GXQuery: A GML Spatial Data Query Language

Jianhua Chen
Institute of Geo-Spatial Information S&T
Univ. of Electronic Science and Technology of China
Chengdu, China
College of Information Engineering
Chengdu Univ. of Technology
Chengdu, China
e-mail: chjh3@163.com

Binbin He
Institute of Geo-Spatial Information S&T
Univ. of Electronic Science and Technology of China
Chengdu, China
e-mail: binbinhe@uestc.edu.cn

Weihong Wang
College of Geology Science
Chengdu Univ. of Technology
Chengdu, China
e-mail: Wangwh78@163.com

***Abstract*—GML spatial data query language is a very important part of native GML spatial database system, it is an indispensable component to GML documents for spatial data query as well. As the data model has significant difference between XML/GML and relational database, it's not suitable using SQL or extending SQL to query GML spatial data. XQuery is W3C recommendation for XML data query, directly using XQuery to query GML spatial data, only the GML non-spatial data can be queried, the GML spatial data can not be queried because of GML's spatial characteristics. Integrally referring previous research achievements, this paper further studied GML spatial data query language based on XQuery and name it GXQuery in order to reflect the integration of GML and XQuery, problem of GML spatial data query, reasons for extending XQuery to support GML spatial data query, features of GML spatial data query language, content of XQuery spatial extension, architecture of GXQuery, implementation methods of GXQuery were detially studied and discussed, and typical query examples of GXQuery were also given.**

Keywords-GML Spatial Data; Query; XQuery; Spatial Extension

I. XML QUERY LANGUAGE: XQUERY

XQuery[1,2], based on Quilt and XPath, is designed to be a common XML query language by the World Wide Web Consortium(W3C) for normalizing XML query languages, and it became the W3C recommend standards in January 2007. XQuery to XML, just as SQL to relational database. XQuery is designed to query XML data, including XML data stored in database. XQuery is entirely based on XML, its element, attribute and variable must be valid XML name.

XQuery1.0 has the same data model with XPath2.0, and supports the same functions and operators. XQuery uses functions to get data from XML documents, and uses path expressions to navigate in XML documents. XQuery is a functional language, a query is composed by expression. XQuery usually has 7 expressions: FLWOR(FOR，LET，WHERE，ORDER BY，RETURN) expressions, path expressions, element constructors, conditional expressions, quantified expressions, functions and operators, data type test and modification. And FLWOR is key expression, FLWOR to XQuery, just as SELECT…FROM…WHERE to SQL.

XQuery includes over 100 built-in functions which for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more. XQuery 1.0 and XPath 2.0 share the same functions library, and people can also define their own functions in XQuery. Its syntax as follows:

```
declare function prefix: function_name($parameter as
    datatype, ...) as datatype
{
    (: ...code... :)
};
```

Use-defined functions can be called the same way with built-in functions.

A XQuery usually is composed by namespace and mode declaration, function defination, and query expression(figure 1).

XQuery supports use-defined function, which provides the possibility for GML(Geography Markup Language)[3,4,5] spatial data query by adding spatial extension, and the extension is adding spatial data types and spatial operators.

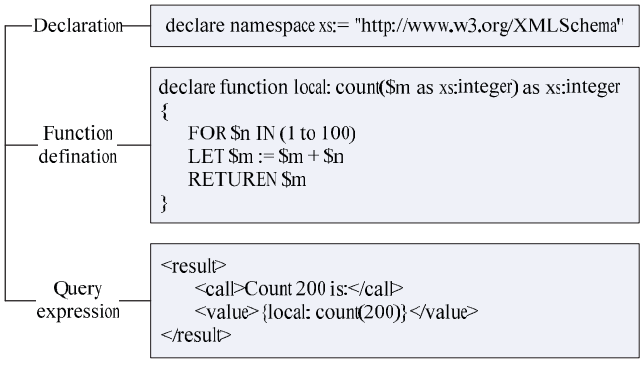

Figure 1. Example of XQuery composition.

978-1-4244-5265-1/10/$26.00 ©2010 IEEE

II. PROBLEM OF GML SPATIAL DATA QUERY

As the data model has significant difference between XML/GML and relational database, it's not suitable using SQL or extending SQL to query GML spatial data. Reasons are as follows:

1) XML/GML organizes data with irregular, hierarchical tree structure, while relational database organizes data with regular, planar table structure. In XML/GML, the same tags of the same hierarchy can contain different numbers of sub-elements, even the sub-element names can be different. In relational database, the records of a relation have fixed length, and their field types must be identical. If XML data embedding depth is unknown, its attributes, elements and values can be queried normally, while relational database can't do this normally if the relations are fragmentary because the relations' some structures and their foreign keys can't be sure at the query moment.

