An Approach for Integrating Heterogeneous and Loosely Coupled Geospatial Databases

Xia Peng[1], Zhou Huang[2*], Xuetong Xie[3]
[1]Institute of Tourism, Beijing Union University
Beijing, 100101, P. R. China
[2]Institute of Remote Sensing & GIS, Peking University
Beijing, 100871, P. R. China
[3]School of Geographical Sciences, Guangzhou University
Guangzhou, 510006, P. R. China

Abstract --***More and more geospatial databases are deployed on different hosts in the distributed environment currently. These databases are often heterogeneous and loosely coupled. They have different backup structures but can be accessed through the same human-machine interface i.e. GSQL (Geographic SQL). How to integrate these databases is very crucial for geospatial applications under the Internet environment. An approach for distributed geospatial query processing is discussed in this paper. The detailed methods on how to translate the global GSQL query into a distributed query workflow is introduced. Using this method, a global geospatial query is translated into a query workflow composed of several local GSQL statements which can be issued onto the distributed and heterogeneous geospatial databases.***

1. INTRODUCTION

With the development of information technology, particularly with the rapid development of distributed computing technologies, more and more geospatial databases are deployed on different hosts in the distributed environment (Local Area Network or Wide Area Network like the Internet). These databases are often heterogeneous and loosely coupled. But applications are often attended to perform integrated query or analysis onto the different and distributed geospatial databases. Thus how to deal with the distributed query processing problem is a challenging issue in the geospatial database research area. The key to achieve distributed query on the distributed geospatial databases is query language processing. As a primary measure taken to work with databases, geospatial query language is one of the foundation elements of spatial database systems [1]. A lot of research has been seen in this field. As indicated in reference [2], extending SQL (Structured Query Language), namely constructing GSQL (Geographic SQL), is proved to be a feasible and effective method to realize geospatial query language and to provide the support for accessing and managing geospatial data. GSQL extended by SQL is widely used in geospatial databases now and adopted by OGC standards. Methods on GSQL implementation, formal definition, geospatial predicates and geospatial functions extension are discussed frequently. In fact, OGC provided the specification for simple feature operations, including the definition of spatial operators, spatial relation measurement, and so on [3].

On the other side, because spatial database are commonly built upon the distributed environment, more and more attentions are drawn to the research for distributed spatial data management. Distributed data storage and processing are the main features of distributed spatial data management. Hence, the research on processing mechanism of distributed geospatial query will turn out to be an important subject in the field of geospatial query language itself in addition to the research on the geospatial database. A major problem we are facing is how to turn the global geospatial query statement to the local query statements which are processed in local computers respectively. The mechanism of the distributed geospatial query processing in the distributed environment, which is discussed in this paper, is the key solution for this problem.

2. FORMALIZATION OF GEOSPATIAL QUERIES ONTO THE GEOSPATIAL DATABASE

Spatial Operations can be divided into two basic types i.e. read operation and write operation. Read operation is more common, e.g. SELECT query or spatial analysis onto spatial data. Write operation is used to accomplish update tasks onto spatial data, e.g. insert or delete. Hence GSQL query statements are classified as four sub-types: select statement (GSQLSelect), insert statement (GSQLInsert), update statement (GSQLUpdate) and delete statement (GSQLDelete). In normal, the grammar of formal languages can be defined through BNF (Backus-Naur Form). So we propose the BNF definitions of GSQL statements as follows:

Table 1. BNF definitions of GSQL

GSQLStatement ::= (GSQLSelect | GSQLInsert | GSQLUpdate | GSQLDelete) ";"
GSQLSelect ::=<SELECT> GSQLSelectCols (<INTO> GSQLSelectCols)? <FROM> GSQLTableList (GSQLWhere)? (GSQLGroupBy)? (GSQLOrderBy)?
GSQLInsert ::= <INSERT> <INTO> GSQLTableList ("(" GSQLSelectCols ")" <VALUES>)? "(" GSQLSelectCols ")"
GSQLUpdate ::= <UPDATE> GSQLTableList <SET> (GSQLUpdateAssignment (",")?)+ (GSQLWhere)?
GSQLDelete ::= <DELETE> <FROM> GSQLTableList (GSQLWhere)?

