USING DATA TOGETHER WITH METADATA TO IMPROVE XML INFORMATION ACCESS

Alda Lopes Gançarski
GET/INT, CNRS Samovar
9 rue Charles Fourier, 91011 Évry, France

Pedro Rangel Henriques
University of Minho
Department of Informatics, 4710 Braga, Portugal

Keywords: XML, XQuery, ontology, Dublin Core, metadata, SPARQL.

Abstract: In this paper, we address the use of data together with metadata to improve information access to XML document collections. We first emphasize the possibility to associate meta-information to parts of XML documents, and not only entire documents. This is accordingly to the fact that XML elements are considered as retrieval units. We, then, propose to explicitly search the desired information using a query language that is composed of XML search and metadata search. We focus our ideas in two types of metadata: Dublin Core and ontologies.

1 INTRODUCTION

XML information access is done using structured query languages such as XPath (Berglund et al., 2005) and XQuery (Boag et al., 2006), the standard proposed by the W3C. These query languages are being extended with the possibility of associating a *score* to an expression that verifies if some phrase exists in the content of some element, as in traditional information retrieval. This functionality is included in the *Full-Text* language also proposed by the W3C (Amer-Yahia et al., 2006).

To improve data processing, document collections and Web resources are associated with semantic descriptions, or metadata. In order to be able to exchange the semantics of information, one first needs to agree on how to explicitly model it. This can be done using a standard set of characteristics, like title or author, or using a more sophisticated description in the form of ontologies. In the first case, a standard set of such characteristics was developed in the *Dublin Core Metadata Initiative* (Dublin Core Metadata Initiative, 2006). This set is composed, among other concepts, of *Creator*, *Date*, *Format*, *Language*, *Publisher*, *Title*, *Subject and Keywords*, *Abstract*.

Concerning the second case, an ontology is a formal explicit specification of a shared conceptualization. Using an ontology, any kind of description can be made about a resource. Ontologies can be used to annotate data with labels indicating their meaning, thereby making their semantics explicit and machine-accessible. To formally define ontologies, W3C has proposed the *Resource Description Language* (RDF) (Manola and Miller, 2004), which allows the representation of metadata about Web resources.

Metadata search can be done using a simple natural language expression (Finin et al., 2005), using a navigational structure (Fluit et al., 2005) or using a dedicated query language (Corby et al., 2004) like SPARQL (Prud'hommeaux and Seaborn, 2006), the W3C recommendation for the standard query language for RDF. In general, works are devoted to search information directly in documents or indirectly in ontology's concepts to get associated resources, not using data and metadata together in the search process. In this article we exploit the use

Lopes Gançarski A. and Rangel Henriques P. (2007).
USING DATA TOGETHER WITH METADATA TO IMPROVE XML INFORMATION ACCESS.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 328-332
DOI: 10.5220/0001262003280332

of XML documents together with the respective metadata to access information. We believe that both may be interesting to the user and can help him to find interesting information.

Metadata descriptions may be done at elements level, giving semantic information for elements retrieval. This is discussed in Section 2. Section 3 explains our proposal for the search based on both data and metadata showing some examples. The article finishes with a brief conclusion, indicating some future work.

2 ELEMENTS METADATA

In accordance with XQuery, XML elements are retrieval units and, thus, it is interesting to allow for metadata descriptions associated to elements, instead of only having document metadata.

To exemplify elements metadata, suppose a book where chapters, not only are written by different authors, but also cover many sub-subjects of the book's main subject. So, different meta-information can be associated to each chapter. Also, a useful metadata concept can be the date of the last update of an element.

If RDF is used to specify Dublin Core (DC) metadata and ontologies, elements can be referenced using *URI references*. A URI reference (or *URIref*) is a URI together with an optional fragment identifier at the end. For example, the URI reference

http://www.example.org/article.xml#section2

corresponds to the second section element in the *article.xml* document.