2) The query result is isomerous to XML/GML and isomorphic to relational database. The XML/GML query returns a document fragment, the same tags of the same hierarchy can contain sub-elements with different names and value types. While every record of the result dataset returned by relational database must have the same fields.

3) XML/GML data maybe sparse, while data in relational databse must be dense. The same tags of the same hierarchy in XML can contain different numbers of sub-elements. But, the fields of every record of a relation muse equal and exist, even their values may be 0, nil or NULL.

4) XML/GML data elements are ordered back and forth, while data in relational database are not. When processing the XML/GML data, it usually is constructed to be a tree, order is exist among them. To records of a relation, order is not exist.

Besides, directly using XQuery to query GML spatial data, only the non-spatial data of GML can be queried, the spatial data of GML can not be queried because of GML's spatial characteristics.

III. GXQUERY: GML QUERY LANGUAGE EXTENDED FROM XQUERY

GML spatial data query language is a key part of native GML spatial database system, it is also a indispensable part for data querying to GML documents. To its research progress, Ranga Raju Vatsavai[6] attempted to propose GML-QL query language extended from XQuery. François-Marie Colonna and Omar Boucelma[7,8] proposed a GML spatial data query language prototype: GQuery based on XQuery for GML document data query. Guan Jihong[9] proposed a GML document data query language: GQL. Lan Xiaoji[10,11] proposed a GML spatial data query language: GMLXQL extended from XQuery by adding spatial extension. Overally, the research work of GML spatial data query language is still at the initial stage. Integrally referring previous research achievements, this paper will further study GML spatial data query language based on XQuery, and name it GXQuery in order to reflect the integration of GML and XQuery.

A. Reasons for Extending XQuery to Support GML Spatial Data Query

Adding spatial extensions to XQuery to implement GML spatial data query, not inventing a entire new GML spatail data query language, reasons are as follows.

1) XQuery is W3C recommendation for XML data query. It benefits for spread, use and evolution of GML query language based on recognized standard.

2) XQuery is a powerful, simple, functional language. The design goal of XQuery is function powerful and easy to use. Based on XQuery, GML query language must inherit its advantages, and is developed and applied quickly.

3) XQuery owns extensibility. By using self-defined functions, XQuery can be extended. This feature provides strong support for GML spatial data query by adding spatial extensions to XQuery.

4) GML spatail data is based on XML, GML data includes both spatail data and non-spatial data, by adding spatial extensions to XQuery and keeping its syntax specifications, the integral query of GML spatial data and non-spatial data can be implemented.

B. Features of GML Spatial Data Query Language

A kind of query language usually has some typical features to indicate its own specialty. Max J. Egenhofer[12] proposed 11 requirements for spatial data query language by analysing traditional query language and spatial data. Dallan Quass[13] proposed 10 features for XML query language by combining characteristics of SQL syntax and XML data. Lan Xiaoji[10,11] summarized 10 features of GML spatial data query language, according to requirements of spatial data query language, features of XML query language and characteristics of GML spatial data. These are as follows.

1) It is a declarative language, user only needs to know what to do, not to know how to do.

2) It supports XQuery syntax, its query expressions entirely follow XQuery syntax , which can be conveniently read or understood by user or software.

3) It supports data types specified in XML Schema specifications.

4) Its query result can be GML document or part of GML document.

5) It can not only query GML documents but also GML database.

6) It supports GML data model and data types defined in GML specifications, such as geometry, topology, time.

7) It support spatial operation functions, general spatial relation computation and spatial analysis computation.

8) The query operation can be done in GUI environment. It makes query operation easy and convenient, which makes user free from the bald text query expression. It not only can implement all function of text type query language, but also can describe and implement advanced query.

9) It support various update operations, including spatial

and non-spatial data update. The GML query language supports insertion, deletion and modification, besides query. The XQuery still does not suport update, which needs to be extended for data update.

10) It supports visualization of query result. The result can be shown with graphics, map symbols and annotations. GML query results usually are subset of GML documents or part spatial objects of GML database, which needs to be displayed graphically to user.

C. Content of XQuery Spatial Extension

Extending XQuery to support GML spatial data query mainly concerns spatial data types and spatial operation functions. Adding spatial data types and spatial operation functions to XQuery can achieve GML spatial data query.

