XQuery As a Spatial Query Language

Xia (Lisa) Li
Senior Software Engineer
Galdos Systems Inc.
#1300-409 Granville St.
Vancouver, BC Canada V6C 1T2

Abstract – *In this paper, we investigate the use of XQuery to retrieve geographic data represented in GML. The proposed approach is aimed at overcoming the limitations of the earlier proposed GML query languages, and is able to handle any GML data. We start with an analysis to show why among various kinds of XML and non-XML query languages we choose XQuery as a fundamental query language for querying GML data. We then describe the peculiarities of GML data and identify the special requirement for querying GML data which are not sufficiently addressed in XQuery language. Finally, we present an approach to extend XQuery to address semantics, navigation mechanism and data types in GML to support query over GML data.*

Keywords: GML, XQuery, Spatial Query, Navigation, Syntax and Semantics.

1 Introduction

The emergence of the World Wide Web brings an unprecedented opportunity to build geographic information systems that can be shared and accessed by various organizations and applications across the Internet. Such sharing systems require a unified format for data interchange among disparate applications. The Geographic Markup Language (GML) was developed exactly for this purpose. GML is XML encoding for the modeling, transport and storage of geographic information [1]. It provides a variety of kinds of objects for describing geography features including geometry, topology and time attributes. Since it was adopted as a standard specification by OGC in 2000, an amount of tools and APIs have been developed to provide access to GML document by various vendors. However, there is no widely accepted and formally defined query language available to allow query posed directly against GML documents yet.

Traditional standard query languages such as SQL cannot be used to query the GML data due to the lack of both spatial query capabilities and XML navigation mechanism. Although GML is XML encoding, the standard XML Query language XQuery [2] is not completely suitable for retrieving data from GML documents. This is because the semantics and the spatial related data types of GML documents require different treatments from generic XML data. In the literature, a couple of attempts have been made to address querying geometry data defined in GML [3], [4]. The fundamental problem of these proposed GML query languages is that they only address GML predefined elements. However, most of GML applications usually have their own element declarations and type definitions. Therefore the previously proposed GML query languages lack the flexibility and generality to process ubiquitous application-defined GML data.

To tackle the above problems in querying GML data, we explore a possible approach to extend XQuery to support query over GML data. The proposed solution is aimed at being able to handle the spatial data and overcoming the limitations of the earlier proposed GML query languages to be able to apply query on any GML data.

Our investigation focuses on three aspects of GIS applications, namely spatial, spatial temporal and spatial networks. We believe that these are the most important applications of GML and the query extension for supporting these data types is sufficient to demonstrate our approach.

2 Why XQuery

GML is XML encoding and the XML model is quite different from the traditional data models such as relational, object-oriented and functional data models. The significant data model differences means the corresponding query languages SQL, OQL or FQL cannot be efficiently used to query over GML data. In addition, XQuery is chosen as the base language in this paper because of the following attributes that are not available in other XML query languages but essential to GML query language,

- XQuery supports XML Schema types. GML is an XML grammar written in XML Schema. It heavily relies on XML schema types. XQuery provides a rich set of operations on these types. A GML query language based on XQuery can make greatly use of these operations to retrieve non-spatial data.
- XQuery emphasizes data-oriented XML [5].A GML document is essentially a data-oriented XML document. Therefore, it is suitable to choose XQuery rather than other XML query language such as XSLT as the base query language for GML data. Furthermore, the data-oriented attribute might lead to a better performance which is important for querying large GML documents that are ubiquitous in GIS applications
- Besides the tree structure, XQuery also provides an additional data structure "sequence" which is also the basic data constructs in GML. The sequence data structure can be used to represent a graph or a path structure which is an essential data structure to model the spatial network represented in GML.
- XQuery is an algebraic language. A number of predefined functions and operations form the basis of the query language. The library of predefined functions and operations can be easily extended with the special operations on GML data types.
- In XQuery, the sequence type syntax allows user to write path expressions that select nodes according to their schema type, type hierarchy or substitution attribute. This is very useful to realize semantic query on GML data [6].
- XQuery is a W3C standard XML query language. The elaborated language definition and strong industry support provide a promising future for building GML query based on XQuery.

3 GML Model, Syntax and Semantics

Although GML is encoded as XML, it has its own underling logical model. The basic constructs of the GML data model are *object* and *property*. An object in GML can be viewed as an entity in E-R model to represent a real world entity, such as a road or a river etc. but with a more generic meaning to represent any meaningful things in an application context such as various kinds of geometry and topology data types. GML properties may be viewed as attributes of objects or relationships between entity objects. The GML properties actually play a role in the same way as the attributes in E-R model. Hence, the GML logical data model is very similar to the traditional E-R model. In fact, it might be more appropriate to say that GML adopts the extended E-R model – an entity relationship model with a notion of inheritance.

