

A new Graphical Query Language for Retrieving Telecommunication Data

Mourad Ykhlef and Sarra Alqahtani

College of Computer and Information Sciences, King Saud University,
Riyadh, Kingdom of Saudi Arabia
ykhlef@ksu.edu.sa, Sarra.alqahtani@gmail.com

Keywords: XML, G-XML, Graphical query, Grouping, Aggregate functions.

Abstract. XML is fast emerging as a standard for data representation and exchange on the World Wide Web. So, the ability to efficiently query XML data becomes increasingly important. The telecommunications industry has not yet established an official markup language for data management, but is rapidly moving toward XML as a technology of choice as are other industries. In this paper, we propose a new graphical query language for querying and restructuring telecommunication XML data, which we call GQLX. GQLX is developed on the base of G-XML data model. The paper presents the basic capabilities of GQLX through a sequence of examples of increasing complexity. We also discuss some complex queries illustrating the use of grouping and aggregate functions in GQLX.

Introduction

XML data are data which are not necessarily constrained by a schema. XML data are different from relational data in several important points. Relational data have a fixed schema which is stored in a separate table known as catalog, in contrast, XML files contain elements and/or attributes tags to describe data. Relational data model represents missing information by NULL value. XML data, in contrast, can represent missing information simply by the absence of an element (Ykhlef,2007). These variations between XML and relational data bring out the necessity of query languages for extracting information from XML documents in the same way as relational query languages like SQL. The true value of XML is its ability to provide a standard means of exchanging information between applications, business partners, businesses and consumers. It is a critical component in application integration, e-commerce, and telecommunication data management. A graphical query language can potentially be very helpful for these applications. With a graphical language, users do not have to remember the syntax of a textual language, all they need to do is to select options and draw diagrams. However, most graphical query languages are not so popular because graphs with vertices and connections always perform messy and unclear for complex queries. In this paper we describe our effort in building a clear and intuitive graphical language, which we called GQLX, for querying and restructuring XML data.

The following list shows the main objectives of the proposed language GQLX:

1. Supporting information extracting and restructuring.
2. Achieving readability and efficiency in our queries for novice users.
3. Using a standard node-edge graphical tree representation to visualize the hierarchical structure of XML document.
4. Hiding the complex parts of queries from a user to make them user friendly.
5. Supporting query operators like selection, projection join, aggregating and aggregation functions.

The remainder of this paper is organized as follows: Section 2 presents G-XML graph for modeling XML data. Section 3 is devoted to the presentation of GQLX language. We conclude in Section 4.

1. G-XML: XML Data Modeling

An XML document is modeled by an ordered labeled rooted graph called G-XML where (Ykhlef, 2007):

1. Each arc is labeled by an XML element or XML attribute.
2. The attribute arc is dashed while the child arc is directed.

3. Each leaf node is labeled with value.
4. G-XML has a distinguished node called the root.

For example, the XML document of figure1, giving information on bibliographic data, is represented by the G-XML graph of figure 2. Remark that the arc labeled by **number** represents an element tag but the dashed arc labeled by **issue-year** represents an attribute tag. Attributes are alternative ways to represent data. It is important to note that G-XML graphs are not only derived from XML document, but may also generate by GQLX queries (Ykhlef, 2007).

Special case: Note that object identifiers (oids) and references in XML document are just syntax. The G-XML graph representing XML data, in presence of oids and references, can be done straightforwardly.

2. GQLX Language

GQLX is a graphical query language for G-XML data model. A GQLX query can be applied either to a single XML document or to a set of documents. Each query produces a new XML document as the result. GQLX query is a pair of G-XML graphs, displayed side by side and separated by a vertical line. With respect to SQL, the left hand side (LHS) graph visually represents from/where parts, while the right hand side (RHS) graph represents select/create view parts. A GQLX query's parts are described as the following:

1. The LHS part is mainly used to specify the scope of the query, by indicating both the target documents and the target elements inside these documents.
2. The LHS part can be used optionally to specify logical conditions.
3. The RHS part specifies the sub-elements of the extracted elements in the LHS part to be retained in the result.
4. The RHS part can be used optionally to create or construct new elements that should be included in the result document.
5. The condition box is an optional part in a GQLX query. It is placed to write logical conditions for complex queries rather than draw them in the graph.

