A Spatial Structural Query Language—G/SQL

Yu Fang Fang Chu Tang Xinming

Peking University, 100871

Beijing, China

[Abstract]

Traditionally, Geographical Information Systems can only process spatial data in a procedure-oriented way, and the data can't be treated integrally. This method limits the development of spatial data applications. A new and promising method to solve this problem is the spatial structural query language, which extends SQL and provides integrated accessing to spatial data. In this paper, the theory of spatial structural query language is discussed, and a new geographical data model based on the concepts and data model in OGIS is introduced. According to this model, we implemented a spatial structural query language G/SQL. Through the studies of the 9-Intersection Model, G/SQL provides a set of topological relational predicates and spatial functions for GIS application development. We have successfully developed a Web-based GIS system—WebGIS—using G/SQL. Experiences show that the spatial operators G/SQL offered are complete and easy-to-use. The BNF representation of G/SQL syntax is included in this paper.

[Key Words] Spatial Structural Query Language, G/SQL, Geographical Data Model, Spatial Database.

G/SQL research background

Geographic Information System (GIS), rapidly developed, is now debouching out from special realm and gaining wide prevalence. It is even now applied in the information services toward public. The development of GIS is not only benefited from the progress of computer science and technology, but also is pushed by the increment of spatial data globally. People are now even conceiving of the prospective of so-called "Digital Earth". The aim of the Digital Earth is to build up a complicated huge information system with a spatial location as a backbone and provides speed, correct, sufficient and complete information for various applications. Under such a situation improving spatial data handling and interfaces is imperative.

In traditional GIS the method to handle spatial data is "process-oriented" and treats geometry and attribute data separately. The key reason causing this problem lies in lacking of data warehouse and system that can uniformly manage geometry and attribute data. This paper proposes a spatial structural query language -- G/SQL, based on a spatial data model and spatial databases, and implements this language in our WebGIS software in Peking University.

Several papers have been published on research

on this field. Allen [1] formalized temporal topological relations for temporal data. Egenhofer and Franzosa [4] proposed their famous 9-intersection models for spatial topological relations between spatial features. Egenhofer [5] then prototyped a spatial data query model. Chen and Zaniolo [3] realized a spatio-tempoal query language for accessing spatio-temporal data on a single map. There are other researches for the related applications [3][6][7][10][12].

This paper offers a canonical definition for spatial data management and spatial query statement. The implementation of G/SQL has referred to manipulation specifications made by OGC (Open GIS Consortium) [8]. Since the OGC does not confine the implementation methods and processes for its series of open development standards, this paper introduces the concepts of spatial structural query language and provides a method that has been implemented by G/SQL in the client/serer structure, referring to the OGC standards. The initiative of the paper is to stimulate the research and discussions as well as more development of new products on spatial query.

A WebGIS --- an implemented system with G/SQL functions

The development of our WebGIS system aims at the GIS application in Web. The system adopts the client/server structure. The spatial data are stored in the spatial database on the server and managed uniformly by the spatial data management system. The system offers the front user by Java applets. A user can browse, query, zoom and roam the spatial data by downloading the system applets.

WebGIS structure

The client end of WebGIS contains Java Applets while the server contains spatial data, and GIS server components containing G/SQL compiler and the geo-information manipulation machine as well as the component for external data transfer.

The Java component at the client end is responsible for the user interaction: it can display the map that is downloaded from the server in plenty of map display models. It supports the manipulation such as browsing, zooming and roaming. It accepts the user requests through a straight and convenient form interacting with G/SQL, and transforms the forms into a G/SQL statement and sends to the server for data handling. It also receives the results which are sent back by the WebGIS server.

The WebGIS server is software running on the SUN platform. The server receives a G/SQL request, compiles by the G/SQL compiler into geo-information manipulation atomic statement sequence, and submits to execute by geo-information manipulation machine. The processed result will be clipped into GDTP data package and be sent back to the user end, where GDTP (Geodata Transfer Protocol) is a protocol followed by spatial data transferring in WebGIS. The server adopts the multi-thread structure in order to offer high performance service and sufficiently utilize the system resources.

The data source of the entire system is obtained by the external data transfer component. The component transfers the data in the old GeoInfo platform, E00 format, and Chinese Spatial Data Transfer Standard (CSDTS) into the spatial database. G/SQL can also implement the organization of these data and generate relating system information in the spatial database.

G/SQL at server end and client end

The client end runs Java Applets of WebGIS in the standard navigator that supports JAVA, while the server end runs the application programs on SUN workstation. Both these ends, though they have different technologies and different environments, can make use of the functions of G/SQL sufficiently.

To Java Applets, G/SQL actually plays a role of data request protocol between server and client ends. G/SQL can express a user's requests completely and concisely and let Java Applet send the request to the server at one time. In this process, the client end enjoys the facility that can be called "non-process-oriented". For example, in traditional GIS, when a user searches for the hotels within 2 kilometers from the train station, he has to do several operations: buffering 2 kilometers centered at the train station, and overlaying the buffer zones with all hotels. Now a user can not only select such a way to find the hotels, but also write one G/SQL statement to express his request entirely. The Java Applet programmer can also simplify the complexion of software development by use of G/SQL.