3 XML COMPOSED SEARCH

To perform information search based on both data and metadata, some extensions to XQuery must be done. The next sections explain such extensions when there is DC metadata and when metadata is in the form of ontologies, respectively.

3.1 Using Dublin Core Metadata

If metadata follows the DC proposal, metadata can be expressed in different ways.

When documents are in HTML format, DC metadata can be embedded in the document using the special *Meta* tag. An example of a *Meta* tag is in the following document:

```
<Html>
  <Head>
      <Title>XML standard</Title>
      <Meta Name="DC.Creator"
      Content="Paul">
  </Head>
  <Body>
      <H1>XML Applications</H1>
      <P>XML stands for eXtensible …</P>
  </Body>
 </Html>
```

In this example, metadata indicates that the creator of the page is "*Paul*". This is indicated in attributes *Name* and *Content* of the *Meta* tag, respectively. Restrictions on metadata are done accessing *Meta* tags' attributes. Thus, no extensions to XQuery are needed, as indicated in Figure 1.

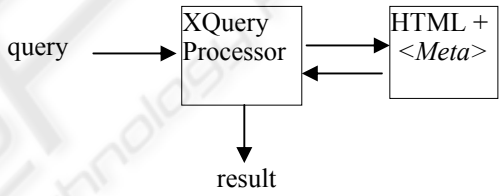

Figure 1: Using XQuery to query HTML documents including the DC <*Meta*> tag.

For example, to get HTML documents created by Pearl, the XQuery query is:

```
for $a in doc("http://...")
      /Html[.//Meta/@Name="DC.Creator" and
      .//Meta@Content="Pearl"]
return $a
```

When DC metadata is expressed in RDF, we propose an extension to XQuery in order to allow for the inclusion of metadata restrictions in structural restrictions. These restrictions are expressed using the so called *metadata:* functions.

As depicted in Figure 2, having a processor for XQuery extended with the *metadata:* functions, the user can make queries over XML collections associated to DC metadata expressed in RDF.

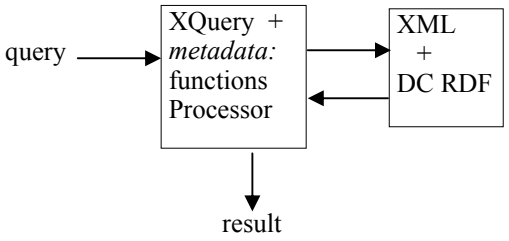

Figure 2: Using XQuery extended with *metadata:* functions to query XML documents associated to DC metadata expressed in RDF.

For example, suppose the following XML document:

```
<Book>
        <Title>XML Guide</Title>
        <Chapter>
                <Title>SGML</Title>
                <Section>…</Section>
        </Chapter>
        <Chapter>
                <Title>HTML </Title>
                <Section>…</Section>
        </Chapter>
        …
</Book>
```

Suppose this document has the following DC metadata expressed in the form of RDF triples (for simplicity, we ignore prefixes that correspond to URIs):

Book.xml#Chapter1 Creator "Kevin".
Book.xml#Chapter2 Creator "Loik".

To get chapters written by Kevin, a simple XQuery query could be:

```
for $c in doc("http://...")
      /book/chapter[metadata:Creator(.)="Kevin"]
return $c
```

In this query, a filter is imposed to chapters. In this filter, function *metadata:Creator()* returns the value of the *Creator* concept in the metadata description of a chapter. The chapter is passed to the function using the "." symbol, which expresses the current element being evaluated. Note that *metadata:* prefix indicates that *Creator* is a function about metadata.

Suppose now that there is information about the subjects treated in the chapters:

Book.xml#Chapter1 Subject "SGML".
Book.xml#Chapter2 Subject "HTML".