A GML object is encoded as an XML element with a type definition that is derived from the GML predefined type *AbstractGMLType*.

The properties of the GML object are represented as sub-elements of the corresponding GML object element.
The property value is represented as the content of the property element

In summary, a GML object is represented in XML syntax as a tree structure as follows,

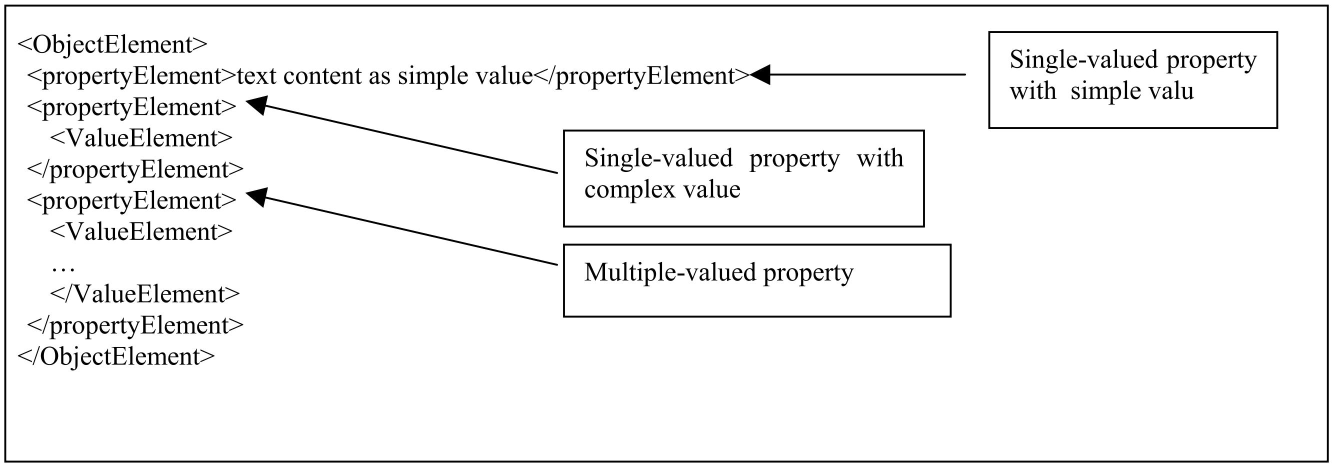
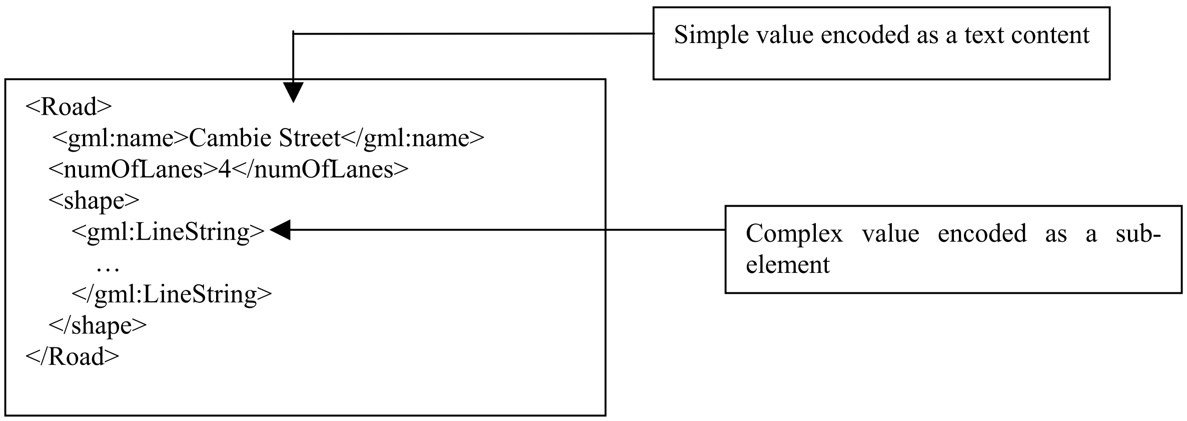

```
<ObjectElement>
  <propertyElement>text content as simple value</propertyElement>
  <propertyElement>
     <ValueElement>
  </propertyElement>
  <propertyElement>
     <ValueElement>
     …
     </ValueElement>
  </propertyElement>
</ObjectElement>
```
Single-valued property with simple valu
Single-valued property with complex value
Multiple-valued property

Figure 1. GML object represented in XML tree.

