

# Contribution to graphical querying language for XML semi-structured data

Sameh Ammar, Ikram Amous, Faiez Gargouri

LARIM, Route Mharza km 1,5, CP 3018, Sfax, Tunisie

ammar\_sameh@yahoo.fr, ikram.amous@isecs.rnu.tn, faiez.gargouri@fsegs.rnu.tn

## Abstract

Recently, XML has become a standard for data representation, manipulation and exchange on the web. The increase in the use of XML in many applicative domains induces a strong need for providing XML query capabilities to many users. In this paper, we propose a solution of XML graphical querying based on the XQuery language. In our approach, we propose to integrate textual, spatial and temporal meta-data in order to improve information retrieve and present document suitable to the user's needs. We, also, propose the extension of the XQuery by new operators in XQuery grammar to facilitate the use of the meta-documents.

## 1. Introduction

Recently, XML has become a standard language for data representation, manipulation and exchange on the web. The increase in the use of XML in many applicative domains induces a strong need for providing XML query capabilities to many users including those who lack in computer programming skills. For these reasons, many languages have been developed to formulate XML queries and express document transformations. XQL [11], XML-QL[12], Lorel [4] and XQuery[5] were designed for XML document retrieval and textual querying. However, these languages are still difficult for common users to use; and an intuitive graphical query languages or interfaces may help people querying data sources.

In this paper, we present a graphical language to represent user queries. We tend to make the graphical language clear and provide a user-friendly query environment. Our object is to help user to query multimedia data with their spatial and temporal relationships. We present, also, an extension to the XQuery by integrating a set of new operators in the XQuery grammar.

The present paper is organized as follows. In section two, we will briefly talk about the different query languages designed for semi-structured data. Section three develops our proposal for a graphical querying XML documents by means of examples. In section four, we introduce new operators in XQuery grammar helping the user to make easily graphical queries and we give some examples of using these operators. Finally, section five concludes this paper and briefly points out some future works.

## 2. Related works

The diffusion of XML in most applicative fields sets a pressing need for providing the capability to query XML data to a wide spectrum of users, including those with minimal or no programming skills. So, several XML query languages were proposed and extensively analyzed in the literature [1,3] before the proposal of XQuery as standard. We can distinguish two types of query languages: textual and graphical ones. XML-QL, XQL and Lorel are query language designed in order to facilitate retrieve for semi-structured data. XQuery [5] combines features from several earlier XML query languages, in particular XPath [13], XML-QL, SQL...It is designed with the goal of being expressive, exploiting the full adaptability of XML and combining information from diverse data sources [5]. All these languages were designed to be easily understood by humans, but, the number of their experts still limited.

With this motivation in mind, many languages were developed to offer a visual interface to query XML documents, such as XQBE [6], XML-GL [14], and GLASS [10]. Those languages were built on the base of graphical representation of XML documents and DTD. XQBE can be considered as a successor to XML-GL, with several new features.

Due to the specificity of XQuery, new constructs have been introduced and some XML-GL constructs have been revised to develop GLASS.

Multimedia documents are annotated and described by a collection of meta-document enriched with spatio-temporal links. Figure 1 [8] gives the meta-model's meta-document.

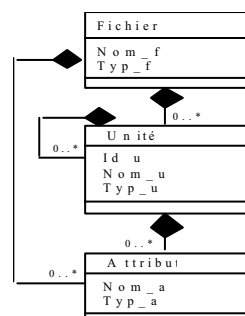


Figure 1: Meta-model's meta-document

A meta-document is composed by different media units. Each unit can be text, video, audio or picture.

Basing on this representation, we claim that querying this type of documents using a traditional language, as XQuery, make complex queries.

Thus, all these query languages, previously quoted, can't support textual and spatio-temporal meta-data such as those proposed in [9].

Our proposal here is to design a graphical query language integrating textual and spatio-temporal meta-data based on the XQuery in order to improve information retrieval. We try to make the graphical language clear and concise in expression and provide a user-friendly query environment. In our approach, we are based on XQuery language because it offers better performances than older languages and supports our requirements.