GSQLSelectCols ::= (<ALL> | <DISTINCT>)? (<ASTERISK> | GSQLSelectList)
GSQLSelectList ::= GSQLSumExpr ("," GSQLSumExpr)*
GSQLTableList ::= GSQLTableRef ("," GSQLTableRef)*
GSQLTableRef ::= <ID> (<ID>)?
GSQLSumExpr ::=GSQLProductExpr (("+" | "-") GSQLProductExpr)*
GSQLAndExpr ::= GSQLNotExpr (<AND> GSQLNotExpr)*
GSQLBetweenClause ::= (<NOT>)? <BETWEEN> GSQLSumExpr <AND> GSQLSumExpr
GSQLColRef ::= GSQLLvalue
GSQLCompareExpr ::=(GSQLSelect | GSQLIsClause | GSQLExistsClause | GSQLSumExpr (GSQLCompareExprRight)?)
GSQLCompareExprRight ::=(GSQLLikeClause | GSQLBetweenClause | GSQLCompareOp GSQLSumExpr)
GSQLCompareOp ::= (<EQUAL> | <NOTEQUAL> | <NOTEQUAL2> | <GREATER> | <GREATEREQUAL> | <LESS> | <LESSEQUAL>| <GEOOP>)
GSQLExistsClause ::= <EXISTS> "(" GSQLSelect ")"
GSQLFunction ::= (<MAX> GSQLFunctionArgs | <MIN> GSQLFunctionArgs | <SUM> GSQLFunctionArgs | <COUNT> GSQLFunctionArgs | <GEOFUNCTION> GSQLFunctionArgs| <ID> GSQLFunctionArgs)
GSQLFunctionArgs::="(" (GSQLSumExpr ("," GSQLSumExpr)*)? ")"
GSQLGroupBy ::= <GROUP> <BY> GSQLOrderByList
GSQLIsClause ::= GSQLColRef <IS> (<NOT>)? <NULL>
GSQLLikeClause ::= (<NOT>)? <LIKE> GSQLPattern
GSQLLiteral ::= (<STRING_LITERAL> | <INTEGER_LITERAL> | <FLOATING_POINT_LITERAL> | <ASTERISK>)
GSQLLvalue ::= (GSQLLvalueTerm)
GSQLLvalueTerm ::= <ID> (<DOT> <ID>)*
GSQLNotExpr ::= (<NOT>)? GSQLCompareExpr
GSQLOrderBy ::= <ORDER> <BY> GSQLOrderByList
GSQLOrderByElem ::= GSQLColRef (GSQLOrderDirection)?
GSQLOrderByList ::= GSQLOrderByElem ("," GSQLOrderByElem)*
GSQLOrderDirection ::= (<ASC> | <DESC>)
GSQLOrExpr ::= GSQLAndExpr (<OR> GSQLAndExpr)*
GSQLPattern ::= (<STRING_LITERAL>)
GSQLProductExpr ::= GSQLUnaryExpr (("*" | "/") GSQLUnaryExpr)*
GSQLTerm ::= (("(" GSQLOrExpr ")") | GSQLColRef | GSQLLiteral | GSQLFunction)
GSQLUnaryExpr ::= (("+" | "-"))? GSQLTerm
GSQLUpdateAssignment ::= GSQLLvalue "=" (GSQLTerm | GSQLSumExpr)
GSQLLValueElement ::= (<NULL> | GSQLSumExpr | GSQLSelect)
GSQLLValueList ::= GSQLLValueElement ("," GSQLLValueElement)*
GSQLWhere ::= <WHERE> GSQLOrExpr

3. QUERY PARSING APPROACH FOR GSQL STATEMENTS

Using GSQL statements, we can depict any geospatial query from a problem-oriented view. The differences between GSQL and normal SQL is GSQL adds particular geospatial functions and operations based on SQL. Some typical GSQL queries are listed as follows:

Table 2. GSQL query examples

E1	*//Get all information about the city objects.* *select citty.** *from city*
E2	*//Get the names of the rivers whose flux is greater than 600.* *select river.name* *from river* *where river.flux>600*
E3	*//Get all information about the residences which is within 3km distance away from the 201 national highway.* *select residence.** *from residence,road* *where road.name =’201’ and withinBufferByDistance(residence.geometry, road.geometry, 3)*

To deal with distributed geospatial query processing, the first step is understanding the GSQL query statement. This can be achieved through the GSQL compiling technique. Using compiling approaches a GSQL statement

in form of string is translated into a well-formed query syntax tree. The process flow for GSQL query compiling is concluded as follows:

1) Lexical analysis. Firstly using the scanning technique the GSQL string is split into several words i.e. "TOKEN".

2) Syntax analysis. Based on lexical analysis, the tokens are processed and translated into a query syntax tree composed of various basic query elements, e.g. table name, attribute list and where clause.

3) Type checking. After the syntax tree is built up, check whether all elements are matched with system symbols or not. If there is no unmatched error, a checked well-formed query syntax tree is built up.