1) Spatial data types

OGC Simple Features Specification for SQL[14](abbr. SFS) defined spatial data types and spatial operation functions. The spatial data types include: Point, LineString, LinerRing, Polygon, MultiPoint, MultiLineString, MultiLinerRing and MultiPolygon. These data types are consistent with data types defined in GML 2.X, which mainly aim at two-dimensional objects and linear geometric entities. While GML 3.X also supports three-dimensional objects and non-linear geometric entities. Based on SFS, adding three-dimensional data types. Using SFS's two-dimensional data types and the newly added three-dimensional data types as the spatial extension data types to XQuery.

2) Spatial operation functions

The spatial operation functions defined in SFS include spatial basic methods, spatial relation methods and spatial analysis methods. Using these functions as spatial operation function library of XQuery extension. As these functions mainly aim at two-dimensional objects and linear geometric entities, functions aiming at three-dimensional objects and non-linear geometric entities should be created and added.

D. Architecture of GXQuery

The Architecture of GXQuery is shown with figure 2. Its contents are as follows.

1) XQuery engine

GXQuery is based on XQuery, and XQuery is the key part of GXQuery. XQuery engine is based on XQuery language standards, which mainly performs parsing of XML query sentences, optimizing of XML query, executing of XML query, and result constructing of XML query. Adding GML parser and spatial extension, XQuery engine can perform parsing, optimizing, executing and result constructing of GML query.

2) GML parser

XQuery engine can only parse and construct the result of query sentences of XML and GML non-spatial data. If parsing and constructing the result of query of GML spatial data, GML general parsing module must be added to XQuery engine, that is GML parser. Then GML spatial and non-spatial data can be queried integrally.

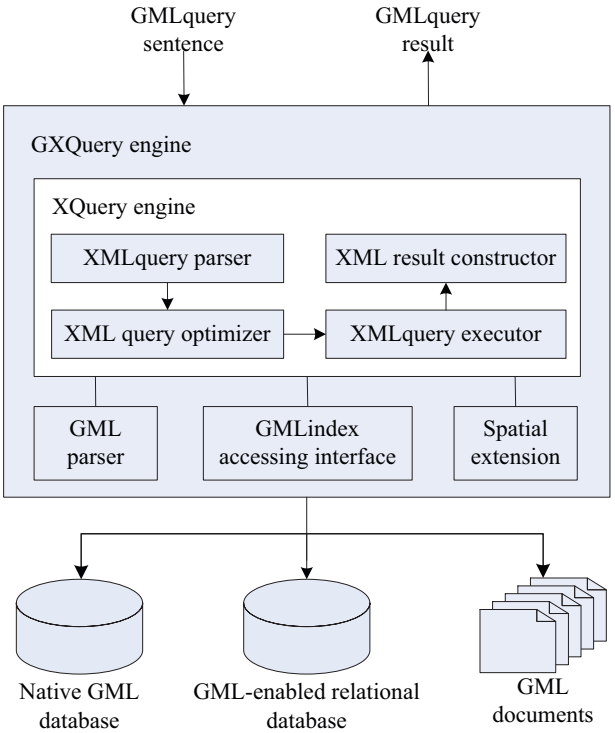

Figure 2. The architecture of GXQuery.(Partially derived from Xiaoji Lan[10].)

3) Spatial extension

Adding spatial extension to XQuery is the key feature for implementing GXQuery language. XQuery originally does not support GML spatial data, adding spatial extension to XQuery engine, and combining with GML parser, XQuery engine can then perform query and processing to GML spatial data. Spatial extension module is mainly composed of spatial data types and spatial operation functions, which conform to XQuery standards.

4) GML index accessing interface

GML index is a very important part of native GML spatial database system, it can significantely upgrade query and processing speeds of GML query engine. Adding GML index accessing interface module to XQuery engine, the GXQuery engine can access GML index management module in native GML database or GML-enabled relational database, or GML index files related to GML spatial data files, before GML data query. GXQuery engine gets the intended GML spatial data via GML index data information, then constructs the GML query result.

GML index is very important to GXQuery engine, while GXQuery engine may not need to access GML index module but just access GML data sets when performing GML data query. Indeed, the query performance is quite different.

5) GML storage source

GXQuery language is designed independent from GML storage sources. GML spatial data can be stored in native GML spatial database, or GML-enabled relational database, even GML documents in file system. GXQuery engine provides different access interfaces to different GML storage sources, and it is abstracted to unified model above the interfaces, which is convenient for independent evolution of GXQuery language.