A geographic feature "Road" can be represented in GML as follows,



```
<Road>
  <gml:name>Cambie Street</gml:name>
  <numOfLanes>4</numOfLanes>
  <shape>
    <gml:LineString>
      …
    </gml:LineString>
  </shape>
</Road>
```

Simple value encoded as a text content
Complex value encoded as a sub-element

Figure 2. Feature 'Road' represented in GML

The basic semantic meaning found in many modelling paradigms (e.g. UML, E-R) [7] implied in GML includes,

- Existence/Type: represented in XML Schema type hierarchy.
- Attribute/Value: represented in the syntax structure of a GML document. The content of a GML property element represents the value of the property.
- Object Relationship: represented in the syntax structure of a GML document. A GML property element associates the parent object element with the child object element.

4 Semantic Query over GML

As described above, the logical model and semantic meaning implied in GML are very common in geo-spatial area. Making use of the semantics to query GML data would be very natural and convenient for end-users and application developers. However, the semantics is not directly reflected in the XQuery data model since generic XML does not bear such semantic meanings. There are several attempts to fill the gap between semantics and syntax. One way is to build semantic model on the top of XQuery data model [8]. Another way is to map the syntax into the semantic and develop functions to facilitate the semantic query [9]. We adopted the latter way to realize semantic query on GML for the reasons a) GML semantics already implies in XML syntax b) we intend to provide semantic view for users rather than embed the semantics into the XML data model.

To facilitate the semantic query over GML, we have developed a function library. This function library can be implemented purely in a schema-aware XQuery processor or implemented with the assistance of programming languages such as Java on a non-schema-aware XQuery processor. Users can use the functions to access GML data based on the common geographic concepts defined in GML core rather than the specific terms defined in various kinds of application domain. In this way, we can achieve a uniform access to GML documents and reduce the significant burden for users to construct queries. A query can be constructed without knowing the concrete tags or structures of a particular GML document. For example, the following query can be posted to any GML document to retrieve all geographic features within a specific region,

```
for $feature in gfn:getFeatures(doc("sample.xml"))
where gfn:within(gfn:getGeometry($feature), gml:Envelop(…))
return $feature
```

Here, the "gfn" represents the prefix of functions defined in our function library.
As you can see, this query is totally constructed with the common vocabularies such as feature, geometry etc and there is no need to list all the concrete feature names in a specific GML dataset.

5 Navigation of GML

In XQuery, path expressions are used to locate nodes in XML documents. The path expressions are defined in XPath 2.0 which is widely used in the XML community as a notation for navigating inside XML documents. The path expression uses a sequence of steps, separated by '/' character, to address nodes within the tree representation of an XML document. Each step in a path expression contains three parts,

- An axis specifies the direction to be traversed.
- A node test, which is either a NameTest or a KindTest, places constraints on the names of kinds of nodes selected by the axis.
- One or more predicates, which are optional, place further restrictions on the sequence of nodes to be selected.

For example, child::chapter/descendant::para selects the para element descendants of the chapter element children of the context node. For the details explanation, see section 3.2 of XQuery spec [2] or XPath spec [11].

The question here is, can the path expression be used to address all nodes in a GML document? Consider the GML fragment below,

```
<Airport>
  <gml:description>Vancouver Airport</gml:description>
  <gml:location xlink:href="#id123"/>
       <facilities>
          <Facility gml:id="f1" …/>
          <Facility gml:id="f2" …/>
         …
          <Facility gml:id="fn" …/>
      </facilities>
</Airport>
```

The <Airport> element is an object node, <gml:description>, <gml:location> and <facilities> are property nodes. The value of the property node <gml:description> is the text node containing the string "Vancouver Airport"; The value of the property node <gml:location> is an object node identified in the xlink:href attribute; The value of the <facilities> property node is a sequence of <Facility> object nodes.

The value of the properties 'facilities' and 'description' are represented directly in the children of the property nodes. Therefore, the value of the property can be given in the path expression "child::gml:description/child::node()" or "child::facilities/child::node()" assume the context node is the node "Airport". However, the value of the property "gml:location" cannot be given by the path expression "child::gml:location/child::node()" since the value is not directly represented as a child element of the property, instead it is indirectly represented through the xlink:href attribute. This example shows that in the case where the value of a GML property isn't represented as the child elements of the property, the child axis cannot be used to address the value node of a property node. In fact, there doesn't exist an axis to locate the value node directly in the XPath expression.

