I.J. Education and Management Engineering **2012, 2, 13-20**
Published Online February 2012 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2012.02.03

Available online at http://www.mecs-press.net/ijeme

Study of Index Mechanism for GML Data

Yajuan Yao, Lijuan Shi*

College of Science, Huazhong Agricultural University, Wuhan, China

Abstract

GML is the encoding specification of geospatial data, currently more and more geospatial data are represented in GML documents. In order to query GML data efficiently, appropriate index structure should be designed. As the application of XML in geospatial information field, the management of GML data can borrow ideas from the management technologies of XML data. Two types of existing XML path index technologies are analyzed, and a method to build path index for the GML document is proposed based on its storage schema. The path index records the sequences and levels of elements in the GML document. Tuples stored in tables can be quickly located for a query with the path index, and the structure information recorded in the path index can help to reconstruct the GML document.

Index Terms: GML; spatial database; path index; spatial index

1. Introduction

GML (Geography Markup Language) is the encoding specification of geospatial data published by the OGC (Open GIS Consortium) to resolve the issue about geospatial data sharing. In fact, GML is the application in geospatial information field of XML (Extensible Markup Language).The self-descriptive and extendibility of XML make it the mainstream data format in many fields of data exchange and storage.

From the appearance of GML to now, with the development of related technologies and the improvement of the specifications, GML has been widely accepted as the criterion for spatial data format, and more and more spatial data are represented in GML documents. It’s efficient to use database technologies to manage and maintain the geospatial data described by GML. It was in this background that began the research of the GML spatial database.

Besides the storage and access for data, the most important function of database is data query, and the GML spatial database is no exception. Because of the particularity of geospatial data, an actual GIS (Geographic Information System) involves a mass of complicated spatial data. In order to query GML data efficiently, appropriate index structure is important to the system efficiency.

Supported by the Fundamental Research Funds for the Central Universities (2009JC006)
*Corresponding author.
e-mail: sljstudent@163.com; kareny@mail.hzau.edu.cn

The index in a database is a data structure associated with the storage location of data, which can help to locate the tuples for a query. Thus the index structure built for the GML document should be based on its storage schema.

2. Storage of GML data

Data in a GML document include non-spatial data and spatial data. The non-spatial data usually describes simple attributes of a spatial object, such as name and population, so it is also called attribute data. Because of its obvious relational feature, it's relatively easy to map the attribute data to fields in a relational table. The spatial data describes geographic features of a spatial object, such as coordinates and ubiety, so it is usually complicated and diverse. It's suitable to use object types to represent spatial data. Thus using the object-relational database to store the data in a GML document is an appropriate selection. The GML spatial database is the object-relational database that organizes and manages GML data.

To store a GML document into the object-relational database, mapping rules between the application schema of the GML document and the database schema are used as follows.

- Create a single table for the first element defined after the element "schema" in the application schema.
- Create a single table for the element appears many times in the same layer of the GML document.
- Data described in the same layer are stored in a same object-relational table, in which attribute data are stored as the common columns, and spatial data are stored as the columns of object types.

Currently, many database products companies, including Oracle, IBM, Informix, etc., have extended the object-oriented mechanism in their latest products, such as Oracle Spatial [1], IBM DB2 Spatial Extender [2], ArcSDE [3] and PostgreSQL [4], etc. Oracle Spatial provides spatial object data type MDSYS.SDO_GEOMETRY, which can represent geometries, such as plane and polygon. Oracle Spatial stores the spatial data as the fields of SDO_GEOMETRY type in a table, and provides functions and operators for this type, thus implement the access, indexing and analysis of spatial data.

The way to store GML documents, setting reasonable mapping rules between the application schema of GML document and the database schema, actually imitate the method of XED (XML-Enabled Database) to store XML documents. Because GML developed from XML, they have the similar formats and ways to express information, so the imitation is natural and reasonable. Similarly, the expression of GML query and the structure of index can also borrow ideas from the existing management technologies of XML data.

3. XML path index technology

Many XML query languages use RPE (Regular Path Expression) to express XML queries. Regular path expression is a very useful tool to describe and represent the structure relationships between elements in semi-structured documents. Such as "bib/article/year, bib//conference", "bib[//year=2008]// conference" are regular path expressions. XML path indexes retrieve the path expressions to locate the elements a query asked for. According to the different building mechanisms, there are two kinds of XML path technologies, one is built based on the content of XML documents, and the other is built based on the schema of XML documents.