Multimedia applications impose some specific characteristics with respect to the temporal and spatial composition of objects in the context of the application. Many researches have proposed a set of operators for representing temporal and spatial composition such as the disjunction (sd), the adjacency (sa), before, after, equals, meets... We, also, suggest the extension of the XQuery by means of new operators to facilitate the use of meta-documents.

### 3. Towards a graphical query language

Our graphical query language is designed for users to extract information from semi-structured data. We target both the unskilled users and the expert ones wishing to speed up the construction of their queries. For these reasons, our language must be able to express various queries clearly and concisely without ambiguity, and be simple to draw and easy to read.

In our proposal, we considered the multimedia document as semi-structured. These documents are composed in time and space by different media, i.e. video, sound, picture or text. [9] proposes the annotation of multimedia document based on temporal and spatial relationships. Our proposal consists in a set of spatio-temporal operators to extend the XQuery.

In the following, we introduce the general ideas including the basic concepts and operators in our proposal. After that, we show how to express queries. In addition, we describe our proposal for XQuery extension defining the syntax of new spatial and temporal operators.

#### 3.1. Main contributions

The visual representation of XML documents relies on simplified XML data model, basically reduced on the notion of Elements, Attributes and PCDATA content with containment hierarchies connecting such elements. In order to develop a user friendly interface

easy to understand and manipulate, we propose a set of concepts and operators.

The following sub-sections present these concepts and operators.




##### 3.1.1. Concepts:

As shown in figure 3, a concept can be:

- ✗ Elements nodes (items) are shaped as rectangles labeled with the element's name (or tag name).
- ✗ Attributes nodes are represented as filled black circles. The label on the incoming arc represents the attribute name.
- ✗ PCDATA nodes are depicted as empty circles and denote the textual content of XML elements.
- ✗ Trapezoidal nodes represent newly generated elements (tags) to be included in the query's result.
- ✗ Lozenge nodes are used to specify that the constraint or the information searched can be on items, attributes or PCDATA.
- ✗ The relationship between two XML items is represented by means of a directed arrow from the container to the contained item. Arrows with double line express the ascendant-descendant relationship; i.e. the transitive closure of the relationship.
- ✗ Binding edges are used to specify constraints between outputs and original data (source part) or outputs and meta-document data. The binding edge between the item nodes states that the query result shall contain as many items (elements) as those matched in the source part or meta-document part. They can only connect XML item to other XML item and attribute to other attribute.

##### 3.1.2. Operators:

In order to specify the temporal and spatial relationships, we propose the integration of new operators.

- ✗ Join connections express the association between the document and its meta-document (Cf. Annex). The join connection is visually represented with this  symbol. It links the attribute "name" of meta-document's root and the source document.
- ✗ The operator "OR" is used to represent the binary OR.
- ✗ Spatial relationships are depicted as dashed and labeled arrows  to connect XML items and attributes in the graph. Their label represents a list of spatial operators [c] as follows: sd (the disjunction), sa (the adjacency), so (the overlapping) and si (the inclusion).
- ✗ Temporal relationships are depicted as dashed and labeled arrows  to connect XML items and attributes in the graph. Their label represent a list of spatial operators [c] as follows: "<" (before),

“=” (equals), “tm” (meets), “to” (overlaps), “td” (during), “ts” (starts) and “tf” (finishes).

The orientation of these last two links is a matter of ordering links between meta-data according to temporal and spatial axis, from the top of the meta-document to its end. Both concepts and these operators allow us to draw queries. Thus, as shown in figure 3, our query environment has three different parts.

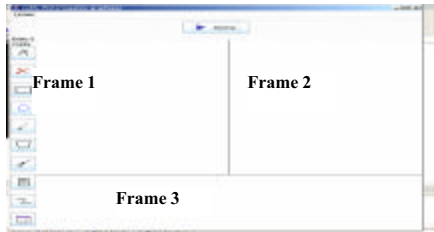


Figure 2: Generic model of a query

The left hand side (Frame 1) is used to specify condition linked to meta-document. The right hand side (Frame 2) is used to specify condition linked to the document source. The third part, which is located under source document and meta-document parts, is used to define the output result. The document and meta-document sides are optional, but the construct side is compulsory. In fact, the user can define the needed output structure based on only the source document or on the meta-document and its original document.