The problem is caused by the fact that XPath is mainly designed to navigate the tree structure of a XML document and the representation of GML objects is not a pure tree. Another reason is the data-oriented characteristic of the GML document determines the most natural way to select nodes is to select object based on the value of their properties, not by a route by which they can be reached [5].

Two possible approaches might be used to address the problem of GML navigation. One is to introduce an external function gfn:value in XPath expression that returns either the child or the node identified by the xlink:href attribute as the property value. Consider, for example, the problem of retrieving Road objects whose geometry property satisfies certain condition from the following document fragment:

```
<Road>
```

```
  <extent xlink:href=”#TopoComplex1”/>
  …
</Road>
<gml:TopoComplex gml:id=”TopoComplex1”>
  <gml:topoPrimitiveMember xlink:href=”#Edge2”/>
  …
</gml:TopoComplex>
..
<gml:Edge gml:id=”Edge2”>
  …
  <gml:curveProperty xlink:href=”#LineString3”/>
</gml:Edge>
…
<gml:LineString gml:id=”LineString3”>
  <gml:coordinates>…</gml:coordinates>
</gml:LineString>
```

The path expression to locate the value of the geometry property would be,
//Road/extent/gfn:value(.)/gml:topoPrimitiveMember/gfn:value(.)/gml:curveProperty/gml :value(.)
The drawback of this approach is that it is difficult to put further constraints on the value of a property.
Another possible approach is to give the child axis a different definition just like the approach used in LMNL document navigation [10]. We could define that the child axis contains either the child nodes or the node specified in the xlink:href attribute. The advantage of this approach is that it does not require any changes to XPath syntax and from the user’s point of view there is no difference to address a value represented as child nodes or a value represented by reference. However, this might bring confusion to the concept of the child axis. Hence we would like to propose the alternative approach which is to add a new axis *value* in XPath that is intended to apply on property nodes to locate the value nodes of the property nodes.

The advantage of this approach over using gfn:value function is that all NameTest and predicates available to other axes can also be used to make further restrictions on the value nodes selected by the *value* axis.

6 Operations on GML Data Types

XQuery defines a rich set of operations and functions in the specification XQuery 1.0 and XPath 2.0 Functions and Operations. However, these operations are mostly defined on the XML schema built-in types. There are no operations defined in XQuery for the spatial related types defined in GML. In order to allow the user address the spatial data in the query language, it is necessary to extend the existing set of XQuery operations and functions with the operations on the spatial related types defined in GML.

A GML application can have its own spatial related type definitions. These types might not be the predefined spatial related types defined in GML core schemas. An operation must be designed to be able to accommodate any application data rather than only applicable to the GML predefined data types. We achieve this by making use of the XQuery sequence type syntax both in the parameters and the return values of the operation definitions. In XQuery, a sequence type construct can be used to represent a class of items that has some common attributes. For example, element(*, gml:PointType) refers to any element with a type definition being gml:PointType or derived from gml:PointType. An operation that takes a sequence type construct such as element(*, gml:PointType) can be applied to any point element in an application document regardless of the element name and the type name as long as the element is declared with a type definition derived from gml:PointType. Since any spatial data type defined in a GML application must be derived from a GML predefined type, an operation with the sequence type constructs based on GML predefined types must be able to be applied to any corresponding application-defined data types in GML documents.
There are three categories of spatial related types in GML, which are spatial, spatial temporal and spatial network data types. Each of them represents an application area in GIS.

- The spatial types are used to describe the geometry locations of geographic data.
- The spatial temporal types describe the data with geometry aspects changing over time.
- The spatial network types describe the connectivity of spatial data.

Below is the typical example operations defined on these three categories of spatial related types in GML,

Table 1. Operations on Spatial, temporal and spatial network data types

Category	Operation definition	Brief Description
Functions test for spatial relationships	*gfn:equal*($param1 as element(*, gml:AbstractGeometryType), $param2 as element(*, gml:AbstractGeometryType)) as xs:Boolean *gfn:disjoint, gfn:touches, gfn:within, gfn:overlaps, gfn:crosses, gfn:intersects, gfn:contains*	Tests the topology relationship between two geometry objects.
Temporal predicate	*gfn:before*($param1 as element(*,gml:TimePrimitiveType), $param2 as element(*,gml:TimePrimitiveType)) as xs:Boolean *gfn:after, gfn:equal, gfn:contains, gfn:meets, gfn:overlaps, gfn:starts, gfn:finishes*	Tests the temporal relationship between two time objects.
Predicates on a GML Graph	*gfn:reachable*($param1 as element(*, gml:EdgeType)+, $param2 as element(*, gml:NodeType), $param3 as element(*, gml:NodeType)) as xs:boolean	Determine whether a node can be reachable from another node in the given graph.