3.1 XML Path Index Based on Content of Documents

For this kind of indexes, XML document is converted into graph structure first, which will be made use of to build the index. Take the index structure proposed by Q.Li and B.Moon in XISS (XML Indexing and Storage System)[5] for example, it's encoding mechanism is based on the tree structure of XML documents. Elements, attributes and text data are converted into nodes of the tree, and there is a parent-child relationship between two nodes connected with an edge. XISS traverses the document tree and encode the nodes with code pair of <order, size> in preorder. By using the code pairs, XISS can quickly find the ancestor-offspring relationship between nodes and subordinate relationship between element nodes and their attribute nodes, and consequently locate the tuples corresponding to the query path. XISS also provides some flexibility for the insertion and update of nodes.

Currently, there are a number of index technologies are similar to XISS, such as XSet[6], ToXin[7], T-index[8], Index Fabric[9], etc.

3.2 XML path index based on schema of documents

This kind of indexes are built by making full use of the structure information the application schema contains, such as SphinX[10], DataGuides[11] proposed in Lore[12] system by Standford University.

Take SphinX for example, the XML document and its DTD (Document Type Definition) are both converted into tree structure. Each leaf node of the DTD links to the root node of a B- tree which contains the indexes for the atomic values in the document tree. SphinX uses structure information the DTD contains and combines the DTD with B- trees. Because DTD is a correct and complete extraction of path structure of the document, using DTD can effectively reduce the paths required to retrieve and improve the efficiency of retrieval when the scale of retrieve object is large.

Because GML documents are described in text and constituted with nested elements, just like XML documents, the GML queries can also be expressed in regular path expressions. For the query path expressions, effective path index should be built for calculating the GML queries.

4. Study of GML path index technology

In the process of mapping the GML document into the database, the document content can be retained well, while the structure information of the document may be lost. GML path index should be based on the storage schema of GML document, describe its data organization and retain its structure information, which will help to reconstruct the GML document.