GQLX language is based on path expressions. Because in our data model, data are associated to leaves, our path expressions may contain data variables as abstractions of the content of leaves. They may also contain path variables which are evaluated to the empty path ϵ or to a path having a length of 1 to n arcs. The graph variables abstract sub-graphs in G-XML graph. Also, our path expressions may contain label variables to preserve labels or tags. We will see examples including these variables in the next sections.

```

<customerInfo>
  <mobile issue-date="2007">
    <company> STC </company>
    <number> 050050505 </number>
    <customer>
      <name> Fahad Al Omar</name>
    </customer>
  </mobile>
  <mobile issue-date="2008">
    <company> Mobily </company>
    <number> 056231233 </number>
    <customer>
      <name> Nasser Al Ali </name>
    </customer>
  </mobile>
  <phone >
    <company> STC </company>
    <number> 014234222</number>
    <customer>
      <name> Fahad Al Omar </name>
    </customer>
    <limit> 500 </limit>
  </phone>
</customerInfo>

```

Fig.1: XML of Telecommunication data

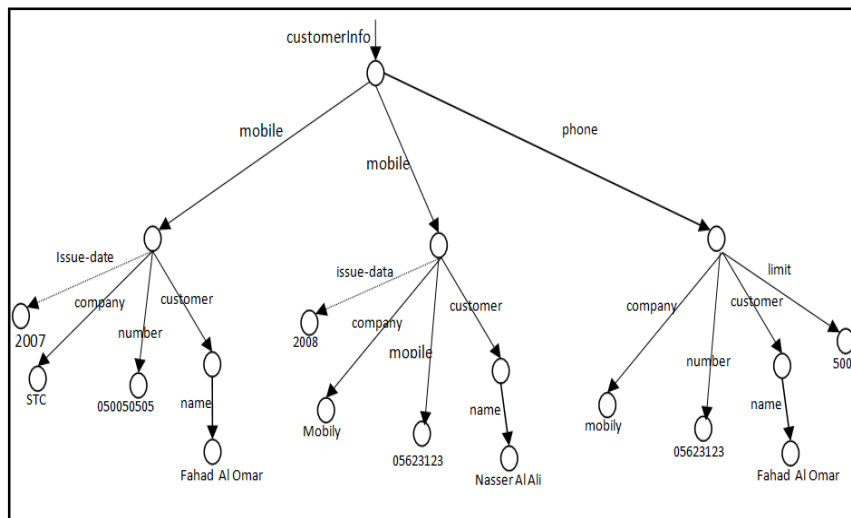


Fig. 2: G-XML of bibliographic data

2.1 Selection and Projection

The simplest form of XML query is the select-project, which extracts some elements of XML document and produces a new document that containing the extracted data.

Query 1 (Simple Projection): Based on G-XML graph in figure 2, the query of figure 3 returns the number of all mobiles.

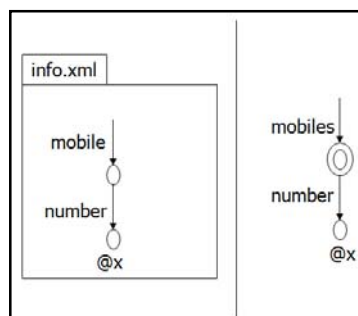


Fig. 3: projection query

In this query, the LHS part specifies the scope of the query to a single target element **mobile** which has a **number** as a sub-element. In general, the LHS of a query may contain several target elements. Also, LHS contains the indication of the target document(s) that should be used as input in order to evaluate the query like "info.xml". In the rest queries of this paper we will ignore the document indication to save the space. The RHS part defines the structure of the result document as a G-XML graph out of the structure of the target elements mentioned in the LHS. When an element mentioned in the LHS part should appear in the result of the query, it must be included also in the RHS graph. The corresponding between LHS and RHS elements is done explicitly by data variables like @x here. The double node that follows the **mobiles** arc in the RHS part (set symbol) used here to environ the collection of **number**. In GQLX, new elements are created directly in RHS if they haven't data variables. This query posing on the G-XML graph of figure 2 returns the following XML data:

```
<mobiles>
  <number>050050505</number>
  <number>056231233</number>
</mobiles>
```

Query 2 (Simple Selection): The below query returns all phones owned by *Fahad Al Omer* and registered in *STC* company. This query displayed in figure 4.