The correspondence between the components of the three parts is expressed by explicit bindings that cross the horizontal line and connect the nodes of the source part or meta-document part to the nodes that will take their place in the output document.

Let us present now some graphical queries and their corresponding translation into XQuery code.

### 3.2. Some query examples

According to several researches, multimedia documents are annotated and designed by a collection of meta-document enriched by spatio-temporal links [9]. The spatial and/or temporal operators enable explaining how meta-data are connected in space and/or in time. Since documents and meta-documents are based on XML structures, it seems that using XML query language is more suitable. Many textual languages were proposed and analyzed by the database community [1,3] such as XQL, XML-QL, Lorel, XQuery, ...

XQuery is the current standard for querying XML data released by W3C [5]. It can extract data from XML documents and construct new XML ones. It allows the integration of new user's defined functions in case of complex queries. So, we choose to use XQuery in our work.

We show in the next sub-sections, two examples of graphical queries and their translation into XQuery. The first example is applied to textual documents, when the second is applied to multimedia ones.

Note, that we take into account the structure of meta-document proposed in [8] and the spatial and/or temporal characteristics.

#### 3.2.1 Querying textual meta-documents

Consider the query q1: "Retrieve the title of French textual documents about Information Retrieve and written by M.Dupont"

To represent the query, we need the `rechvalatt` function, defined as follows:

```
declare function rechvalatt( $e as element(),
    $val as xs:string) as xs:integer
{
  for $a in distinct-values($e)
  if ($a//attribute()=$val) then 1
  else 0
}
```

The function `rechvalatt` verify if the `val` argument is any attribute of the first parameter or no. Using this function, query could be written as follows:

```
<Result>
let $p:= document("meta-doc.xml")
/Text_File[@name]
for $b in document($p)
let $c:=$p[@language= "French "]
where
$b/authors/text () ="M.Dupont" and
[(rechvalatt($c, "retrieve")=1 and
rechvalatt($c, "Information")=1) or
rechvalatt($c, "Information Retrieve")=1]
return { $a/title/text()}
</Result>
```

XQuery is designed for meeting the requirement of skilled programmers, but it's rather cryptical for unskilled users.

Let us translate this query in graphical representation exploiting our concepts and operators. It is depicted in figure 3.

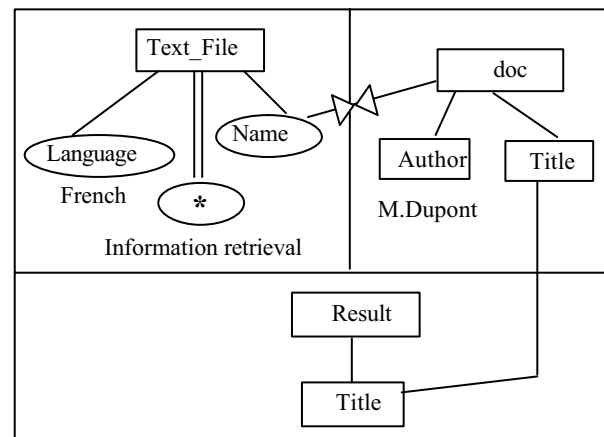


Figure 3 : Example of a graphical querying textual document

First, we extract data from the meta-document part. The figure lists all Text\_File nodes with a "language" attribute whose value is "French" and words "Information Retrieve" can be contained in any attribute of descendant's nodes. The constraint of retrieving all documents having an author whose name is "M.Dupont" is depicted as the author node whose PCDATA content equals "M.Dupont". The binding edge connecting two titles on both sources and constraint sides is a constraint that the titles in the result on the construct side are just the titles that satisfy the condition on the source part and meta-document part. Finally, a join connection between source part and meta-document part is necessary for construction of result query.