4.1 Characteristics of GML data

GML documents describe geospatial data, so usually the amount of data they contain is a lot and the scale is large. But the application schema of GML document is usually not complicated and with relatively simple structure. As shown in figure 1 is the application schema of a GML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by gml (207) -->
<schema xmlns=http://www.w3.org/2001/XMLSchema
          xmlns:gml="http://www.opengis.net/gml"
……>
<element name="Map" type="ex:mapType" substitutionGroup="gml:_FeatureCollection"/>
<element name="Layer" type="ex:layerType"/>
<element name="Feature" type="ex:featureType" substitutionGroup="gml:_Feature"/>
<element name="SimpleProperty" type="ex:SimplePropertyType"/>
<element name="GeometryProperty" type="ex:GeometryPropertyType"/>
<element name="FloatProperty" type="ex:FloatPropertyType"/>
<complexType name="mapType">
  <complexContent>
  <extension base="gml:AbstractFeatureCollectionType">
   <sequence>  <element ref="ex:Layer"/>  </sequence>
  </extension>
</complexContent>
</complexType>
<complexType name="layerType">
  <sequence>
   <element ref="gml:featureMember" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
<complexType name="featureType">
  <complexContent>  <extension base="gml:AbstractFeatureType">
  <sequence>   <element ref="ex:SimpleProperty"/>
  <element ref="ex:GeometryProperty"/>  </sequence>
  </extension>
  </complexContent>
</complexType>
<complexType name="GeometryPropertyType">
    ……
<complexType name="SimplePropertyType">
  ……
<complexType name="FloatPropertyType">
    ……
</complexType>
</schema>
```

Fig. 1. Application schema of a GML document

The size of the application schema is 4.11K, and the size of its corresponding GML document, describing the information of states in the United States, is actually 1.3M.

4.2 GML path index

4.2.1 Analysis of the XML path indexes for reference

The GML document is usually in a large scale and contains a mount of elements. If encode each element like the XISS index mechanism, this encoding will cost too much, enough to offset the benefits the method brings. And there are deficiencies in the encoding method, it's hard to distinguish the parent-child relationship among the common ancestor-offspring relationships, which will not support the reconstruction of GML document very well. SphinX index is based on the schema of documents, using structure information the DTD contains to reduce the paths required to retrieve and improve the efficiency of retrieval. The application schema of GML documents also contains documents structure information, so it's reasonable to borrow ideas from SphinX to build GML path index based on the application schema of GML documents.

First, convert the GML document application schema into a tree structure, which is called schema tree, and then encode elements in the schema tree. Because the document application schema is usually not in a large scale, it's practicable to encode the schema tree. And the codes can help to ascertain the nested hierarchy between elements in GML document, which is useful for the reconstruction.

4.2.2 Generating application schema tree

When converting the application schema of a GML document into a tree structure, the first element defined after the element "schema" is converted as the root node, and then traverse all elements in the application schema layer by layer. The elements at the same layer in the schema are also at the same layer in the tree. Two elements one contains the other in the schema are parent-child nodes in the tree. Elements contained in the same element in the schema are sibling nodes in the tree. In the schema tree, a parent node and its child nodes are connected with an edge each; for the element appears many times at the same layer in the application schema, it can be considered that there is a one-to-many relationship between the element contains it and itself, and use an edge marked with '*' to connect them. As shown in figure 2 is the schema tree corresponding to figure 1.

4.2.3 Building GML path index

Travers the schema tree of GML document in preorder and encode every element at the same time (the codes in figure 2 are just the preorder encoding of this schema tree). And then generate an index table, which records the name, code, parent node code and child node code for every node in the schema tree. The structure of this index table is shown as table 1.

In this table, field "NOrder" means the preorder code of a node, which is a unique value; field "Node" means the name of a node; field "POrder" means the parent node code of a node; field "COrder" means the child node code of a node, which usually is a set of codes. If a node has no parent node or child nodes, the value of its POrder or COrder will be 0. Take the schema tree in figure 2 for example, the "NodeOrder" table for it is shown in table 2.

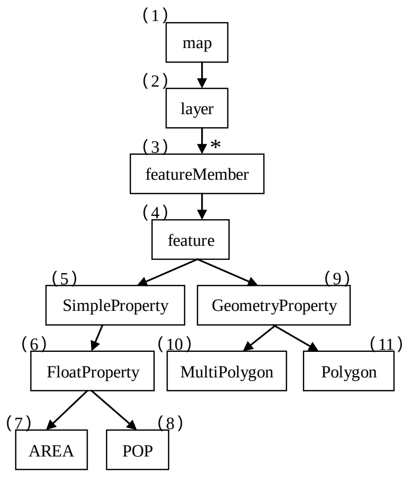

Fig. 2. Schema tree of a GML document application schema

Table 1. Structure of index table "nodeorder"

field	type
NOrder	NUMBER
Node	VARCHAR(32)
POrder	NUMBER
COrder	VARCHAR(32)

Table 2. "NodeOrder" generated according to schema tree

NOrder	Node	POrder	COrder
1	.map	0	2
2	map.layer	1	3
3	.featureMember	2	4
4	featureMember.feature	3	5,9
5	featureMember.feature.SimpleProperty	4	6
6	featureMember.feature.SimpleProperty .FloatProperty	5	7,8
7	featureMember.feature.SimpleProperty .FloatProperty.AREA	6	0
8	featureMember.feature.SimpleProperty .FloatProperty.POP	6	0
9	featureMember.feature.GeometryPrope rty	4	10,11
10	featureMember.feature.GeometryPrope rty.MultiPolygon	9	0
11	featureMember.feature.GeometryPrope rty.Polygon	9	0

When generating the node names in NodeOrder, the node without an entering edge or with an entering edge marked with ‘*’ use its own name, and add a ‘.’ before it, for easy to realize the algorithm; the node can be located by a nodes sequence starting from the above two nodes use the nodes sequence as its name.