To the external data transfer component, G/SQL provides the facility and security mechanisms for transferring of external spatial data into the spatial database. G/SQL has a series of definition statements and updating statements for manipulation of various spatial objects in databases, for guarantee of the completeness, consistency and correctness of these spatial objects, and for realization of the operation and updating of normal objects and data in spatial databases.

Spatial data model

The spatial data model in WebGIS refers to the Open Geodata Interoperability Specification (OpenGIS) [8]. The model takes the spatial features as its center and organizes the spatial database in multiple levels, which is guided by the Open GIS Specifications [9].

Spatial feature model

In WebGIS, a spatial entity is described by a composition of its geometry and attribute. Spatial features are organized into map layers, and each layer corresponds to a map layer table in the relational databases, which stores the attribute data and a special type GEOID. The geometry database adopts the geometry layer, which was used in the old GeoInfo platform. Every spatial feature in the map layer table is

linked with its geometry in the map layer through an identity.

The structure sketch of the geometry class is illustrated in Figure 1.

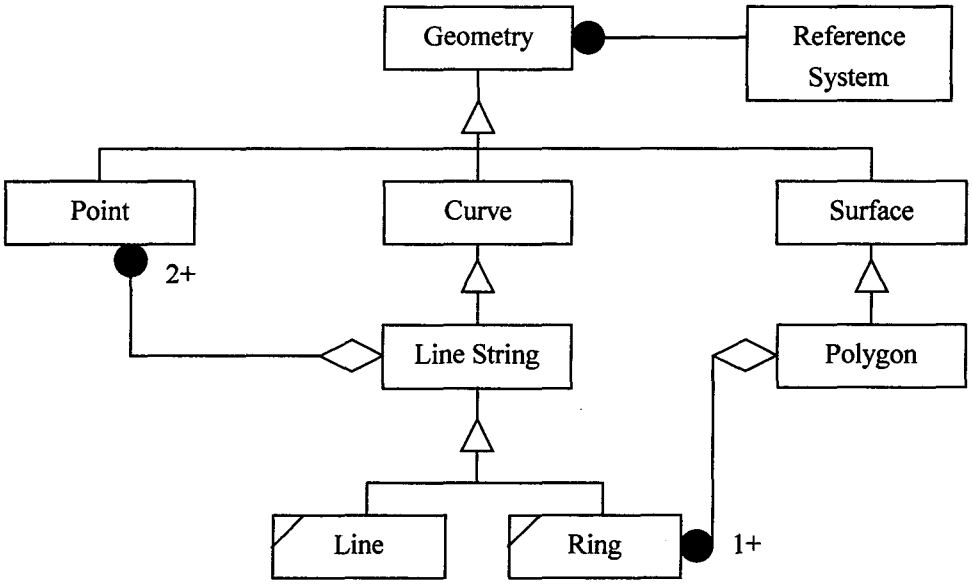


Figure 1: Levels of geometries

We adopt object-oriented method to analyze the class level of each geometry. The top layer refers to the "super geometry" in the spatial reference system. The geometry can be divided by:

0-dimensional topological structure – point: denotes the single location in the coordinate space. A point has a x and a y coordinates, without boundary.

1-dimensional topological structure – curve (Figure 2): is the sequence of points. The interpolation method between points is defined by the sub-class of the curve. If a curve does not cross at any point, then it is called "simple". It the start point and the end point is the same, then the curve is called "closed". A closed curve has no boundary, while the boundary of a non-closed curve is composed of its both end points.

Currently WebGIS defines one sub-class --- Line string, of a simple curve only. That is, a linear interpolation method is adopted between points. The adjacent points of a line string define a line. If a line string has only two points, then it becomes a line. If a line string has the same point at its two ends, it becomes a ring.

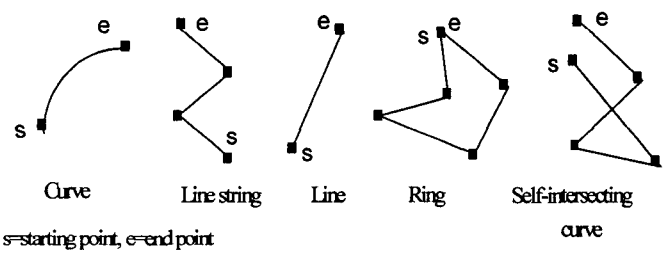


Figure 2: Curve geometries

2-dimensional topological structure – surface (Figure 3): is composed of one external boundary and zero / more internal boundaries. The boundary of a surface is the set of these boundaries. If a surface is not-self-intersected and simple, then it becomes a polygon. WebGIS supports Polygon. A polygon is composed of finitely mutually non-intersected rings, where one of these rings is the external boundary of the polygon, and the interior of the polygon is the area between external boundary and internal boundary.