4. GSQL PARSER IMPLEMENTATION

Based on the aforementioned compiling steps, we implement a GSQL compiling tool i.e. GSQL Parser to build geospatial query trees. GSQL Parser is able to perform lexical analysis, syntax analysis and check the correctness of query elements. Furthermore, GSQL Parser provisions a GUI interface for viewing the GSQL query in the form of both graphical tree (Figure 1) and XML (Figure 2).

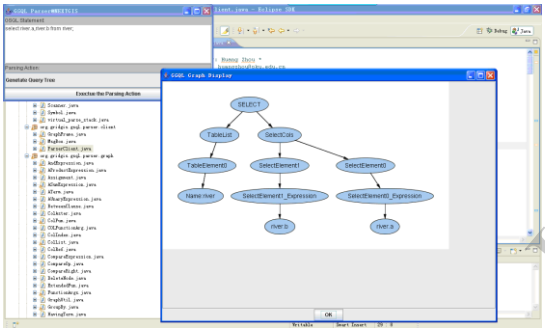

Figure 1. Graphical GSQL query tree

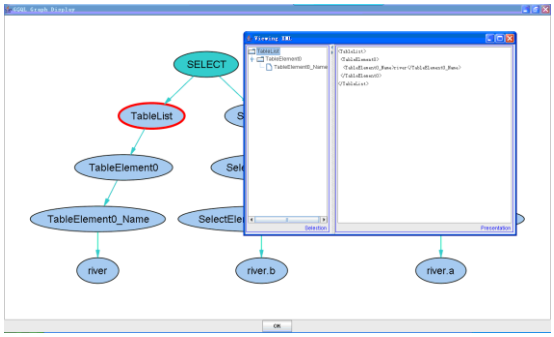

Figure 2. Query tree after statement optimization

For example, if we want to view attribute a and b information about the river objects, a GSQL query "*select river.a,river.b from river*" is submitted. GSQL Parser would parse the query string into a graphical tree as shown in Figure 3, and provide an XML view of the query tree as shown in Table 3.

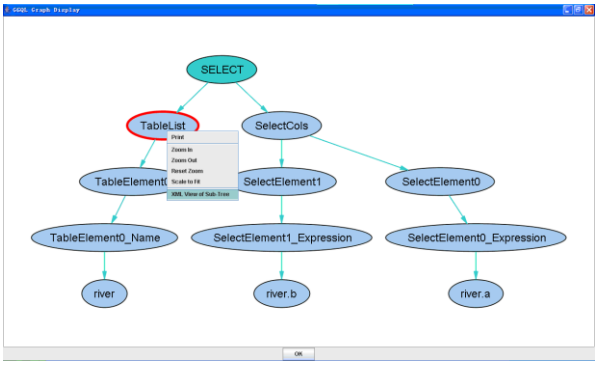

Figure 3. Query tree after statement optimization

Table 3. BNF definitions of GSQL

```
<SELECT>
 <TableList>
  <TableElement0>
   <TableElement0_Name>river</TableElement0_Name>
  </TableElement0>
 </TableList>
 <SelectCols>
  <SelectElement0>
<SelectElement0_Expression>river.a</SelectElement0_Ex
pression>
  </SelectElement0>
  <SelectElement1>
<SelectElement1_Expression>river.b</SelectElement1_Ex
pression>
  </SelectElement1>
 </SelectCols>
</SELECT>
```

5. DISTRIBUTED PROCESSING MODEL FOR THE GSQL QUERY

Under the distributed computing environment, data is logically integrated while distributed physically [4]. A distributed data management system should be featured with the functions of data independence, centralized and autonomous management, data redundancy and distributed transaction [5]. Compared with centralized management model, the benefits of distributed data manangement are as follows:

(1) Meeting the need for geospatial data sharing among different organizations. (2) Load balancing. By balancing the load between computing nodes we can avoid system critical bottle-neck. (3) High reliability. Geospatial data is stored and distributed with duplicates, so accidents in single computing note won't cause failure of the entire system.

Meanwhile, the query and operation on spatial data are more complicated because of the distribution and redundancy of data. Therefore, the research on distributed query and operation is of great necessity.

Not like the centralized query, the distributed query is processed by multiple data nodes. Spatial data is cut into slices which are logically as a whole and physically stored in different data nodes. Therefore, when the distributed

query (namely global query) is submitted, we need to parse the global query into subsidiary queries which are processed in individual data notes respectively. At the same time, for the parsing of global query, data redundancy and duplicate both provided the possibility of parallel processing and raised the problem of data consistency. As for the geospatial data, the general distributed geospatial query process can be summarized as the following three steps:

(1) Query decomposition. After understanding the query semantic through GSQL parsing, the global query submitted by users is converted and decomposed into a combination of multiple sub-queries. During this step, geospatial data distribution status is required, which is provided by the geospatial data resource list.

