Storing and Manipulating Simple and Complex Features in Database Management Systems

Peter van Oosterom[1], Edward Verbree[1] and Aegidius Kap[2]
Section GIS Technology[1] - Section GeoInformation and Land Development[2]
Department of Geodesy, Faculty of Civil Engineering and Geosciences
Delft University of Technology, Thijsseweg 11, 2629 JA Delft
The Netherlands.
e-mail: oosterom@geo.tudelft.nl, verbree@geo.tudelft.nl, kap@geo.tudelft.nl

Abstract

The architecture of Geographic Information Systems (GISs) is changing: more and more the systems are based on the integrated architecture [21], that is also storing geometric data in the data base management system (DBMS) together with the other administrative data. The first step is having data types and operators for the geometric primitives: point, line, and polygon. This has reached the level of standardization and is now implemented in several commercial DBMSs. The next step is also having support for the topologically structured features in the DBMS, that is complex features. The DBMS can check and guarantee consistency and complex operations can be executed within the DBMS. The rule of thumb whether something belongs to the DBMS tasks or to a specific application is: whenever it concerns general and reusable aspects then this belongs to the DBMS. Despite the fact that the topology model are well know, it still remains an open issue how to implement these models completely within a relational DBMS.

Background

This integrated architecture can be contrasted to traditional approaches such as: the dual architecture (separate data base management systems for geometric and administrative data) and the layered architecture (all data stored in a single DBMS, but spatial knowledge is contained in a layer between the application and the DBMS, some examples are ERSI's SDE, Oracle's SDO). In the integrated architecture the DBMS is extended with spatial data types (point, polyline, polygon) and function (overlap, distance, area, length). The first DBMSs offering these capabilities were experimental systems, such as Postgres [18], O2, Gral [6], and others [4] and, of course, the functionality was not yet standardized in SQL92 [22, 8]. Immediately, also the first GISs based on the spatially extended DBMSs became available based on either an extended (object) relational database (GEO++) or on a pure object oriented database (GeO2). The importance of the integrated architecture was recognized by industry and the OpenGIS consortium [2] standardized the basic spatial types and functions, or in the OpenGIS terminology the Simple Feature Specification (SFS). The implementation specification for the SFS are described for three different platforms: SQL [14], Corba, and OLE/COM. The SQL/SFS implementation specification will also be part of the future ISO SQL3 standard [10]. In 1999 the first implementations of the OpenGIS SQL/SFS became available, which marked an important step forward in the maturing of GIS. It should be noted that the main attention is on 2D spatial data types, but also 3D spatial data types are used. There is an OpenGIS proposal to introduce 'z' values in the simple feature co-ordinates [5] but further research will be necessary to investigate whether the 2D datatype can be extended with an extra dimension, or that total reconsideration will be required.

Complex features

Having spatial types and operators is one part of the DBMS services required by a GIS. The other components are: 1. spatial indexing (quadtree, r-tree [7, 16]) and spatial clustering and 2. representing and manipulating complex features. Complex features can used to represent planar partitions without redundancy. Another possible use of a complex feature is representing a linear network. In this paper

we will focus on the topology structure for a planar partition. Topology structures for planar partitions are well know for a long time. They are used in e.g. TIGER [1, 11], DIME [19], the Arc/Info system [13], the Netherlands Cadastre LKI [20], and many other systems. They are also studied a lot, e.g. by Molenaar [12, 3]. In this paper the Cadastral map will be used as a case study, because topology plays a key role in this spatial data set. The current implementation in the Netherlands will be described as it is based on a relational DBMS with spatial extensions. Relational DBMSs can very well store the topology references: area left and right of a boundary, boundary to boundary references, treatment of islands, etc.. That is, the modeling aspect of topology. However, they do not support this in the sense of a complex features, that is, being able to do consistency checks (area closed?, topology references correct?) and operations. The operators can be the basic edit operations (split or merge area features by inserting or removing boundary features), but can also be complex operators such as map-overlay or compute the perimeter and area of a topologically represented area or solving the question: which areas are crossed by this query polyline? (or route planning in a linear network represented by a complex feature). The problem with standard relational DBMS is that the declarative language SQL can not handle the 'transitive closure', which is needed for the functionality described above. The unknown number of steps (references) needed to find a closed boundary loop (or path in a linear network) forms the problem. Of course, it is very simple in a functional programming language using one of the basic iterator concepts.