Given a dataset represented in GML as follows,

```
<app :Country>
  <gml :name>…</gml :name>
  <app :population>…</app :population>
  <app :extent>
    <app :MyPolygon>
        …
    </app :MyPolygon>
  </app :extent>
</app :Country>
<app :River>
  <gml :name>…</gml :name>
  <app :curveProperty>
      <app :MyLineString>
        …
      </app :MyLineString>
  </app :curveProperty>
</app :Rive>
```

Here the element 'Country' and 'River' are defined as GML features, the application-defined geometry elements 'MyPolygon ' and 'MyLineString' are defined as being subtypes of GML predefined geometry type 'gml :PolygonType' and 'gml :LineStringType' respectively.
The query to find all countries that pass through the rivers can be formulated as follows using the operations given above,

```
for $river in doc('sample.xml')//app :River
for $country in doc('sample.xml)//app :Country
where gfn :crosses(gfn :value($country/app :extent), gfn :value($river/app :curveProperty))
return $coutry/gml :name, $river/gml :name
```

Here, the gfn :value function returns the value of the specified property which are the element <app :MyPolygon> and the element <app :MyLienstring> in this case. The function gfn :cross takes these two geometry elements and returns a boolean value that indicates whether they are crossed to each other. The query finally returns all names of countries that pass through the rivers.

7 Conclusions

In this paper, we have presented an extensive approach to extend XQuery to support query over GML data. In particular, the basic semantics of the GML data model is identified and an approach to exploit semantics to achieve uniform access to GML data is developed. The peculiarity of navigation of GML document is investigated and two solutions are introduced to make it amenable to XQuery navigation mechanism, XPath processing model. The XQuery type system is augmented with the data types defined in GML. A set of operators and functions on GML data types that cover the most typical queries over spatial, spatial temporal and spatial network are developed.
In contrast to the related work such as [3], [4] where the proposed query language can only be applied to the data with predefined GML types, our major contribution lies in that the extended query language developed by our approach is applicable to any GML data. This is very important in that most existing GML applications always have their own data types which are derived from GML predefined types but not the GML predefined types. That means a query language only applicable to the data with GML predefined types would have limited usage in GML applications. Our other contributions are that we proposed a uniform access mechanism that provides easy query construction to users and enables to query/combine heterogeneous and autonomous GML resources. We developed a suitable set of operations on GML spatial temporal and spatial network data types, and also addressed the issue of navigation of GML documents. To our knowledge, there is no precedence work on these issues.

8 References

[1] S. Cox et al, Open GIS Geography Markup Language (GML) Implementation Specification. Open GIS recommendation paper. 2004.

[2] W3C, (2005, April). [Online]. XQuery 1.0: An XML Query Language. Available: http://www.w3.org/TR/2005/WD-xquery-20050404/

[3] J, E, Ciorcoles and P, Gonzalez. A specification of a spatial query language over GML. The ACM Digital Library. [online] Available: http://portal.acm.org/citation.cfm?id=512186

[4] R, R, Vatsavai, GML-QL: A spatial query language specification for GML. Available: http://www.cobblestoneconcepts.com/ucgis2summer2002/vatsavai/vatsavai.htm

[5] H, Katz et al, *XQuery from the Experts: A Guide to the W3C XML Query Language*. Addison-Wesley Professional, 2003.

[6] Xia (Lisa) Li, "Ontology-based semantic access to GML documents," presented at the GML Days 2003, Vancouver, BC, 2003.

[7] B, Matthews, "Using RDF to derive schema mappings". Available: http://www.xmluk.org/images/content/RAL_06_2004/Brian%20Matthews%20XMLandRDFJune2004.ppt

[8] P, Patel-Schneider and J, Simeon "The Yin/Yang web: XML syntax and RDF Semantics". Available: http://www2002.org/CDROM/refereed/231/

[9] J, Robie, "The syntax web". Available: http://www.idealliance.org/papers/xml2001/papers/html/03-01-04.html

[10] J, Tennison and W, Piez 'Layered markup and Annotation Language(LMNL) Extreme Markup Language 2002

[11] W3C, (2005, April). [Online]. XML Path Language (XPath) 2.0. Available: http://www.w3.org/TR/xpath-20