GIS ANALYSIS USING DIFFERENT SPATIAL DATABASES

Martínez-Llario, José
Coll, Eloina

Abstract - The use of a spatial database to store geographic objects and perform spatial analysis operations is a task that can get to be complex for a traditional user of Cartography or Geographic Information Systems (GIS). In this paper a practical small example of vectorial spatial analysis is presented. This analysis is carried out in the first place using SQL language and spatial extensions included in PostGIS (an open source backend spatial database). Next, the same example is solved with Oracle Spatial. After a brief introduction of the available cartography it will proceed to look for the good location to locate a certain example infrastructure. The first phase will be to load the cartographic data in both systems, and next to carry out the necessary analysis operations until obtaining the looked for surface. The problem will be developed using the corresponding necessary SQL sentences for each one of the spatial databases.

PhD. Jose Martinez-Llario[1], PhD. Eloina Coll[2]
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022, Valencia, Spain
Tef.: +34 96 3877550, Fax: +34 96 387759, email: {jomarlla@cgf.upv.es[1], ecoll@cgf.upv.es[2]}

1. INTRODUCTION

Traditionally the spatial analysis operations (raster and vectorial) has been made in a local way using a GIS desktop software. Already some years ago the most important databases in the market have incorporated spatial extensions, mainly with the impulse of the organization Open GeoSpatial Consortium (OGC) leaning on in the specification 'Simple Features for SQL' [5]. Deepening in this specification, they approach us a series of questions that this article we will try to solve and to outline some of them for a possible debate:

- Could it delegate himself in the system agent of databases, the characteristics to carry out the spatial analysis it needs a Geographical Information System?
- But, does it contribute some benefit that the client delegates the vectorial analysis in the server?
- The specification SFS defines in a sufficiently wide way the features that a DBMS should implement to be able to make a complete vectorial spatial analysis?
- Does the current computer GIS software use these characteristics?
- Where the specification SFS should evolve? Does it exist some normative one or more current alternative?

This paper tries to describe the most important tools they exist at the moment in the market, what characteristics they have and how they should be used to carry out a vectorial spatial analysis. For it, two spatial databases, a commercial one and another free one have been selected. PostGIS [2] (based on PostgreSQL, version 1.0.4) is the most advanced alternative of free software, on the other hand the commercial software Oracle Spatial has been used [1] (version 10.2).

In the first part of the paper it is described a simple vectorial analysis and the cartography needed shortly. Next, analysis is solved using these two spatial DBMS. In a second part, the more important characteristics to keep in mind to carry out the analysis are emphasized, as well as the fidelity of these software with the SFS specification. The article concludes with some conclusions to try to answer the questions enunciated previously.

2. PROPOSED SPATIAL ANALYSIS

The objective of this spatial analysis is to show in a practical way how it is carried out using two different programs and if it is possible to carry out this analysis following in a rigorous way the SFS specification. In any case it has wanted to carry out calculations of performance of the used algorithms. Therefore, all the used cartographic layers are very small and they have less than one hundred of geometric entities.

2.1. Spatial analysis conditions

Localization of a certain infrastructure with following approaches:

- Zonal approach (layer ‘suelos’: tsuelo = 300)
- Zonal approach (layer ‘usos’: tuso > 0)
- Vicinity (layer ‘alcanta’: < 300 meters)
- Vicinity (layer ‘rios’: > 20 meters (trio = 1) > 40 meters (trio = 2))
- Final area > 5 000 m^2

2.2. Cartography

- Layer ‘suelos’ (Figure 1). Polygon layer formed by 43 entities describing types of soils. Scheme: (shape: polygon, tsuelo: short int {0,1,2,3})
- Layer ‘usos’ (Figure 2). Polygon’s layer formed by 76 entities describing land-use soils. Scheme: (shape: polygon, tuso: short int {100,200,300,400,500,600,700})
- Layer ‘rios’ (Figure 3). Line’s layer formed by 106 entities describing rivers. Scheme (shape: line, trio: short int {1,2})
- Layer ‘alcanta’ (Figure 4). Line’s layer formed by 6 entities describing sewer system net. Scheme: (shape: line, id: short int {0}).

Figure 1: Soils type layer

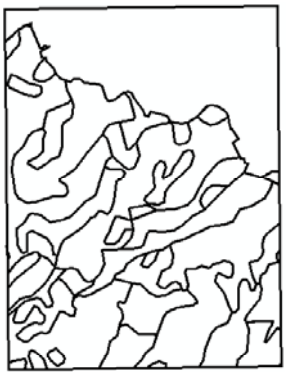

Figure 2: Land-use soils layer

Figure 3: Rivers layer

Figure 4: Sewers layer

This cartography is debugged so that repeated vertexes, polygons with overlaps or close vertexes under tolerance specified in the analysis (Oracle Spatial) don't exist. The features must be valid according to OGC.