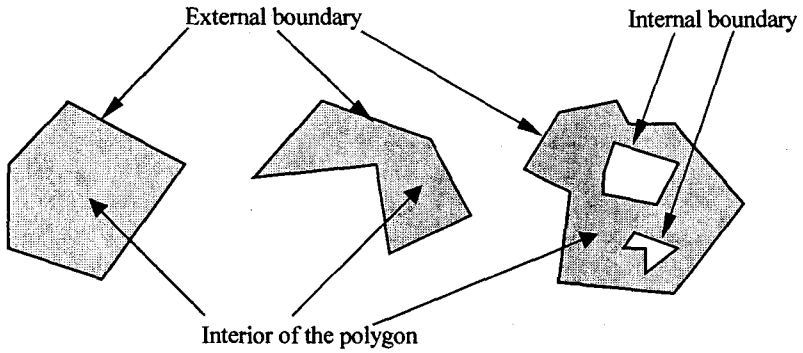


Figure 3: Polygon geometries

Spatial database model

The data model of the spatial database is designed based on the spatial features defined by OGC. A spatial feature is composed of attribute and geometry parts. These two types of data are stored separately in the spatial database: the geometric data is stored in the file system that was adopted in the old GeoInfo platform with quad-tree and B+ tree indices; the attribute data is stored in the relational database. The spatial database integrates the attribute and geometry data through the notation of spatial features. By use of spatial structural query language (G/SQL), the system shields the difference between the screen and internal storage, so that a user "touches" the data as one uniform relational database.

The organization of spatial features adopts multi-layer database model (Figure 4)

The following is the description of each spatial object and notation supported by multi-layer spatial database model in WebGIS:

1.Spatial feature: It is the abstract of entire spatial entity in the real world, with geometric and semantic attributes,which are represented by its geometric and semantic forms respectively.

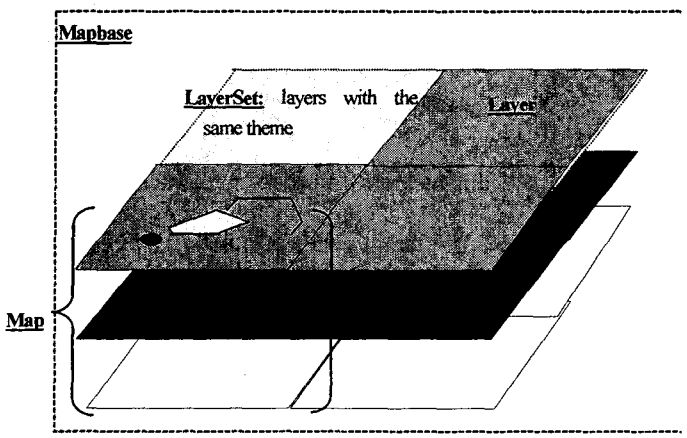


Figure 4: Level structure of spatial features

2.Geometry: It is the geometric attributes of a spatial feature, represented by point, line string or polygon etc. Actually it is the union of coordinate geometry with the spatio-temporal reference system.

3.Layer: the minimum set of spatial features is the layer. A layer defines the representation form of the semantics of spatial features. The geometric attributes of all spatial features constitute a set, which is called GeoLayer. The layer itself has some attributes like its name, coverage and subject. The spatial manipulation and analysis of G/SQL are exerted on the layers. It is the minimum unit of G/SQL manipulation.

4.GeoLayer: It is the layer consisting of geometric attribute of spatial features in the layer. It defines the map size, outline, projection and topological relations between geometries of a layer.

5.Map: It is the set of related layers. Generally it is the set of layers covering the same geographic area. All layers of one map have the same maps size, outline and projection.

6.Theme: it is the ordered set of the attribute structure of layers, representing some meaningful combination of attributes. Only the layers with the same attribute structures can be called the same theme.

7.LayerSet: It is the set of the same themes. Generally these layers cover a certain coverage, but they need not to have the same spatial coverage. That is, there is no limitation if these layers are adjacent or separated.

8.Mapbase: It is the set of LayerSet. All layersets should have the same coverage size, outline and project and structure, i.e., the layers in a MapBase should have one-one relations between each other and can form a map logically.

From the above description we can perceive that various constraints including completeness and consistency can be defined between layers, layersets, maps and mapbases. All these constraints are the important part of the level model of the spatial database. The spatial database stores and maintains relations between different levels of spatial data, and the constraints on completeness and consistency between levels through the system table. G/SQL provides a series of definition statements and updating statements in order to realize the creation and updating of objects in each level in the spatial database. The various constraints between levels can be also maintained during the statements execute. Furthermore, G/SQL provides a series of statements for consistent checking the consistency between levels explicitly.