If the user wants chapters about SGML, he can make the following XQuery query:

```
1 for $c in doc("http://...") /book/chapter
2 score $s as
3       $c ftcontains "SGML" OR
4       metadata:Subject($c) ftcontains "SGML"
5 order by $s
6 return $c
```

In this query, the *for* clause stores in variable *$c* all the book chapters (line 1). Then, the *score* clause associates to each chapter a score stored in variable *$s* (line 2). The score represents how much the chapter is about the desired subject "SGML". In XQuery, the *ftcontains* expression verifies if some phrase exists in an element. If it is included in a *score* clause, it verifies how much the element is about the subject expressed in the phrase, i.e., the score of the element with respect to the subject. In the example query, the computation of the score takes into account both the content of the chapter (line 3) and the *Subject* concept of its metadata description (line 4). This can help making better score computations. Finally, the result of this query is the list of chapters (line 6) ordered by their score (line 5). In general, for each DC concept, a function is used to access to the respective metadata value. XQuery includes function calls. Functions can be XQuery pre-defined ones or user defined functions, such metadata ones.

Each DC concept is associated to a mapping, or table, from elements (or nodes) to metadata values. This table is created when DC metadata is associated to elements. Suppose the mapping of concept *X* is:

$MapX : node() \times xs:string$

The XQuery node test *node()* matches any node. A value is generaly represented by a string which is denoted here by *xs:*string. The metadata function *metadata:X* can, then, be defined by:

```
declare function metadata:X ($a as node())
      as xs:string
{
return MapX[$a]
}
```

Here, *$a* is the variable that stores the node. The result of the function is the value corresponding to the node in the *MapX* table.

3.2 Using Ontologies

When a XML document is associated to semantic descriptions in the form of ontologies expressed in RDF, we propose to integrate SPARQL queries in XQuery queries. This can be done by adding a new clause *metadata* to the *for* clause of both languages. As depicted in Figure 3, the user can make queries over XML collections associated to RDF metadata using a processor for XQuery extended with the *metadata* clause for SPARQL queries.

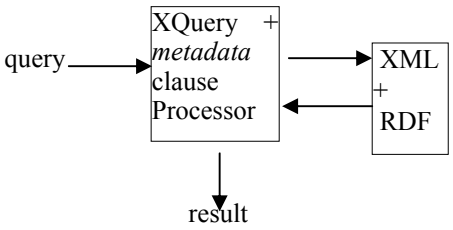

Figure 3: Using XQuery extended with *metadata* clause to query XML documents associated to RDF metadata.

Suppose the following ontology including references to elements of an XML document.

```
Book.xml                about Animals.
Book.xml#Chapter1       about Fishes.
Book.xml#Chapter2       about Birds.
Book.xml#Chapter3       about Mammals.
Book.xml#Section1       about Men.
Fishes          is      Animals.
Birds           is      Animals.
Mammals         is      Animals.
Men             is      Mammals.
Men             eats    Birds.
men             eats    Fishes.
```

If the user wants sections about animals that are eaten by men, he can specify the following XQuery query:

```
1  for $s in
2  doc("http://.../Book.xml") /book//section
3  metadata  $s  in
4       SELECT   ?s   WHERE    (?o eats Fiches)
5                              (?o eats Birds)
6                              (?s  about ?o)
7  return $s
```

In this query, the *for* clause associates to variable *$s* the set of sections of the document (line 1 and 2). The *metadata* clause (line 3) includes a SPARQL query (line 4 to 6). This SPARQL query selects all the elements (stored in variable *?s*) that are about some subject (stored in variable *?o*) which eats fishes or birds. The resulting set of elements of the internal SPARQL query (in variable *?s*) is intersected with the set of elements of the XQuery external query (in variable *$s*) to get the desired sections of the book.

To show another example, suppose the following products catalogue:

```
<products>
<product><name>Water Corola</name>
                <price>15<price>
        </product >
        <product>... </product> ...
</products>
```

Suppose also that this document is associated to the following RDF description:

```
Catalog.xml#product1   lastUpdate   "10/1/2005".
Catalog.xml#product2   lastUpdate   "10/2/1990".
```

To get products which last update is before 1990, the query is:

```
1  for $p in doc("http://.../catalog.xml") //product
2  metadata  $p  in
3        SELECT    ?p
4        WHERE (?p lastUpdate ?u)
5       AND ?u < "1/1/1990"
6  return $p
```

In this query, variable *$p* stores the product elements of the document (line 1). The SPARQL query (lines 3 and 4) stores in variable *?p* elements which last update, stored in variable *?u*, is before 1990. The content of variable *$p* in the external XQuery query is intersected with the content of the *?p* variable in internal SPARQL query, yielding the set of desired products.