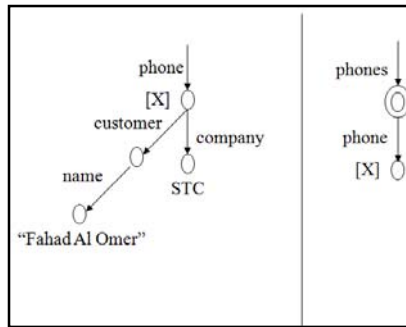


Fig. 4: Selection query

The condition of a query normally involves several sub-elements of the target element(s) in the LHS part. In this query, logical conditions are booleans and attached to specific nodes **name** and **company**. The node that has a graph variable [X] beside it in LHS is used mainly to gather all sub-graphs starting with **phone** node and match logical conditions. In the RHS, we just use the graph variable [X] to include all its sub-elements in the result. The result of this query likes that:

```

<phones>
  <phone>
    <company> STC </company>
    <number> 014234222</number>
    <customer>
      <name> Fahad Al Omar </name>
    </customer>
    <limit> 500 </limit>
  </phone>
</phones>

```

Query 3 (Complex Selection): The query of figure 5 returns customers who have owned phones and mobiles.

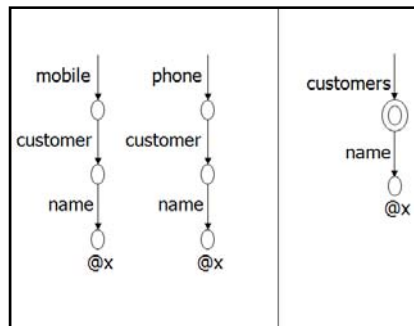


Fig. 5: Complex Selection

This query performs the “inner join” of **mobile** and **phone** based on their customer name. The equality between the values is expressed by using data variable @x. The resulting document is:

```

<customers>
  <name> Fahad Al Omer </name>
</customers>

```

2.2 Join

Now, we will see how to combine and integrate information collected from different portions of documents.

Query 4: Let us consider the G-XML graph of figure 2 and let us consider the following XML data file (yellow.xml) giving addresses and phone numbers of people:

```

<persons>
  <person>
    <name> Fahad Al Omer</name>
    <address> Riyadh</address>
  </person>
</persons>

```

The following query returns the name and the address of each mobile customer (figure 6):

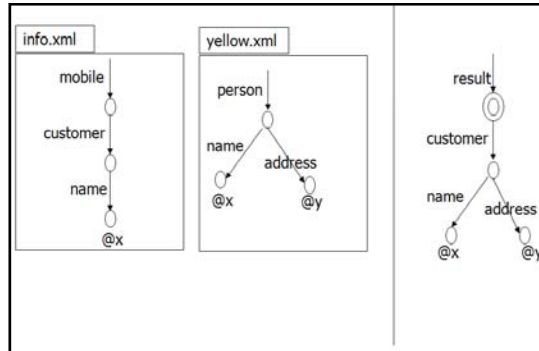


Fig. 6: Join query

The join condition on *name* is expressed in the LHS part by expanding the graph of the **mobile** and the **person** to the inner **name** elements. After that, the joining operation is done explicitly by data variable @x. The result is shown below:

```

<result>
  <customer>
    <name>Fahad Al Omer</name>
    <address>Riyadh</address>
  </customer>
</result>

```

Note that, each **customer** sub-element is extended with the inclusion of the **name** and **address** elements coming from the **person** element retrieved in the LHS.

2.3 Grouping:

One of the most important features of XML query language is its ability to organize the result as groups. We illustrate this feature by the following example.

Query 5 (Simple grouping): The query of figure 7 lists for each customers all of his/her phones and mobiles.