G/SQL statements

The emphasis of G/SQL designing is the SELECT statement. The design of this statement starts from the research and classification of topological relations and spatial functions. G/SQL preserves the topological relations and spatial functions as key words, which are called spatial functors. G/SQL offers plenty of spatial functors with strong expressions and

can meet most requirements of GIS applications.

This section will describe the predicates of topological relations and spatial functions, then gives the grammar table (BNF) and discusses the SELECT statement.

Predicates of topological relations

A topological predicate is a Boolean function to determine whether a certain topological relation between geometry holds or not.

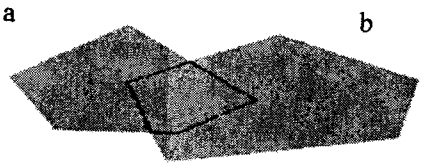
The basic method to determine the topological relation between two geometries is to classify the nine kinds of relations between the interior, boundary and exterior of spatial objects, and ascertain the meaningful combinations for overlaying, and then analyze the result of calculation. This method is called the 9-intersection model.

The boundary of a spatial feature is the set of geometries whose dimension is one dimension less than the feature. The boundary of a point is empty, the boundary of a non-closed curve is the set of two end points, and the boundary of a polygon is the set of curves that compose the polygon. The interior and the exterior are only fitful to the spatial features that are closed. The point set excluding the boundaries is the interior of the feature. The point set, which is neither in the interior nor at the boundary forms the exterior of the feature.

The 9-intersection model lists exhaustively all possible topological relations between two features and can express these relations powerfully. Given a spatial feature *a* and denote its boundary, interior and exterior by B(a), I(a) and E(a) respectively. Then the intersection results between B, I and E of two features can be the set of geometries of different dimensions, as well as the empty set. For example, the intersection between two lines that are crossed but not overlaid contains only the point of dimension 0. The intersection result between two lines that are partly overlaid contains a line of dimension 1. Denote the maximum dimension of the intersection between two geometries by Dim(x), then the value ranges {-1, 0, 1, 2}, where $-1 = \dim(\phi)$. Then the dimension matrix of the 9-intersection model will be the following:

	B (b)	I (b)	E (b)
B(a)	Dim(B(a)∩B(b))	dim(B(a)∩I(b))	dim(B(a)∩E(b))
I (a)	Dim(I(a)∩ B(b))	dim(I(a)∩I(b))	dim(I(a)∩E(b))
E(a)	Dim(E(a)∩B(b))	dim(E(a)∩I(b))	dim(E(a)∩E(b))

The following example shows the dimension matrix between polygon *a* and *b*.



	B(b)	I(b)	E(b)
B(a)	0	I	1
I (a)	1	2	2
E(a)	I	2	2

The topological relation between two spatial features can be determined by the following way. Given two spatial features, and a modal matrix of the 9-intersection model representing the dimension that is acceptable, if the result after the spatial operation is corresponding to any one of acceptable dimension values, then the predicate returns true, otherwise false. A modal matrix has 9 values, corresponding to the 9 elements in the 9-intersection matrix, and the results ranges {T, F, 0,1,2}. Suppose x is the result of an operation, the returned value will have the following meanings:

T: $\dim(x) \in \{0, 1, 2\}$, $x \neq \phi$;

F: $\dim(x) = -1$, $x = \phi$;

*: dim(x) is any value of set {-1, 0, 1, 2};

0: $\dim(x) = 0$;

1: $\dim(x) = 1$;

2: $\dim(x) = 2$。

The 9-intersection model can detect many kinds of topological relations, and it also allows a user to assign the topological relations that he interests. However, it is not like a natural language and is difficult to use. Therefore, in more cases, a user would like to adopt a simple and clear query such as, searching provinces that are *intersected* with a river.

To meet the practical requirements, OGIS defines five named spatial predicates: disjoint, touch, contain, and overlay. These predicates meet the following conditions:

They are mutually independent;

They are a complete covering set of all topological relations;

They are applicable for the geometries of the same or different dimensions;

Every predicate can be formalized by the 9-intersection model;

Given a method to express the boundary of a geometry and the start and end points of a line, any mode of the 9-intersection model can be formalized as a Boolean expression of these four predicates.

The meaning of these predicates is listed in the following:

Disjoint: there is no common point between two geometries.

Touch: If two geometries have a common point, and the intersection between their interiors is empty, then the relation *touch* holds. It is applicable for polygon/polygon, polygon/line, line/line, polygon/point, line/point, but not applicable for point/point.

Contain: If Geometry *a* is contained in the interior of Geometry *b*, and is not intersected with the exterior of the boundary, then b *contains* a.

Overlay: Part of Geometry *a* is superposition (overlapped) with Geometry *b* and they are not contained with each other. It is applicable to point/point, line/line/ line/polygon and polygon/polygon.