Similarly to [13], decompose the GML query path expression into atomic paths first when using this path index. And then starting from the first atomic path, do the following by making use of the parent-child relationship information the “NodeOrder” records.

1. For the first atomic path, check “Node” field of every record in “NodeOrder” to get the characters sequence lies on the right side of the first ‘.’ from right to left. If the characters sequence of a record is the same as the first atomic path, pick the content of its “Node” field, represented as letter ‘P’. If the processing of the query path is not finished, turn to 2 °; else, turn to 5 °.

2. Get the “COrder” field of the record that ‘P’ exists, which means getting the child nodes of the node that ‘P’ represents. They are supposed to be a set of codes: {c1，c2，……， cn }, called ‘C’, ci(1≤i≤n) is the code for a child node of ‘P’.

3. Pick the next atomic path.

4. Check the codes in set ‘C’ one by one and find their corresponding records in “NodeOrder”. Get the characters sequences lies on the right side of the first ‘.’ from right to left in field “Node” of these records. If the characters sequence of a record is the same as the current atomic path, pick the content of its “Node” field and make it the current ‘P’. If the processing of the query path is not finished, turn to 2 °; else, turn to 5 °

5. Get the characters sequence lies on the left side of the first ‘.’ from left to right in ‘P’. This characters sequence is the name of the table that includes the tuple the query path asked for, and ‘P’ is the name of the field in the table. So through ‘P’ the table and the field can be located directly.

In the process of retrieving a GML query path, the child node codes recorded in index “NodeOrder” are made use of to reduce the searching scope of a next atomic path. According to the ‘P’ finally found, it can be quickly and easily to locate the table, the field and the detailed data.

On the other hand, index “NodeOrder” can help to reconstruct GML documents. The parent and child nodes of a node can be known with its field “POrder” and “COrder”. According to their names, the tables and fields corresponding to the parent and child nodes can be ascertained. Extract the data of these fields in these tables, and the GML documents can get reconstructed. If the parent and child node are not in a same table, the data can also be obtained by the join operation between the tables they exist.

Since GML queries also involve calculation and analysis of spatial data, it is necessary to create spatial index for these queries, such as R-tree series and quad-tree series. Oracle Spatial supports and maintains these two kinds of indexing methods, so the spatial indexes can be created for the fields of SDO_GEOMETRY type in the object-relational table. Oracle Spatial provides spatial operators and geometry functions to process spatial query, and the spatial index created on spatial data will be helpful in the process.

5. Conclusion

GML is the application of XML in geospatial information field, its index technology is one of the key technologies to implement the management of mass GML data, and the existing XML path index technologies are worthy to use for reference.

Data in a GML document include non-spatial data and spatial data, which are stored into object-relational tables in spatial database. GML queries are usually for both non-spatial data and spatial data, so the retrieval requires a combination of GML path index and spatial index. The GML path index “NodeOrder” proposed in this paper records the sequences, levels and parent-child relationships between elements in a GML document, so it can help to locate the tuples for a GML query quickly and also can help to reconstruct the GML document. If the analysis and calculation of the query also involve spatial data, with the spatial index created on the fields of object type which store the spatial data, spatial operators and geometry functions provided in spatial database can be used to get the query results.

References

[1] Oracle Spatial user's guide and reference. http://www.oracle.com/ technology/products/ oracle9i / release_1_techlisting.html, 2005
[2] IBM Corporation. IBM DB2 Spatial Extender User's Guide and Reference, Version 7.
[3] http://www.esri.com/software/arcgis/arcsde/about/overview.html
[4] http://postgis.refractions.net/documentation/
[5] Q.Li, B.Moon. Indexing and querying XML data for regular path expressions. Proc of 27th Intl. Conf. on Very Large Data Bases .2001. 361 - 370.
[6] B.Zhao, A.Joseph. XSet: A lightweight XML search engine for internet applications. http:// www.cs.berkeley.edu/%7Eravenben/xset/
[7] F.Rizzolo. ToXin: An indexing scheme for XML data. M. Sc. Thesis, Canada: Dept. of Computer Science, University of Toronto, January,2001.
[8] TMilo ,D Suciu. Index structures for path expressions. Intl. Conf.on Database Theory. 1997. 277 - 295.
[9] B.Cooper ,N.Sample ,M.Franklin , et al. A Fast Index for Semistructured Data. Proc. of 27th Intl. Conf. on VeryLarge Data Bases. August 2001. 341 - 350.
[10] L.K.Poola, J.R.Haritsa. SphinX: Schema-conscious XML indexing. Database Systems Laboratory Dept. of Computer Science &Automation Indian Institute of Science.
[11] R.Goldman, J.Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. Proceedings of 23th International Conference on Very Large DataBases (VLDB), Athens, Greece, August 1997, 436-445.
[12] J.McHugh, S.Abiteboul, R.Goldman, et al. Lore: A Database Management System for Semistructured Data. SIGMOD Record, 26(3): 54-66, September 1997.
[13] Dan Connolly. Evolution of Web Data Formats [DBOL] http://www. w3c.org/ Talks/9803xml-seattle, 19982031