(2) Processing sub-queries. The sub-queries are combined and sent to individual nodes for processing. The query result of each sub-query is sent back and assembled into the final result. The operations in this step are undertaken by the distributed processing engine.

(3) Displaying query result. Users can get the information of the query result intuitively through the graphical interface.

Figure 4 illustrates a typical distributed geospatial query process. A user submit a global query in form of GSQL string and the query datasets road and residence are distributed on different hosts. Hence, the query string would be parsed automatically, and then decomposed into several sub-queries which are issued onto different data nodes.

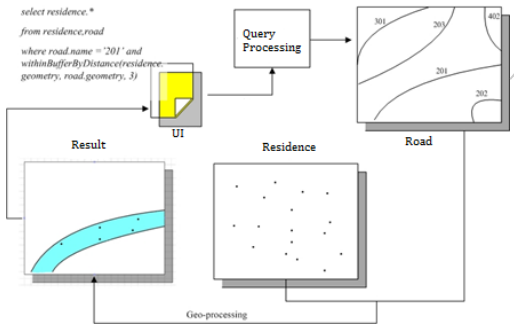

Figure 4. Distributed query processing demonstration

6. DISTRIBUTED GEOSPATIAL QUERY PROCESSING EXAMPLE

Suppose a user submit a global GSQL statement: "*select river.name from river where river.flux>600*", and the distributed computing environment is composed of 3 data nodes: {node1, node2, node3}. The distributed geospatial query process is illustrated as follows:

(1) The construction of GSQL query tree

The query string: "*select river.name from river where river.flux>600*", through lexical analysis and syntax analysis, is transformed into the query tree shown in Figure 5.

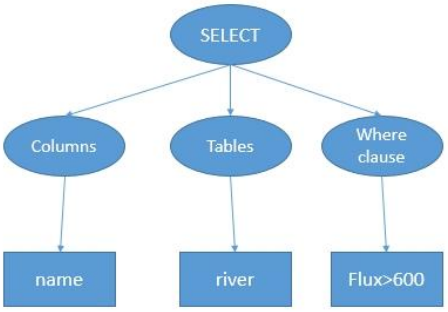

Figure 5. Geospatial query syntax tree

(2) GSQL workflow building

The node list (regardless of origin and duplicate) retrieved through spatial resource list interface is { node1, node2, node3}, the GSQL workflow constructed by our method is shown in Figure 6.

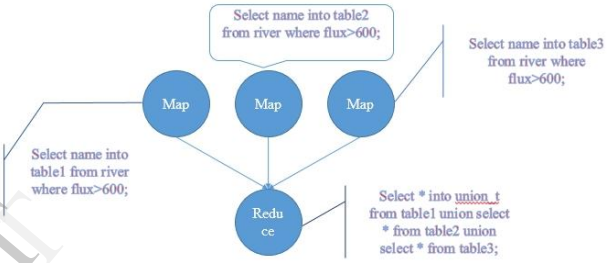

Figure 6. Distributed geospatial query workflow

From the figure we could see the workflow is a typical MapReduce process. Three sub-queries are issued onto node1,node2 and node3 in parallel and then a union operation is performed to reduce the results. After execution of the reduce operation the whole distributed query process is completed.

7. CONCLUSIONS

Nowadays, various applications of distributed geospatial database are increasing. The mechanism of distributed geospatial query processing is a major problem toward this issue. The work carried out in this paper mainly includes the following aspects:

(1) Analyzing the characteristics of geospatial query. The general BNF definitions of GSQL query language are discussed.

(2) Implementing a GSQL Parser that enables lexical analysis, syntax analysis and query tree construction. The parser provisions both graphical tree and XML view for geospatial queries.

(3) Three major steps of distributed GSQL query processing is thoroughly discussed, and a demonstration geospatial query is provided to describe the query process.

Furthermore, study on distributed GSQL query optimization based on dynamic strategies will be performed to improve the query efficiency.

REFERENCES

[1] S. Shekhar, S. Chawla. Spatial Databases——A Tour[M]. Prentice Hall,2003.

[2] M. J. Egenhofer. SpatialSQL,A query and presentation language[J].IEEE Transactionson Knowledgeand Data Engineering,1994,6(1):86-95.

[3] OpenGIS Consortium Inc.OpenGIS simple features specification for SQL1.1[S/OL], http://www.opengis.org/docs/99-049.pdf,1999.

[4] R. A. Haraty, R. C. Fany. Query Acceleration in Distributed Database Systems[J]. Colombian Journal of Coputation, 2001,2(1):19-34.

[5] M Tamer Ozsu, Patrick Valduriez. Principles of Distributed Database Systems[M].Tsinghua Press,2002.