For the convenience, the following predicate can be derived based on the above predicates:

Within: If Geometry *a* contains Geometry *b*, then *b is within a.*

Intersect: If Geometry *a* and *b* are not disjoint, then *a intersects with b.*

Based on the above predicates, we can get a list of practical predicates for determination of the topological relations between geometries:

Equal: determine if two geometries are the same in space.

Disjoint: determine if two geometries are spatially disjoint.

Intersect: determine if two geometries are spatially intersected.

Touch: determine if two geometries are spatially touched.

Within: determine if one geometry is within the other spatially.

Contain: determine if one geometry contains the other spatially.

Overlap: determine if two geometries are spatially overlapped.

In the current version of WebGIS, G/SQL has implemented the following predicates: equal, disjoint, touch, within, contain and overlap.

Spatial functions

A spatial function operates on a single or more

geometries and calculates a value or a new geometry after the operations. The definition of the spatial functions also refers to the 9-intersecion models.

The spatial functions are classified into unary and binary spatial functions according to the number of objects in the functions. In our system the functions are all limited within the 2-dimensional coordinate system.

1. Unary spatial functions

Point-based spatial functions:

Function name	Function
GEOXcoor	Get the *x* coordinate
GEOYcoor	Get the *y* coordinate

Curve-bases spatial functions:

Function name	Function
GEOLength	Calculate the length of a curve
GEOSlope	Calculate the gradient of a curve
GEOStartPnt	Get the first point of a curve
GEOEndPnt	Get the last point of a curve

Polygon-based spatial functions:

Function name	Function
GEOArea	Calculate the area of a polygon
GEOPerimeter	Calculate the perimeter of a polygon
GEOMER	Calculate the minimum bounding rectangle of a polygon
GEOCentroid	Calculate the centroid of a polygon.
GEOInnerPnt	Calculate a point within a polygon.
GEOArcs	Get all line strings of a polygons

Spatial functions on points, curves and polygons:

Function name	Function
GEOBuffer	Calculate the buffer zone

2. Binary spatial functions

Function name	Function	Geometry variables (arguments) and their order
GEODistance	Return the minimum distance between two geometries	All variables
GEOOverlay	Return the overlapped part of two geometries	Polygon-polygon
GEOMinus	Return the difference of two geometries	Polygon-polygon
GEOLeftCut	Calculate the left part of the polygon when a curve crosses through a polygon.	Polygon-curve
GEORightCut	Calculate the right part of the polygon when a curve crosses through a polygon.	Polygon-curve
GEOIntersect	Calculate the part of a curve when a curve crosses through a polygon.	Curve-polygon

4.3 BNF representation of G/SQL

The BNF representation of G/SQL is summarized in the following expressions:

check-coherence-on-layer-statement ::=
CHECK COHERENCE ON LAYER layer-name
check-coherence-on-layerset-statement ::=
CHECK COHERENCE ON LAYERSET layerset-name
check-coherence-on-map-statement ::=
CHECK COHERENCE ON MAP map-name
check-coherence-on-mapbase-statement ::=
CHECK COHERENCE ON MAPBASE mapbase-name
create-layer-statement ::=
CREATE LAYER layer-name
BINDING file-path

create-layerset-statement ::=
CREATE LAYERSET layerset-name
TEMPLATE layerset-template-name
create-map-statement ::=
CREATE MAP map-name
TEMPLATE map-template-name
create-mapbase-statement ::=
CREATE MAPBASE mapbase-name
TEMPLATE mapbase-template-name
delete-layer-from-layerset-statement ::=
DELETE LAYER layer-name
FROM LAYERSET layerset-name
delete-layer-from-map-statement ::=
DELETE LAYER layer-name
FROM MAP map-name
delete-layerset-from-statement ::=
DELETE LAYERSET layerset-name
FROM mapbase-name
delete-map-from-statement ::=
DELETE MAP map-name
FROM mapbase-name
drop-layer-statement ::=
DROP LAYER layer-name
[HOLD GEOMETRY]
drop-layerset-statement ::=
DROP LAYERSET layerset-name
[HOLD LAYER]
drop-map-statement ::=
DROP MAP map-name
[HOLD LAYER]
drop-mapbase-statement ::=
DROP MAPBASE mapbase-name
[HOLD LAYERSET]
insert-layer-into-layerset-statement ::=
INSERT LAYER layer-name
INTO LAYERSET layerset-name
insert-layer-into-map-statement ::=
INSERT LAYER layer-name
INTO MAP map-name
insert-layerset-into-statement ::=
INSERT LAYERSET layerset-name
INTO mapbase-name
insert-map-into-statement ::=
INSERT MAP map-name
INTO mapbase-name

