be found on top of the Web pages and to consist of maximum 4 paragraphs etc. That information is stored into WQFL documents. Each found Web document will be processed and it will be retained only the position (top and bottom) and the occurrences of some HTML (or XML) elements and attributes. For each element, we will retain three values that represent the occurrences of that element on top, middle and bottom of the Web page. For the entire user's query, a WQFL document (which will be locally stored) is generated.

Instead of HTML element names, the WQFL documents can include position occurrences information of any XML tags (such as SMIL, XHTML or MathML elements). From this point of view, WQFL can be viewed as a query language for XML data.

More details about WQFL can be found in [4].

XQuery Language

The Web Consortium constituted the XML Query working group in order to design the *XQuery* [9] language intended for querying the XML documents. XQuery is a functional language, containing some expression types which can be combined, and being based on the datatype system from XML Schema, so that to be compatible with the related XML standards.

The initial design of the XQuery language is focused on the information retrieval by querying the desired XML documents.

We present below a syntactic survey of the XQuery language, followed by a query example applied to a SMIL document.

The XQuery syntax was constituted by incorporating the numerous influences, among which the most important are the following standards: XPath (XQuery being a superset of XPath), XML Schema, XSLT, and XML itself (see [2] and [8] for details).

Being a functional language, XQuery is constituted by expressions that return values and do not have side effects. XQuery has several kinds of expressions – in majority, composed from lower-level expressions, combined by operators or keywords. The simplest kind of XQuery expression is a literal, which represents an atomic value, and could be numerical (integer, decimal, double) or strings.

XQuery allow creating atomic values of other types – such elements, attributes, text nodes, processing instructions, and comments – by calling constructors. A *constructor* is a function that creates a value of a particular type from a string containing a lexical representation of the desired type. In general, a constructor has the same name as the type it constructs. For example, the following constructor creates a value of type date: date("2003-12-14"). In XQuery could be used variables, their names prefixed by “$” symbol.

XQuery provides a core function library, and a mechanism whereby users can define additional functions. For example, the *substring()* function could be used to extract seven characters from a string, beginning with the fourth one: substring(“Tim Berners-Lee", 4, 6).

For referring to a specific XML sub-element, there could be used XQuery path expressions, which are based on the syntax of XPath [8]. A *path expression* consists of a series of steps, separated by the slash character (e.g. *doc("books.xml")/bib/book*). For selecting the sub-elements which satisfy certain conditions, XQuery allows the use of *predicates*. In the following example, *@year=”2003”* is the predicate:
doc("books.xml")/bib/book[@year="2003"].

One of the most powerful features in XQuery is FLWOR expressions. They are similar to the SELECT-FROM-WHERE statements in SQL. The name FLWOR is an acronym, standing for the first letter of the clauses that may occur in a FLWOR expression:

- **for** clauses: associate one or more variables to expressions, creating a tuple stream in which each tuple binds a given variable to one of the items to which its associated expression evaluates;
- **let** clauses: bind variables to the entire result of an expression, adding these bindings to the tuples generated by a for clause, or creating a single tuple to contain these bindings if there is no for clause;
- **where** clauses: filter tuples, retaining only those tuples that satisfy a condition;
- **order by** clauses: sort the tuples in a tuple stream;
- **return** clauses: build the result of the FLWOR expression for a given tuple.

For example, if the *books.xml* document contains elements <book> having the attribute “year” and the sub-elements <title>, <author>, and <publisher>, the next query constructs a new element named <results>, containing a list of <result> elements, each having an <author> sub-element and a list of <title> sub-elements corresponding to all books written by the respective author:

```
<results>
{
  let $a := doc("books.xml")/bib/book
  for $name in
         distinct-values($a/author),
  order by $name
  return
    <result>
      <author> { $name } </author>
      {
           for $b in
             doc("books.xml")/bib/book
           where some $ba in $b/author
             satisfies
                ($ba/author = $name)
           return $b/title
       }
    </result>
}
</results>
```

The XQuery language provides, also, an entire set of operators, such as arithmetic, relational and traversal operators (see [9] for details).

Using XQuery Constructs for Synchronized Multimedia Retrieval

The paper proposes an XQuery technique that can be used to search within the multimedia presentations written in SMIL. The SMIL language is a Web Consortium’s standard in order to annotate information about the temporal scenario of different Web multimedia objects.