### 3.2.2: Querying multimedia documents

Consider the query 2: "retrieve the title, author and speaker of the documents about multimedia when the first text element starts a musical segment". To express the query with XQuery, we need to define two functions of our own.

```
declare function rechvalelt($e as element(),
$val as xs:string) as xs:integer
{ for $a in distinct-values($e)
  if (contains($a/*/text(),$val) then 1
  else 0
}
```

The first function rechvalelt verifies if the val argument is contained in any element of the first parameter or no.

```
declare function linkatt($el1 as element(),
$val as xs:string, $el2 as element(),$el3 as
element() ) as xs:integer
{ for ($c in distinct-
values($el3/ST_LINK/TEMPORAL_LINK)
  if( $el1[@id1]=$c[@id1] and $c[@link]=
$val and $el2[@id2]=$c[@id2]) then 1
  else 0
}
```

The second function linkatt search if the third first arguments were existed on the tag "ST\_LINKS". Using these functions, query could be written as follows:

```
<Result>
let $a:=(document(" meta-doc.xml")//
document#0[@name])
for $b in document ($a)
where
(rechvalatt($a/Text_File, "multimedia" ) =1 or
rechvalelt($a/Video_File,"multimedia")=1 or
rechvalelt($a/audio_File,"multimedia")=1) and
linkat($a/Text_File[1],
"ts",$a/Segment[@nature=music], $a)=1
return
{
<Title> $b/title </Title>
<Authorr> $b/Author</Auteur>
```

```
<Speaker> $a/Segment[@nature=parole]/ Speaker
</Speaker>
}
</Result>
```

Let us translate this query in graphical representation exploiting our concepts and operators. It is depicted in figure 4.

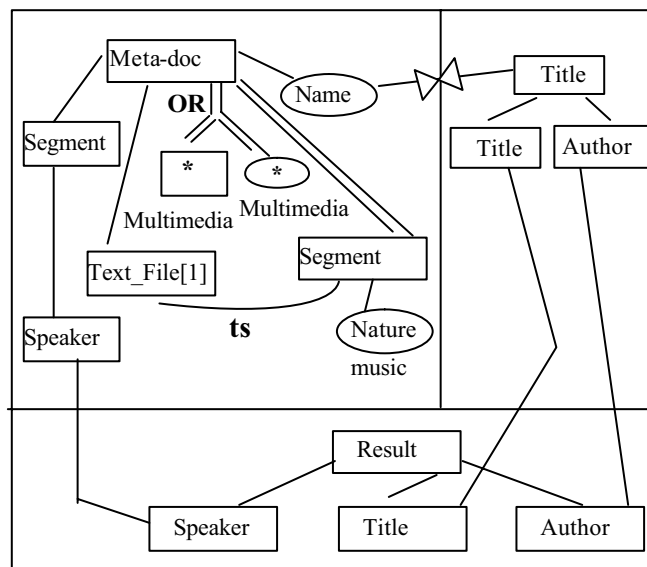


Figure 4: Example of graphical querying multimedia document

First, we extract data from the meta-document part. "Multimedia" can be contained as "key-word" attribute or as Title node for Textual\_File, or as key\_word attribute in video or audio file. The second constraint "the first text element starts a musical segment" is depicted as arrows from Text\_File[1] element to segment element labeled as "ts". The binding edge connecting two Speakers on both meta-document and construct sides is a constraint that Speakers in the result on the construct side are just the Speakers that satisfy the condition on the meta-document and source document part. Note that not only Speaker element is bound in the construct part, but also Title and Author elements, which are connected respectively to Title and Authors items in the source document part. In this way, the result contains element extracted by meta-document and source document part that satisfy conditions.

The following section presents the extension of the XQuery helping the user to make easily graphical queries.

## 4. XQuery extension

Xquery is designed for meeting the requirement of skilled programmer. It can express rich queries over structured data and multimedia documents, it can only express very rudimentary queries over text data. For instance, XQuery allows user to define new functions in order to express their real complex needs. We claim

that the supported definition of functions is sufficient for simple queries but it is woefully inadequate for more complex searches. We thus propose an extension of this language, called XQuery ++, which provides a rich set of spatial and temporal operators.

This section formally defines our proposal for the extension of XQuery which can be generated by our application interface. We describe this extension by means of an EBNF grammar; terminals are enclosed in double apexes and non terminals are bold characters. In the following, we introduce the syntax of the new introduced operators and we give some examples using these operators.

We propose to add new operators called *ComputedDocumentConstructor*, *RepExpr* and *OperatorExpr* expressions. Note that in our extension, we used the common XQuery expression's syntax.