lookup-layer-statement ::=
LOOKUP LAYER layer-name
lookup-layerset-statement ::=
LOOKUP LAYERSET layerset-name
lookup-map-statement ::=
LOOKUP MAP map-name
lookup-mapbase-statement ::=
LOOKUP MAPBASE mapbase-name
lookup-sdb-statement ::=
LOOKUP SDB sdb-name
rename-layer-statement ::=
RENAME LAYER layer-name layer-name
rename-layerset-statement ::=
RENAME LAYERSET layerset-name layerset-name
rename-map-statement ::=
RENAME MAP map-name map-name
rename-mapbase-statement ::=
RENAME MAPBASE mapbase-name mapbase-name
select-statement ::=
SELECT [ALL | DISTINCT] select-list
FROM table-reference-list
[WHERE search-condition]
[group-by-clause]
[HAVING search-condition]
[order-by-clause]
[DISPORIGIN x, y] [WINDOW w, h] [DISPSCALE scale_denominator]
statement ::= check-coherence-on-layer-statement
| check-coherence-on-layerset-statement
| check-coherence-on-map-statement
| check-coherence-on-mapbase-statement
| create-layer-statement
| create-layerset-statement
| create-map-statement
| create-mapbase-statement
| delete-layer-from-layerset-statement
| delete-layer-from-map-statement
| delete-layerset-from-statement
| delete-map-from-statement
| drop-layer-statement
| drop-layerset-statement
| drop-map-statement
| drop-mapbase-statement
| insert-layer-into-layerset-statement
| insert-layer-into-map-statement
| insert-layerset-into-statement

| insert-map-into-statement
| lookup-layer-statement
| lookup-layerset-statement
| lookup-map-statement
| lookup-mapbase-statement
| lookup-sdb-statement
| rename-layer-statement
| rename-layerset-statement
| rename -map-statement
| rename -mapbase-statement
| select-statement

The BNF representation of elements in the statement SELECT (omitting the same part as ODBC 2.0 SQL [11]) is listed in the following:

between-predicate ::= expression [NOT]BETWEEN expression AND expression
boolean-factor ::= [NOT]boolean-primary
boolean-primary ::= predicate | (search-condition)
boolean-term ::= boolean-factor[AND boolean-term]
comparison-predicate ::= expression comparison-operator expression
comparison-operator ::= < | > | <= | >= | = | <>
exists-predicate ::= EXISTS(sub-query)
expression ::= term | expression { + | - }
binary-function ::= binary-functor(expression, expression)
binary-functor ::= GEODistance | GEOLeft | GEOMinus | GEORight | GEOUnion | GEOOverlap | GEOOverlay
factor ::= [+ | -]primary
in-predicate ::= expression [NOT]IN { (value [,value]...) | (sub-query) }
like-predicate ::= expression [NOT] LIKE pattern-value
unary-function ::=unary-functor(expression)
unary-functor ::= GEOArcs | GEOArea | GEOBuffer | GEOCentroid | GEOLength | GEOMER | GEOPerimeter | GEOSlope | GEOXcoor | GEOYcoor
null-predicate ::= column-name IS [NOT] NULL
predicate ::= between-predicate | comparison-predicate | exists-predicate
| in-predicate | like-predicate | null-predicate | quantified-predicate
| toporelation-predicate | toporelation-function
primary ::= column-name | literal | USER | set-function-reference
| (expression) | spatial-function-reference
quantified-predicate ::= expression { comparison-operator | toporelation-operator } {ALL | ANY }(sub-query)
search-condition ::= boolean-term[OR search-condition]
select-list ::= * | select-sublist [, select-sublist]
select-sublist ::= expression [[AS]column-alias] | {table-name | correlation-name }.*
spatial-function-reference ::= unary-function | binary-function
sub-query ::=

```
SELECT [ ALL| DISTINCT ] select-list
FROM   table-reference-list
[ WHERE   search-condition ]
[ GROUP BY   column-name,[column-name ]... ]
[ HAVING search-condition ]
table-reference ::= table-name[ correlation-name ]
table-reference-list ::= table-reference[ , table-reference ]..
term ::= factor | term { * | / }
toporelation-function ::= toporelation-inbuffer
toporelation-inbuffer ::= GEOInBuffer( expression, expression, exact-numeric-literal )
toporelation-predicate ::= expression toporalation-operator expression
toporelation-operator ::= ADJACENT | CONTAIN | WITHIN | DISJOINT | EQUAL | OVERLAP
```

The G/SQL statement is classified into four classes: data definition statement, data updating statement, consistency check statement and data query statement. The data definition statement can create, delete and rename a layer, a layerset, a map and a mapbase. The data updating statement includes the insert and the deletion of a layer on a map or a layerset, the insert and the deletion of a map and a layerset on a mapbase, and the updating of information of a layer, a map, a layerset and a mapbase. The consistency check statement is the consistent check on a map, a layerset and a mapbase. The data query statement will comprehensively query the spatial features on a layer or a layerset, and query all layers, layersets, maps and mapbases in the spatial database.

SELECT statement

Abstract spatial data type (Geoid)

Geoid is the unique identity of a spatial feature. It links the attribute in the relational table and the geometry in the file system of the feature together. No matter what type a geometry holds, it always has a unique Geoid. Therefore the Geoid can be regarded as an abstract data type. Every layer table has one and only one Geoid column. The spatial functors will take the operations on Geoids.