E. Implementation Methods of GXQuery

Figure 2 shows that GXQuery engine is composed of XQuery engine, GML parser, spatial extension module and GML index accessing interface module. For GXQuery engine, XQEngine component can be used as XQuery engine, JTS(Java Topology Suite) API component can be used as spatial extension, and combined with XQEngine, GML parser and GML index accessing interface module are developed by authors and combined with XQEngine.

XQEngine[15] is a kind of open source implementation of XQuery language, which intends to query XML documents. XQEngine is a compact and embedded component implemented by Java, it can be embedded in applications in many ways. The latest version is 0.69.

JTS[16] is an open source API component intending to operate and process 2D linear geometry spatial objects. It provides lots of spatial predicates and functions, which conform to OGC's SFS. The latest version is 1.9. For 3D objects and non-linear geometry spatial objects, JTS should be extended.

GXQuery language usually has three kinds of queries.

1) GML non-spatial data query

GML non-spatial data query can be executed as general XML data query by XQEngine which is a part of GXQuery engine.

2) GML spatial data query

As GML spatial data query includes simple spatial data query, spatial data basic character query(such as 'Dimension'), spatial relation query(such as 'Equals'), spatial analysis query(such as 'Buffer'), the query process goes as follows.

First, after GML spatial query sentence is processed by GML parser and XQEngine, combining with GML index accessing interface module(of course, this is not indeed necessary), the intend GML spatial data is returned. If the query request is simple spatial data query, the returned spatial data is straight sent to upper caller as result, else do step two.

Second, using GML parser to transform the former returned spatial data(in step one) into 'Well Known Text Format'(abbr. WKT) string, for an instance, transforming GML format polygon geometry object into text string : Polygon((30 60,30 80,50 90,40 80,30 60)). Then, the WKT string is read by JTS component API, and the corresponding objects are created, the spatial operation functions in JTS then operate those objects according to specified types and produce result. To this result, if its type is integer, or logical type, or string type, it is sent to XQEngine directly. If its type is geometry object type, it should be transformed into GML format data by JTS functions and sent to XQEngine. Finally, combining with GML parser, XQEngine construct the returned value as GML format and send to upper querier.

3) GML spatial and non-spatial data mixed query

To GML spatial and non-spatial data mixed query, the query sentences should be decomposed into spatial query part and non-spatial query part. As spatial query usually is more time-consuming, to mixed query, first non-spatial data query is performed and primary result is produced, then spatial query is performed on primary result and the finally result is produced, the finally result is constructed as GML format and sent to upper querier. By doing so, the mixed query performance can be remarkably upgraded.

In FLWOR query expression of GXQuery language, the functions of basic operation, spatila relation calculation, spatial analysis and topology processing can be put into every sub-sentence of the five: FOR, LET, WHERE, ORDER BY, RETURN. And spatial operation functions can be embeded.

F. Query Examples of GXQuery

Several typical query examples of GXQuery language are given as follows.

1) The query sentences return 'Building' features conforming to specified name and specified ID from 'CDUTCampus.gml' data sets. (Showing non-spatial data query according to multiple conditional statements.)

```
FOR $var IN doc("CDUTCampus.gml")//Building
WHERE $var/gml:name/node() = "Geology Building"
  AND $var/@fid = "b0004"
RETURN $var
```

2) The query sentences return geometry object's dimension of 'Road' feature conforming to specified name from 'CDUTCampus.gml' data sets. (Spatial and non-spatial data mixed query, showing the usage of 'geo: Dimension' function.)

```
FOR $var IN doc("CDUTCampus.gml")//Road
WHERE $var/gml:name/node() = "Yanhu Road"
RETURN geo:Dimension($var)
```

3) The query sentences check whether the two geometry objects corresponding to two 'Building' features from 'CDUTCampus.gml' data sets are spatially equal. (Spatial and non-spatial data mixed query, showing the usage of 'geo: Equals' function.)

```
FOR $var1 IN doc("CDUTCampus.gml")//Building,
  $var2 IN doc("CDUTCampus.gml")//Building
WHERE $var1/gml:name/node()="CDUT Gymnasium"
  AND $var2/@fid = "b0028"
RETURN geo:Equals($var1, $var2)
```

4) The query sentences return 'Road' features which classification is 'motorway' and intersect with specified geometry object. (Spatial and non-spatial data mixed query, showing the usage of 'geo: Intersects' function.)

```
FOR $var IN doc("CDUTCampus.gml")//Road
```

WHERE $var/classification = “motorway” AND
geo:Intersects($var, “30.600 104.150 30.700 104.250”)
RETURN $var

5) The query sentences check whether ‘Palestra’ feature spatially contains ‘Gymnasium’ feature, both coming from ‘CDUTCampus.gml’ data sets. (Spatial and non-spatial data mixed query, showing the usage of ‘geo: Contains’ and ‘geo:Envelope’ functions.)