Discussion

In an implementation not all functionality has to be provided by the DBMS. It is possible to provide part of the functionality in a front-end application. This enables the implementation to be based on standard tools without modifying the relational DBMS (server). However, as the support for complex features is quite generic, it should optimally be in the DBMS. This avoids reimplementation of the same functionality in several applications and it also the best guarantee for consistency control. Further, it also allows analysis queries on topologically structured features to be executed within the DBMS. So, no unnecessary data transfer to a front-end applications takes place. Currently, the OODBMSs do seem to offer the most flexible platform for implementing the complex features. Their relatively weak acceptance by the market, the lack of a standard query language and the fuzzy boundary form the motivation to try include support for complex features in (extended/object) relational DBMSs.

Since the subject of implementing complex features in a DBMS is a fairly empty field, variable issues will have to be taken into account. Questions that arise are if it is possible to implement a fully operational solution, that covers the total domain of all possible complex elements. Since we will firstly focus on a subset of the complex features - the topological structures - the question will also be how far this solution will solve the total domain, and how this subset can be used as an basis to extend to other complex data types.

The chosen solution of implementing the topology in the database will have to be tested to examine whether the solution has a real benefit above other scenarios, where this functionality is implemented in the front end or middle-ware application, or distributed over the DBMS, the middle-ware and the front end. Also we will have to keep in mind that a solution must have a possibility to be extended. Users must have the freedom to add on their own applications, without creating conflicts with the database.

Standards

In this paper we will analyze the latest development with respect to complex spatial features in the international standards (ISO and OpenGIS). Having a standard is one important step, but the implementation forms the next required step in practice. The OpenGIS Consortium mentions topology several times in its abstract specifications (AS). They enhanced the abstract geometric and topology description a lot in version 4 of the AS1: Feature Geometry [15]. This abstract specification is very similar, and more or less at the same abstraction level as the work of the ISO TC 211 [9]. These two standards are quite harmonized and they both make a difference between geometric primitives and topology. What still is missing is the implementation specification of topology (or sometimes also called complex features) for specific platform comparable to the implementation specification for simple features. The question remains whether this would be possible with a relational DBMS using

SQL (due to the 'transitive closure' problem). Other topology related OpenGIS abstract specifications are AS5: Features (persistent object identifiers) and AS8: Relationship between Features [15].

An attempt to extend SQL

For the modeling part of topology and one important operator, map-overlay, an attempt to extend the (post)quel (a language similar to SQL, used in the Postgres DBMS) was described [17]. It was suggested to use 'prototypes' to define a topological layer:

```
define prototype faces (id=oid, boundary=edges.id[])

define prototype edges (id=oid, line=POLYLINE2, left=faces.id, right=faces.id)

create layers(layer_id=unique text, boundaries=prototype edges, areas=prototype faces)

define topology on layers using wheel_topology (boundaries, areas)
```

There are several ways to implement topology, the method described above is based on edges and faces (no explicit nodes). There are references from a face to all its boundaries (exterior and possible also interior) and there are references from a boundary to the left and right face. This type of topology is sometimes called, wheel topology. Other topology implementations (with other restrictions) are possible, but can be based on different prototypes. These prototype structures can now be used to create actual map layers (that is instances of map layers with edges and faces at least having the attributes specified in the prototypes):

```
define prototype faces2 (name=text, owner=text, value=int4)
        inherit faces

append layers (layer_id="parcels", areas=prototype faces2,
        boundaries=prototype edges)

append l1.boundaries (polyline="(....)"::POLYLINE2)
        from l1 in layers where l1.layer_id="parcels"

replace l1.areas (name="....", value="....", owner="....")
        from l1 in layers where PointInPolygon ("(x,y)"::POINT2, current))
        and l1.layer_id="parcels"
```

It is assumed that the appends and the replaces are correct and consistent with the topology rules. Otherwise a transaction can not be committed. This is checked by the DBMS as a result of the 'define topology' statement. After this first layer "parcels", a second layer "soils" can be created in a similar manner. Having created two layers, it is now possible to perform the complex map-overlay operation within the DBMS:

```
append layers (layer_id="combined layer")

retrieve (count=overlay(l1,l2,new_layer,"FaceAttrSpecStr", epsilon, sliver))
        from l1, l2, new_layer in layers
        where l1.layer_id="parcels" and l2.layer_id="soil" and
        new_layer.layer_id="combined layer"
```

The DBMS must be extended to support this overlay operation, which returns the number of faces as result and as a side effect is able the compute the new layer as overlay of the input layers. Note that the support of topology is more or less the same level in the DBMS as the support of indices or referential integrity constraints. We will try to implement and test the described approach in a SQL environment.