#### 4.1. ComputedDocumentConstructor operator

The *ComputedDocumentConstructor* has the following syntax:

```
ComputedDocumentConstructor ::= <("document"
|"metadocument")">{">ExpeSequence"}"
```

"document" and "metadocument" are terminals specify that the search must be into source file or meta-document file.

Such example for the query1 we can use these operator as follows:

```
<Result>
let $p:= document("meta-doc.xml")
/Text_File[@name])
for $b in document ($p)
let $c:=$p[@language= "French "]
where
$b/authors/text () ="M.Dupont" and
[(rechvalatt($c, "retrieve")=1 and
rechvalatt($c, "Information")=1) or
rechvalatt($c, "Information Retrieve")=1]
return { $a/title/text()}
</Result>
```

The keyword "metadocument" sets the search for meta-document, when, the key word document set search for only simple document.

#### 4.2 RepExpr operator

The *RepExpr* has the following syntax:

```
RepExpr ::= <("Repdoc"| Repmetadoc")
("NmChar")> Expr
```

We have introduced *RepExpr* within *ValueExpr* XQuery expressions. This new operator represents the search into document directory or meta-document directory. The above expression returns a document

file or a meta-document file or a meta-document that meets the query condition.

### 4.3 OperatorExpr operator

The operator *OperatorExpr* is used within a whereClause expression as follows:

```
whereClause ::= "where"(Expr| OperatorExpr)
```

The *OperatorExpr* has the following syntax:

```
OperatorExpr ::= (SpatialOperator |
TemporalOperator)*
```

The " | " operator builds the union of two operators: spatial and temporal. The wildcard "\*" in *OperatorExpr* expression means that we can take zero or more sequences of SpatialOperator | TemporalOperator as input.

#### 4.3.1. SOperator operator

The *SpatialOperator* has the following syntax:

```
SpatialOperator ::= "("Expr <SOperator> Expr ")"
```

```
SOperator ::= DisjunctionOperator |
AdjacencyOperator |
OverlappingOperator|
inclusionOperator
```

```
DisjunctionOperator ::= "sd"
```

```
AdjacencyOperator ::= "sa"
```

```
OverlappingOperator ::= "so"
```

```
inclusionOperator ::= "si"
```

This operator represents the spatial relationship between items into meta-document, it defines the space used for the presentation of an image.

#### 4.3.2. TOperator operator

In the same way, *TemporalOperator* represents the temporal relationship. It has the following syntax:

```
TemporalOperator ::= "(" Expr <TOperator> Expr ")"
```

```
TOperator ::= beforeExpr | equalsExpr | meetsExpr |
overlapsExpr | duringExpr | startsExpr |
finishiesExpr | afterExpr | metbyExpr |
overlapedExpr | containsExpr |
startdbyExpr | finishedbyExpr
```

```
beforeExpr ::= "before" | "<T"
```

```
equalsExpr ::= "equals" | "=T"
```

```
meetsExpr ::= "meets" | "tm"
```

```
overlapsExpr ::= "overlaps" | "to"
```

```
duringExpr ::= "during" | "to"
```

```
startsExpr ::= "starts" | "ts"
```

```

finishiesExpr ::= "finishies" | "tf"
afterExpr     ::= "after" | ">T"
metbyExpr     ::= "metby" | "tmi"
overlappedExpr ::= "overlappedby" | "toi"
containsExpr  ::= "contains" | "tdi"
startedbyExpr ::= "startedby" | "tsi"
finishedbyExpr ::= "finishedby" | "tfi"

```

Both, *SpatialOperator* and *TemporalOperator* return true if a spatial or temporal relationship between two XML items within Meta-document file.

SOperator is a list of spatial elementary operators such as *DisjonctionOperator*, *AdjacencyOperator*, *OverlappingOperator* and *inclusionOperator*.

The second query can use these operators. In this case, we need only to define rechvalatt function.