SPARQL queries are included in the XQuery grammar extending the production that derives the FLWOR expressions with the *metadata* clause. The current production is:

```
[33] FLWORExpr ::= (ForClause |
        LetClause)+ WhereClause?
        OrderByClause?
        "return" ExprSingle
```

Extending it with metadata restrictions yields:

[33] FLWORExpr ::= (ForClause | LetClause)+ MetadataClause? WhereClause? OrderByClause? "return" ExprSingle

Symbol *MetadataClause* derives the *metadata* clause by the following productions:

MetadataClause ::= "metadata" "$" VarName ("," "$" VarName) "in" SPARQLQuery*

The symbol *VarName* belongs to the XQuery grammar and derives a variable name. In its turn, symbol *SPARQLQuery* derives a SPARQL query using the grammar defined in (Prud'hommeaux and Seaborn, 2006).

4 CONCLUSION

In this paper, we informally propose to integrate data and metadata search in the same query language. We believe this can help users to get useful information. As future work, we intend to formalize the proposed XPath and XQuery extensions, like the *metadata* clause. This formalization will take into account many aspects concerning metadata. For example, intra- and inter-document links must be considered.

Once the extension is formalized, we intend to create a prototype processing environment for the extended XQuery. We can use existing XQuery and SPARQL query processors integrated with editing and results visualization. We intend, then, to test the prototype and the usefulness of the approach using existing document collections and respective metadata.

REFERENCES

Amer-Yahia, S., Botev, C., buxton, S., Case, P., Doerre, J., Holstege, M., McBeath, D., Rys, M., Shanmusgasundaram, J., 2006. XQuery 1.0 and XPath 2.0 Full-Text W3C Working Draft Draft 1 May 2006, http://www.w3.org/TR/xquery-full-text/.

Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., Siméon, J., 2006. XML Path Language (XPath) 2.0 W3C Proposed Recommendation 21 November 2006, http://www.w3c.org/TR/2006/PR-xpath20-20061121.

Boag, S., Chamberlin, D., Fernandez, M., Florescu, D., Robie, J., Siméon, J., 2006. XQuery 1.0: An XML Query Language W3C Proposed Recommendation 21 November 2006, http://www.w3c.org/TR/2006/PR-xquery-20061121/.

Corby, O., Dieng-Kuntz, R. and Faron-Zucker, C., 2004. Querying the Semantic Web with Corese Search Engine, *3rd Prestigious Applications Intelligent Systems Conference (PAIS),* Valencia, Spain, 2004.

Dublin Core Metadata Initiative, 2006. URL: http://dublincore.org/ (last updated: 18 December 2006).

Finin, T., Ding, L., Pan, R., Joshi, A., Kolari, P., Java, A. and Peng. Y., 2005. Swoogle: Searching for knowledge on the Semantic Web, *National Conference on Artificial Intelligence (AAAI) 2005*, Intelligent Systems demo, Pittsburgh, Pennsylvania, USA, July 2005.

Fluit, C., Sabou, M. and Harmelen, F., 2005. Ontology-based information visualization: towards semantic web applications. In *Visualising the semantic web (2nd edition),* Valdimir Geroimenks Editor, Springer Verlag, 2005.

Manola, F. and Miller, E., 2004. RDF Primer W3C Recommendation 10 February 2004. URL: http://www.w3.org/TR/rdf-primer/.

Prud'hommeaux, E. and Seaborn, A., 2006. SPARQL Query Language for RDF W3C Working Draft, 4 October 2006. URL: http://www.w3.org/TR/rdf-sparql-query/.