WebGIS has the following definitions and mechanism based on the Geoids:

The storage, grammar and semantics of any concrete type of geometry and the spatial index mechanism;

A set of spatial functors whose variable is the Geoid and the grammars and usage mechanism for query when a spatial functor applies;

A function base for realization of spatial operations.

Statement classification and query Results

In a SELECT statement, the FROM sub-statement can appear at the layer level, denoting a query on the layer. The expression with spatial

functors can be not only in the WHERE sub-statement and HAVING conditional expression, also in the object list in the SELECT sub-statement, denoting the spatial operations on spatial features and the selections on spatial relations in a layer.

A query statement can be classified into 4 classes, showing different query results respectively:

A SELECT sub-statement contains neither Geoids nor spatial functions, while WHERE and HAVING sub-statements do not contain any spatial predicate so they are actually the common SQL statement. The result of the SELECT generates a common relational table.

A SELECT sub-statement contains Geoid, but does not contain spatial functions. The WHERE and HAVING sub-statements do not have any spatial predicate. Then the result of the SELECT is a layer.

A SELECT sub-statement does not contain GEOID and spatial functions generating spatial objects, but it contains spatial measurement functions. The WHERE and HAVING sub-statements contain some spatial predicates. Then the SELECT statement returns a common relational table.

A SELECT sub-statement contains GEOID or spatial functions generating spatial objects, and WHERE, HAVING sub-statements contain spatial predicates also. Then the SELECT returns a layer.

According to the above list, we can find out that if a SELECT statement contains neither GEOID nor spatial functions that generate spatial objects, then it returns a common relational table, otherwise it returns a temporary layer. This layer has a certain life cycle in the system: when expiring some time it will be erased automatically by the system. The system ensures the uniqueness of the layer name by naming rules for a temporary layer. The server will send the results as well as the temporary layer name back to the front end. The front end can query, rename and delete the temporary layer.

The following example is query of the fourth class that searches the hotels within 50 kilometers away from the Beijing railway station:

```
SELECT  building.name,  building.GEOID
FROM  building X, building Y
WHERE  X.type = 4  AND  Y.type = 3
AND Y.name = "北京西站"
AND X.GEOID OVERLAP Y.GEOID
```

In the above example we suppose the type of hotels is 4 and the type of Beijing railway station is 3.

Display parameters

There are three optional sub-statements in the query language: DISPORIGIN, WINDOW and DISPSCALE. They are designed for assignment of display parameters when the result is a layer. DISPORGIN shows the absolute geographic coordinates of the window origin; WINDOW shows

the height and width of the window. DISPSCALE shows the denominator of the display scale. The merit of providing the display parameters is helpful to reduce the data flow under the client/server structure so as to accelerate the speed at the client end. This is specially designed for WebGIS.

There are four procedures from the request sending by a client to display of the result in the navigator at the client end:

Query: query the spatial features that meet the condition and generate the temporary layer.

Generate: determine the final display elements according to the display parameters and the display model parameters of every spatial feature. For example, the same spatial feature could hold different display modes under different scales. The filling mode at a large scale is not necessary possibly at a small scale. At the same time, a layer has its own color and font perhaps. Thus in the generation stage the system has to obtain the concrete font and color.

Submit: according to the drawing information of every feature like line style, filling, symbol and color in the temporary layer, the system puts the display forms mentioned above into the inner memory display element table, or forms GIF or other raster images. In most cases it is difficult to differentiate the submission stage with the generation stage.

Display: display the results which are formed at the submission stage through the interface provided by the hardware and/or of the basic graph display.

When the server returns a query result, there are three options: return the result of the submission stage; return the result of the generation stage, and return the temporary layer. The requirement to the client end increases gradually on these three stages. If the server returns GIF image to the client end, then the client can browse the image in the navigator directly, without any development for display at the client end. Many WEB applications adopt this method to send the image data. However, in Web GIS, when a user is browsing the query result, he is not satisfied at the image on the screen. They wish when the mouse move to some building, there is an attribute window that can highlight the building's attribute row. Or when a client clicks on an administrative region, this region could change into a remarkable filling mode. Another requirement of the client could be, when he points on the Beijing railway station from the attribute window, the small dot on the map can be enlarged into a big red dot, representing the station. Such these operations frequently happen and are reasonable. If all these simple operations needs server to process, then the system speed is not endurable. Therefore, there should be certain capabilities of processing GIS data at the client end. Therefore it is the most suitable that the server sends back the spatial data when the request is processed.

On the other hand, when a user is roaming or zooming a map, the screen can only show a part of

map. At this case, it will be wise if the JAVA Applet provides the current display parameters, and asks the server to return the part of data only, which are clipped for the display area by the server on the results of the query.