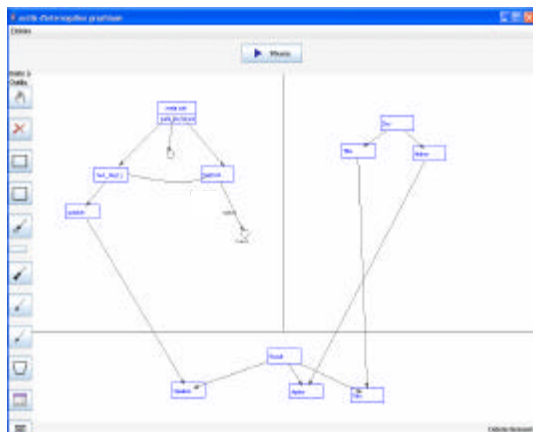
```

<Result>
let $a:=(document("meta-doc.xml")//
document#0[@name])
for $b in document ($a)
where
(rechvalatt($a/Text_File, "multimedia" ) =1 or
rechvalelt($a/Video_File,"multimedia")=1 or
rechvalelt($a/audio_File,"multimedia")=1) and
($a/Text_File[1],
"ts", $a/Segment[@nature=music])
return
{
<Title> $b/title </Title>
<Author> $b/Author</Author>
<Speaker> $a/Segment[@nature=parole]/ Speaker
</Speaker>
}
</Result>

```

## 5. Overview of our query environment

This section, briefly, describes the implementation of our proposal.



Queries are displayed in a window composed of three parts (source, meta-document and construct). Using the tools box, users can draw his queries. Let's take the second query

example (Cf. § 3.2.) to represent it using our interface. Once the users complete their queries they can compile them using XQuery button to find the query result. The tool assists the user draw correctly their queries.

## 6. Conclusions and future Work

In this paper, we have presented a graphical query language for semi-structured data based on the XQuery language. We have, also, proposed the extension of the XQuery language by means of new spatio-temporal operators. This contribution may improve information retrieval and querying XML data offering a user friendly interface.

Our future research works concern first, to enhance the language and to map the proposed graphical expressions into a standard formalism. Then in the second step, we will expand our concepts and operators in order to resolve many query problems. We will, also further, develop tools to help users.

## 7. References

- [1] M.Fernandez, J.Siméon, P.Waleder, S.Cluet, A.Deusch, D.Florescu, A.Levy, D.Maier, J.McHugh, J.Robie, D.Suciu and J.Widom. "Xml Query languages: Experience and examples". 1999
- [2] Irna M.R.Evangelist Flha, Alberto H.FLLaender and Altigran S.daSilva. *Querying Semistructured data by example: the qsby interface*.
- [3] Z.G.Ives and Y.Lu. *XML query languages in practice: an evaluation*. In WAIM'00 2000
- [4] R.Goldman, J.McHugh, J.Widom, "From Semistructured Data to XML: Migrating the Lore Data Model and Query Language", International Workshop on the Web and Databases (WebDB'99), pp.25-30, Philadelphia, June 1999, Pennsylvania, USA.
- [5] D.Chamberlin, D.Florescu, J.Robie, J.Siméon et M.Stéfanescu, "XQuery 1.0: An XML Query Langage", W3C Working Draft, juin 2001.
- [6] M.Zloof. "Query by example: A database language". IBM systems Journal, Vol. 21, N°3, pp.324-343, 1977.
- [7] S.Kepser, "A.proof of the turing-completeness of xslt and xquery", Technical report SFB 441, Eberhard karls Universitat Tubingen, May 2002.
- [8] I.Amous. "Méthodologies de conception d'applications hypermédia-Extension pour la réingénierie ds sites web". Thèse de doctorat, Université Paul sabatier-Toulouse 3, Décembre 2002.
- [9] I.Amous, A.Jedidi, F.Sèdes, "A Contribution to multimedia document modeling and querying". XXI ème congrès d'infosid, pp.407-422, 25-28 Mai 2004, Biarritz/France.
- [10] W.Nei, T.W. Ling. "GLASS: a Graphical Query Langage for Semi-structured Data", Dasfaa2003, pp.363-370, March 26-28, 2003, Kyoto, Japan.
- [11] <http://www.w3.org/TR/xquery/#XQL>
- [12] <http://www.w3.org/TR/1998/NOTE-xml-q1-19980819>
- [13] <http://www.w3.org/TR/xquery/#XPath>
- [14] <http://www.w3.org/TandS/QL/QL98/pp/xml-gl.html>