FOR $var1 IN doc(“CDUTCampus.gml”)//Building,
$var2 IN doc(“CDUTCampus.gml”)//Building
WHERE $var1/gml:name/node() = “CDUT Palestra”
AND $var2/gml:name/node() = “CDUT Gymnasium”
RETURN geo:Contains(geo:Envelope($var1), $var2)

6) The query sentences firstly perform 200 meters buffering operation to geometry objects corresponding to ‘Geology Building’ features, then return common geometry objects which are created via intersection operation between new buffering geometry objects and geometry object corresponding to ‘Yanhu Road’ feature. (Spatial and non-spatial data mixed query, showing the usage of ‘geo: Intersection’ and ‘geo: Buffer’ functions.)

FOR $var1 IN doc(“CDUTCampus.gml”)//Building,
$var2 IN doc(“CDUTCampus.gml”)//Road
WHERE $var1/gml:name/node() = “Geology Building”
AND $var2/gml:name/node() = “Yanhu Road”
RETURN geo:Intersection(geo:Buffer($var1,200),$var2)

IV. CONCLUSIONS

GML spatial data query language is very important to GML spatial data query. Integrally referring previous research achievements, this paper further studied GML spatial data query language based on XQuery, problem of GML spatial data query, reasons for extending XQuery to support GML spatial data query, features of GML spatial data query language, content of XQuery spatial extension, architecture of GXQuery, implementation methods of GXQuery, and query examples of GXQuery were detially studied and discussed.

REFERENCES

[1] W3C. XQuery 1.0: An XML Query Language[EB/OL]. http://www.w3.org/TR/ xquery/, 2007.

[2] W3Schools. XQuery Tutorial[EB/OL]. http://www.w3schools.com/ xquery/default.asp, 2007.

[3] OGC. OpenGIS Geography Markup Language (GML) Encoding Standard(Version 3.2.1) [EB/OL]. http://www.opengeospatial.org/ standards/gml, 2007.

[4] OGC. OpenGIS Geography Markup Language (GML) Implementation Specification (Version 3.1.1)[EB/OL]. http://www. opengeospatial.org/standards/gml, 2004.

[5] OGC. OpenGIS Geography Markup Language (GML) Implementation Specification (Version 2.1.1)[EB/OL]. http://www. opengeospatial.org/standards/gml, 2002.

[6] Ranga Raju Vatsavai. GML-QL: A Spatial Query Language Specification for GML[EB/OL]. http://www.cobblestoneconcepts. com/ucgis2summer2002/vatsavai/vatsavai.htm, 2002.

[7] François-Marie Colonna, Omar Boucelma. Querying GML Data[C]. In: Proceedings of CoPSTIC'03, Rabat, 2003, pp.1-5.

[8] Omar Boucelma, François-Marie Colonna. GQuery: A Query Language for GML[C]. In: 24th Urban Data Management Symposium, Elfriede Fendel, Massimo Rumor, UDMS 2004, Chioggia, Italy, 2004, pp.23-32.

[9] GUAN Jihong, ZHU Fubao, ZHOU Jiaogen, et al. GQL: Extending XQuery to Query GML Documents[J]. Geo-spatial Information Science, 2006, 9(2):118-126.

[10] Lan Xiaoji, Lu Guonian, Liu De'er, et al. XQuery-based GML Query Language[J]. Science of Surveying and Mapping, China, 2005, 30(6): 99-102.

[11] Lan Xiaoji, Xiao Huihui, Duan Yanming. Data Query System of GML Spatial Database Based on Expanded NXD[j]. Journal of Geodesy and Geodynamics, China, 2008, 28(1): 85-91.

[12] Max J. Egenhofer. Spatial SQL: A Query and Presentation Language[J]. IEEE Transactions on Knowledge and Data Engineering, 1994, 6(1):86-95.

[13] Dallan Quass. Ten Features Necessary for an XML Query Language[EB/OL]. http://www.w3.org/TandS/QL/QL98/pp/quass. html, 1998.

[14] OGC. OpenGIS Simple Features Specification For SQL(Revision 1.1)[EB/OL]. http://www. opengeospatial.org/standards/sfs, 1999.

[15] Howard Katz. XQEngine[EB/OL]. http://sourceforge.net/projects/ xqengine, 2005.

[16] Vivid Solutions Inc. JTS Topology Suite[EB/OL]. http://sourceforge. net/projects/jts-topo- suite/, 2008.