Compiling and executing G/SQL

There are four steps in order to compile and execute a G/SQL:

Analyze the G/SQL statement. At this step, the compiler will decompose the statement into words and symbols independently to ensure that the statement has an effective order, operators and conjunctions. The compiler will detect the grammar and spelling errors.

Verify the statement. At this step the compiler will check the system table to verify: if the table used by the layer tables exists in the spatial database? if there exist layer tables? If the parameters of spatial operators meet the definition of spatial operation? If there exist the geolayers of layers for spatial manipulation and if they are effective? This step can detect the semantic error furthermore.

Generate a atomic statement sequence. The atomic statement is an object code that a geo-information machine can recognize and execute. The process includes the optimization of compiling.

Execute the statement: the geo-information machine executes the statement through the atomic statement sequence.

Optimization of compiler

There are two optimizations for the G/SQL to generate a atomic statement sequence: the optimization of executing order and re-use of the middle results. We know that obtaining of data in the relation database is much faster than that of geometric data and spatial manipulations. Therefore, when handling the relational condition and spatial condition whose conjunction is "AND", if we execute the relational selection at first and then execute the spatial selection, then the performance will be remarkably improved. The relational query will select the features that meet this condition, and the following condition need only to select the geometric set that has been filtered. By this way, though the spatial calculation is tremendous, the objects that need to be processed are just a relatively small set, which has been selected by the previous process.

Another rule for the optimization of the compiler is to remember the middle results by the compiler and re-use these results in the later manipulations. For example, a user wants to query the third class polluted factories within 500 meters where the residential area is over 10,000 square meters and to show the buffer zone of the polluted area. This query will generate a temporary layer of the buffer zones around the factories when it executes at first. If a compiler can detect this buffer zone has been in the query object list, then it can re-use the middle results.

Atomic statement directives

The atomic statement sequence is the object codes that a compiler uses the set of source statement directives. The source statement sequence is linear, without control flow statements like branch, circulation or loop.

The atomic statement has two classes including geo-information manipulation source statement and relational atomic statement language. The geo-information atomic statement is corresponded to the spatial functors in G/SQL, and does the analysis, determination of spatial relations and spatial measurement. The relational atomic statement is only related to the relational database, that is, only a SQL statement needs to be executed.

There are other atomic statements including the secondary atomic statement and the return atomic statement. The first statement is used for the assistant process on manipulation of objects in the spatial manipulation atomic statement and the relational atomic statement. These include the verification and matching of the manipulation of objects, the generation, index, process and deletion of the middle objects. The second statement will pack the result of a query. It will clip the result if necessary, generate the data package and finally send back to the client end.

Conclusions and application prospective

This paper proposed the concepts and methods to query spatial data based on OGC. The designed G/SQL can not only provide a series of operations on spatial data, but also check the consistency on different levels of the spatial data in the databases. G/SQL has been implemented in the WebGIS software and has shown its efficiency at both server end client end. The software has applied in several applications for GISs.

GIS applications are walking into thousands of thousands of families as with rapid spreading of Internet and with broad and real-time obtaining of spatial data. The popularization of GIS applications is becoming the engine of propelling the spatial data management and query language. The spatial structural query language will go forward along with development. As a testing of spatial structural query language, G/SQL still has several aspects to be improved, such as, it does not provide the definition mechanism of spatial type and spatial functors. The normalization, interoperability and high efficiency is its important development direction.

References

1.Allen J.F. "Maintaining knowledge about temporal intervals, Communications of the ACM 26(11): 832-843, 1983.

2.Cai M., Keshwani S. and Revesz P.Z. "Parametric Rectangles: A Model for Querying and Animation of Spatiotemporal Databases", In Proceedings of the 7th International Conference on Extending Database Technology, pp.430-444, 2000.

3.Chen C.X. and Zaniolo C. "SQLst: A spatio-temporal data model and query Language", http://citeseer.nj.nec.com/512778.html, 2001.

4.Egenhofer, M. J. and Franzosa, R. "Point-set Topological Spatial Relations", International Journal of Geographic Information Systems 5(2): 161-174, 1991.

5.Egenhofer M.J. "Spatial SQL: A Query and Presentation Language" In IEEE Transactions on Knowledge and Data Engineering Vol.6. No.1. pp.86-95, 1994

6.Fang Y., Chen B. and Xue W.W. "The theory and implementation of open GIS application platform", Journal of China image and graph, 1998.

7.Fang Y. "General introduction of digital earth", China computer world, 1999.

8.OGC, "The OpenGIS abstract specification", http://www.opengis.org, 1996

9.OGC, "The OpenGIS simple feature specification for SQL", http://www. opengis.org, 1998.

10.Oosterom, P. and Masessen, B. "Geographic query tool", *JEC-GI '97*, Vol. 1 pp. 177-186, Vienna, Austria, 1997.

11."SQLspecifications" http://www.oracle.com.,

12.Worboys, M.F. "A unified model for spatial and temporal information", The Computer Journal, Vol 37, No. 1, pp. 26-34, 1994.