Examples

We consider the following SMIL document, called books.smil, which contains information about diverse published books and their associated synchronized video-clips (we omit some syntactic details):

```
<smil>
<head></head>
```

```
<body>
<seq>
<b:books xmlns:b=" books.dtd">
 <par>
   <seq>
   <video   region="rvideo"   begin="0s"
end="15s"
     id="v1" src="../movies/web.avi" />
   <img region="rvideo"
        src="../images/web.jpg"
        dur="20s" begin="v1.end">
   </seq>
   <text region="rtext"
        dur="v1.dur+20s">
      <b:title>Web
Technologies</b:title>
      <b:author>S. Buraga</b:author>
<b:publisher>MatrixRom</b:publisher>
      <b:year>2001</b:year>
   </text>
 </par>
 <!-- other constructs -->
</seq>
</body>
</smil>
```

To generate a SMIL slide-show with all books published by a certain publishing house (e.g. Matrix Rom) after 1998, sorted by title, we can compose the following XQuery assertions:

```
<seq>
 {
  for $b in doc("books.smil")
             /body/b:books/seq/par/
  where $b/text/b:publisher =
"MatrixRom"
     and $b/text/b:year > 1998
     order by  $b/text/b:title
     return
    <par>
       <video region="rvideo" dur="30s"
           src="$b/seq/video[@src]" />
        <text region="rtext" dur="30s">
            <b:title>
                {$b/text/b:title}
            </b:title>
            <b:year>
                Year: {$b/text/b:year}
            </b:year>
         </text>
     </par>
 }
</seq>
```

The result of this query could be:

```
<seq>
    <par>
      <video region="rvideo" dur="30s"
           src="../movies/fla.avi" />
          <text region="rtext"
dur="30s">
            <b:title>
                Formal Languages
                and Automata
            </b:title>
            <b:year>
                Year: 1999
            </b:year>
          </text>
   </par>
   <par>
     <video region="rvideo" dur="30s"
           src="../movies/web.avi" />
          <text region="rtext"
dur="30s">
            <b:title>
                Web Technologies
            </b:title>
            <b:year>
                Year: 2001
            </b:year>
          </text>
     </par>
</seq>
```

In the second example, we'll generate a SMIL presentation which includes all books having in the title "Language" or "Programming" words. For each found book, the associated video-clip and other relevant information is provided.
The XQuery construct is:

```
<seq>
for $b in doc("books.smil")
                /body/b:books/seq/par
   let $e :=
    $b/text/b:title[contains(string(.),
                 "Language")
         or contains(string(.),
"Programming")]
   where exists($e)
   return
    <par>
      <video region="rvideo" dur="30s"
          src="$b/seq/video[@src]" />
      <text region="rtext" dur="30s">
        <b:title>
              {$e}
        </b:title>
        <b:year>
              $b/text/year
        </b:year>
      </text>
    </par>
</seq>
```

Implementation

The XML processing techniques use both DOM (Document Object Model) [2, 9] and SAX (Simple API for XML) [2] models, implemented in PHP. In order to process XQuery constructs, the *XQuery Lite* [7] PHP library is used. To view SMIL presentations, different players can be used. Our tests adopted GRiNS and RealOne players.

Conclusion

In this paper, we presented first a short survey on XML-based query languages and their applications. The paper focused on XQuery language – proposed recommendation of the World-Wide Web Consortium.

One of the interesting problems regarding data retrieval is the multimedia retrieval. For this, the paper presented different techniques of generating SMIL presentations starting from existing (semi-)structured synchronized multimedia information. These techniques used XQuery constructs.

As a further work, we intend to investigate different approaches in order to automatically formulate XQuery constructs from a set of querying templates. Another direction – following an idea presented in [3] – is to generate RDF assertions that can be used to associate metadata for multimedia resources, as a possible contribution to Semantic Web.

References

[1] Berger, S. *et al.* (2003) *Xcerpt and visXcerpt: From Pattern-Based to Visual Querying of XML*, Proceedings of Intl. Conf. on Very Large Databases – VLDB03.

[2] Buraga, S. (2001) *Web Technologies*, Matrix Rom, Bucharest.

[3] Buraga, S. (2002) *Modeling Relations Between Web Resources*, Transactions on Automatic Control and Computer Science, vol.47 (61), No.2, Politehnica Press, Timisoara.

[4] Buraga, S., Brut, M. (2002) *Different XML-based Search Techniques on Web*, Transactions on Automatic Control and Computer Science, vol. 47 (61), No.2, Politehnica Press, Timisoara.

[5] Buraga, S., Brut, M. (2001) *A Proposal for a Web Structural Search Language Based on XML Technologies*, Scientific Annals of the "Al.I.Cuza" University of Iasi – Computer Science Section, Tome X, 2001, "Al.I.Cuza" University Press House, Iasi.

[6] Ceri, S. et al. (1999) *XML-GL: A Graphical Language for Querying and Restructuring XML Documents*, Proceeding of the Eight International World-Wide Web Consortium – WWW8, Toronto.

[7] * * * (2003) *XQuery Lite*: http://sourceforge.net/projects/phpxmlclasses/

[8] * * * (2003) *World-Wide Web Consortium's Technical Reports*: http://www.w3.org/TR/

[9] * * * (2003) *World-Wide Web Consortium's Activity on Query Languages*: http://www.w3.org/XML/Query