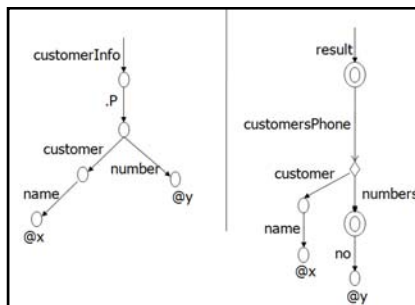


Fig. 7: Grouping query

In GQLX, the grouping operation is directly done by using the grouping symbol (the diamond node). In RHS part, we group each customer with the numbers he or she has owned by using grouping symbol that preceded by **customersPhones** arc. In LHS part, the path variable **P** used to find all occurrences of **customer** and **number** elements available at any level of nesting, so **P** will be evaluated to the empty

path ϵ or to a path having 1 to n arcs. We mainly use **P** here to specify any telecommunication unit regardless of its type is a mobile or a phone. The result of this example is shown below

```

<result>
  <customersPhones>
    <customer><name>Fahad Al Omer</name></customer>
    <numbers>
      <no>050050505</no>
      <no>014234222</no>
    </numbers>
  </customersPhones>
  .....
</result>

```

2.4 Aggregate Functions

It is possible to apply aggregate functions such as sum, count, min, max and so on to groups of homogeneous elements to write conditions on aggregate values in the condition box placed in LHS part of the GQLX query.

Query 6: The query of figure 8 illustrates the use of Grouping with having. It returns customers with phone number above one.

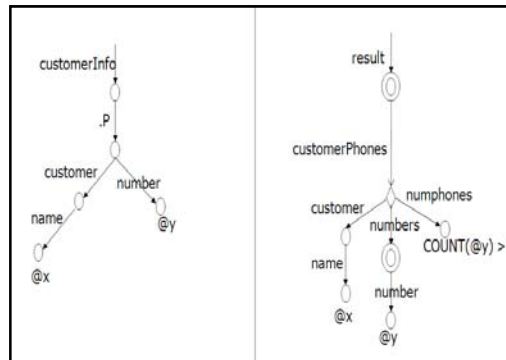


Fig. 8: Group by with having in GQLX

In this query, we mainly use the condition (COUNT(@y)>1) to verify having condition. In RHS, **COUNT(@y)** used to count the number of phones for each customer and store the result in a new sub-element labeled with **numphones**. The result of this query is shown below:

```

<result>
  <customerPhones>
    <customer>
      <name> Fahad Al Omar</name>
    </customer>
    <numbers>
      <number> 050050505 </number>
      <number> 014234222</number>
    </numbers>
    <numphones>2</numphones>
  </customerPhones>
</result>

```

GQLX enables more features like sorting, ordering, attribute querying and restructuring but the detail of these features are omitted for lake of space.

3. Conclusion

In this paper, we have proposed a new graphical query language for telecommunication XML documents. We presented how a user can express queries and restructure new XML documents in a natural way. Based on G-XML data model, the proposed language GQLX uses the tree representation to preserve XML structure. It combines the advantages of graphs and texts which keep the graphical part clear and the textual part easily understood like SQL queries. The future research work on our language is that expanding it to include recursion and nested queries. We are currently designing an environment to exploit the full power of GQLX.

4. References

Braga, D., Campi, A. and Ceri, S., (2003), "XQBE (XQuery By Example): A visual interface to the standard XML query language", ACM Transactions on Database Systems (TODS), Vol.30.

Ceri, S., Comai, S., Damiani, E., Fraternali, P., Paraboschi, S., and Tanca, L., (1999), "XML-GL: A Graphical language for querying and restructuring XML documents", Computer Networks, Vol.31.

Erwig, M., (2003), "Xing: A visual XML Query Language", Journal of Visual Languages and Computing1.

Evangelista Filha, Laender, A.H.F., and Silva, A.S., (2001), "Querying Semistructured Data by Example: The QSByE Interface," Proc. Int'l Workshop Information Integration on the Web.

Wei Ni, Tok Wang Ling, (2003), "GLASS: A Graphical Query Language for Semi-Structured Data," the Eighth International Conference on Database Systems for Advanced Applications.

World Wide Web Consortium. Extensible Stylesheet Language (XSL) Version 1.0, (2001), W3C Recommendation. <http://www.w3.org/TR/xsl/>, viewed 21 April 2008.

World Wide Web Consortium. XQuery 1.0: An XML Query Language W3C Working Draft,(2001) <http://www.w3.org/XML/Query>, viewed 25 April 2008.

Ykhlef, M., (2007), "Recursive SQL-Like query Language for XML," Jakarta: The 9th International conference on Information Integration and web-based Applications and Services.