DBMS meta information

The meta information (or system catalog) of a DBMS contains descriptions of the data stored in the database: tables, attributes and types, and also contains descriptions of the available types and operators. This enables dynamic SQL applications. In case the (relational) DBMS has to support topology, somewhere the structural knowledge has to be stored (and be accessible for applications): e.g. topology layer name, which table plays the role of the boundary table, which table plays the role of

the area table and how are the relevant attributes, which metrical and topological information called within these tables. One solution for this problem is providing prototypes as a basis for the possible complex structures. The topology elements (object ids, references and also the metric attributes) have fixed names.

An alternative is a topology implementation at the front-end application level. Again, somewhere it must be declared which tables and which attributes carry the topology information. An example of the extension of the meta information of the DBMS will be given in the context of GEO++ (dyn_info table).

The Oracle 8i spatial DBMS will be used in analyzing (and implementing) complex spatial features. It is described how the relational model is extended (PL/SQL, stored procedures) to support complex spatial features. In this context it is not right to talk about a relational DBMS, or extended relational DBMS, but it is an object relational DBMS. We will also investigate the use of a pure object-oriented DBMS (such as Jasmine) and compare the different approaches.

References

[1] Gerard Boudriault. Topology in the TIGER file. In AutoCarto 8, pages 258-269, 1987.

[2] Kurt Buehler and Lance McKee. The OpenGIS guide - introduction to interoperable geoprocessing. Technical Report Third edition, The Open GIS Consortium, Inc., June 1998.

[3] Sylvia de Hoop, Peter van Oosterom, and Martien Molenaar. Topological querying of multiple map layers. In COSIT'93, Elba Island, Italy, pages 139-157, Berlin, September 1993. Springer-Verlag.

[4] David J. DeWitt, Philippe Futtersack, David Maier, and Fernando Velez. A study of three alternative workstation-server architectures for object oriented database systems. Technical Report 907, Computer Sciences Department, University of Wisconsin-Madison, January 1990.

[5] Adam Gawne-Cain. Z values in simple feature co-ordinates. Technical Report (99-402.doc), OGC, Open GIS Consortium, Inc., July 1999.

[6] Ralf Hartmut Güting. Gral: An extensible relational database system for geometric applications. In Proceedings of the Fifteenth International Conference on Very Large Data Bases, Amsterdam, pages 33-44, New York, August 1989.

[7] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. ACM SIGMOD, 13:47-57, 1984.

[8] ISO. Database language SQL. Technical Report Document ISO/IEC 9075, International Organization for Standardization, 1992.

[9] ISO TC 211. Geographic information - Spatial schema. Technical Report second draft of ISO 19107, International Organization for Standardization, November 1999.

[10] ISO/IEC JTC1/SC21/WG3. SQL 3/MM spatial. Technical Report Part 3, International Organization for Standardization, 1996.

[11] Christine Kinnear. The TIGER structure. In Auto-Carto 8, pages 249-257, 1987.

[12] Martien Molenaar. An Introduction to the theory of spatial object Modelling for GIS, Taylor and Francis, 1998.

[13] Scott Morehouse. The architecture of Arc/Info. In AutoCarto 9, pages 266-277, April 1989.

[14] Open GIS Consortium, Inc. OpenGIS simple features specification for SQL. Technical Report Revision 1.1, OGC, May 1999.

[15] Open GIS Consortium, Inc. The OpenGIS abstract specification, http://www.opengis.org/techno/specs.htm

[16] Hanan Samet. The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading, Mass., 1989.

[17] Vincent Schenkelaars and Peter van Oosterom. Map-overlay within a geographic interaction language. In Auto-Carto 12, Charlotte NC, pages 281-290, 27 February-1 March 1995.

[18] Michael Stonebraker, Lawrence A. Rowe, and Michael Hirohama. The implementation of Postgres. IEEE Transactions on Knowledge and Data Engineering, 2(1):125-142, March 1990.

[19] US Bureau of the Census. The DIME geocoding system. Technical Report 4, Census Use Study, US Department of Commerce, Bureau of the Census, Washington, DC, 1970.

[20] Peter van Oosterom. Maintaining consistent topology including historical data in a large spatial database. In Auto-Carto 13, pages 327-336, April 1997.

[21] Tom Vijlbrief and Peter van Oosterom. The GEO++ system: An extensible GIS. In Proceedings of the 5th International Symposium on Spatial Data Handling, Charleston, South Carolina, pages 40-50, August 1992.

[22] C.J. Date with Hugh Darwen. A Guide to the SQL standard, 4th edition. Addison Wesley, 1997.