3. SPATIAL ANALYSIS

3.1. Cartography load with PostGIS

Import of the cartography can be carried out using the PostGIS command ‘shp2pgsql’:

```
#!/bin/bash

#Files:
#lwpostgis.sql
#spatial_ref_sys.sql
#rios.shp ,rios.dbf, rios.shx
#suelos.shp, suelos.dbf, suelos.shx
#usos.shp, usos.dbf, usos.shx
#alcanta.shp, alcanta.dbf, alcanta.shx

createdb test
createlang plpgsql test #
psql -f lwpostgis.sql -d test #PostGIS
psql -f spatial_ref_sys.sql -d test #EPSG

#Shapefiles to SQL
shp2pgsql rios.shp rios > rios.sql
shp2pgsql suelos.shp suelos > suelos.sql
shp2pgsql usos.shp usos > usos.sql
shp2pgsql alcanta.shp alcanta > alcanta.sql

#Layers load
psql -d test -f rios.sql
psql -d test -f alcanta.sql
psql -d test -f suelos.sql
psql -d test -f usos.sql
```

3.2. PostGIS resolution

In the following listing appears the SQL sentence of the analysis. Those sentences didn't considered in SFS but necessary to carry out the analysis are in bold. To simplify the code of the example, it has been substituted the expressions of geometry creation tables by the sentence 'createTable (type, name)'

```
createTable (POLYGON, tmp1) =
        create table tmp1 (gid serial);
        select addgeometrycolumn
        ('','tmp1','the_geom',-1,'POLYGON',2);
        alter table only tmp1 add constraint tmp1_pkey primary key (gid);
```

'alcanta' buffer *createTable (POLYGON, tmp1)* *1:* insert into tmp1(the_geom) select **buffer(the_geom,300,32)** from alcanta; *createTable (POLYGON, alcantabuf)* *2:* insert into alcantabuf(the_geom) select **geomunion(the_geom)** from tmp1; *'rios' buffer* *3:* create table riodist (trio integer primary key,dist float); *4:* insert into riodist values (1,40); *5:* insert into riodist values (2,20); *createTable (POLYGON, tmp2)* *6:* insert into tmp2(the_geom) select **buffer(r.the_geom,d.dist,32)** from rios as r, riodist as d where r.trio = d.trio; *createTable (MULTIPOLYGON, riosbuf)* *7:* insert into riosbuf(the_geom) select **geomunion (the_geom)** from tmp2; *'suelos' and 'alcanta' intersection* *8:* create index suelos_the_geom_idx ON suelos USING GIST (the_geom GIST_GEOMETRY_OPS); *9:* vacuum analyze suelos (the_geom); *10:* create index usos_the_geom_idx ON usos USING GIST (the_geom GIST_GEOMETRY_OPS); *11:* vacuum analyze usos (the_geom);	*createTable (MULTIPOLYGON, inter)* *12:* insert into inter (tuso,tsuelo,the_geom) select u.tuso, s.tsuelo, **multi(**intersection (u.the_geom,s.the_geom)**)** from usos as u, suelos as s where u.the_geom && s.the_geom and intersects (u.the_geom,s.the_geom); *13:* create view vinter as select i.gid as gid,i.the_geom as the_geom from inter as i where i.tuso = 300 and i.tsuelo > 0; *Buffer difference* *createTable (MULTIPOLYGON, difbuf)* *14:* insert into difbuf (the_geom) select **multi(**difference (c1.the_geom,c2.the_geom)**)** from alcantabuf as c1, riosbuf as c2 where intersects (c1.the_geom,c2.the_geom); *createTable (MULTIPOLYGON, final)* 15: create index inter_the_geom_idx on inter using gist (the_geom GIST_GEOMETRY_OPS); 16: vacuum analyze inter (the_geom); 17: create index difbuf_the_geom_idx on difbuf using gist (the_geom GIST_GEOMETRY_OPS); 18: vacuum analyze difbuf (the_geom); 19: insert into final (the_geom) select **multi(**intersection (v.the_geom,b.the_geom)**)** from vinter as v,difbuf as b where v.the_geom && b.the_geom and intersects (v.the_geom,b.the_geom); *Final result* 20: select gid, area(the_geom) as area from final order by area desc;

Table 1: SQL sentences of PostGIS analysis

3.3. Cartography load with Oracle Spatial

Import of the cartography can be carried out using the Oracle Spatial command 'shp2sdo':

```
#Shapefiles to SQL
shp2sdo suelos suelos –g Geom –d –x (0,10000) –y (0,10000) –t 0.000001 –v –i gid
shp2sdo usos usos –g Geom –d –x (0,10000) –y (0,10000) –t 0.000001 –v –i gid
shp2sdo rios rios –g Geom –d –x (0,10000) –y (0,10000) –t 0.000001 –v –i gid
shp2sdo alcanta alcanta –g Geom –d –x (0,10000) –y (0,10000) –t 0.000001 –v –i gid

#SQL scheme (isqlplus)
start suelos.sql
start usos.sql
start rios.sql
start alcanta.sql
```

```
#Layers load (sqlloader)
Sqlldr user/password suelos
Sqlldr user/password usos
Sqlldr user/password rios
Sqlldr user/password alcanta
```

3.4. Oracle Spatial resolution

In the following listing appears the SQL sentence of the analysis. To simplify the code of the example, it has been substituted the expressions of geometry creation tables by the sentence 'createTable (type, name)'

```
createTable (tmp1) =
        CREATE TABLE TMP1 (
          GID NUMBER PRIMARY KEY,
          GEOM MDSYS.SDO_GEOMETRY);

        INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
            VALUES ('TMP1','GEOM',
               MDSYS.SDO_DIM_ARRAY
                (MDSYS.SDO_DIM_ELEMENT('X', 0.000000000, 10000.000000000, 0.000001000),
                 MDSYS.SDO_DIM_ELEMENT('Y', 0.000000000, 10000.000000000, 0.000001000)),
               NULL);
```

'alcanta' Buffer

createTable (tmp1)

1: insert into tmp1 (gid,geom) select a.gid, SDO_GEOM.SDO_ARC_DENSIFY (sdo_geom.sdo_buffer (a.geom,300,0.000001),0.000001, 'arc_tolerance=1') from alcanta a;

createTable (alcantabuf)

2: insert into alcantabuf (gid,geom) select 1, SDO_AGGR_UNION(
  SDOAGGRTYPE(a.geom, 0.000001)) from tmp1 a;

'rios' Buffer

3: create table riodist (trio number(1) primary key,dist number(2));
4: insert into riodist values (1,40);
5: insert into riodist values (2,20);

createTable (tmp2)

6: insert into tmp2(gid,geom) select r.gid, SDO_GEOM.SDO_ARC_DENSIFY (sdo_geom.sdo_buffer(r.geom,d.dist, 0.000001),0.000001,'arc_tolerance=1') from rios r, riodist d where r.trio = d.trio;

createTable (riosbuf)

7: insert into riosbuf (gid,geom) select 1, SDO_AGGR_UNION(SDOAGGRTYPE (a.geom, 0.000001)) from tmp2 a;

'suelos' and 'alcanta' intersection

8: CREATE INDEX suelos_idx ON suelos (geom)INDEXTYPE IS MDSYS.SPATIAL_INDEX;
9: CREATE INDEX usos_idx ON usos (geom)INDEXTYPE IS MDSYS.SPATIAL_INDEX;

createTable (inter)

10: insert into inter (gid, tsuelo, tuso, geom) select u.gid*1000+s.gid, s.tsuelo, u.tuso, sdo_geom.sdo_intersection (u.geom,s.geom, 0.000001) from usos u, suelos s where SDO_FILTER(u.geom,s.geom) = 'TRUE' and (sdo_geom.relate(u.geom, 'OVERLAPBDYINTERSECT',s.geom,0.000001) = 'OVERLAPBDYINTERSECT' or sdo_geom.relate(u.geom,'INSIDE', s.geom,0.000001) = 'INSIDE');

11: create view vinter as select i.gid gid, i.geom geom from inter i where i.tuso = 300 and i.tsuelo > 0;

Buffer difference

createTable (difbuf)

12: insert into difbuf (gid, geom) select c1.gid, sdo_geom.sdo_difference (c1.geom,c2.geom,0.000001) from alcantabuf c1, riosbuf c2;

createTable (final)

13: CREATE INDEX inter_idx ON inter (geom)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;
14: CREATE INDEX difbuf_idx ON difbuf (geom)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

15: insert into final (gid, geom) select v.gid*1000+b.gid, sdo_geom.sdo_intersection (v.geom,b.geom, 0.000001) from vinter v, difbuf b where SDO_FILTER(v.geom,b.geom) = 'TRUE' and (sdo_geom.relate (v.geom, 'OVERLAPBDYINTERSECT', b.geom,0.000001) = 'OVERLAPBDYINTERSECT' or sdo_geom.relate (v.geom,'COVEREDBY', b.geom,0.000001) = 'COVEREDBY' or sdo_geom.relate (v.geom, 'INSIDE', b.geom,0.000001) = 'INSIDE');

Final result

16: select gid, sdo_geom.sdo_area (geom, 0.000001) area from final order by area desc;

Table 2: SQL sentences of Oracle analysis

4. SYSTEMS FUNCTIONALITIES

4.1. Spatial data import

Both systems have a command for the import of files in shape format (*shp2pgsql* in PostGIS and *shp2sdo* in Oracle), these commands are invoked from the system console and they import as much the spatial data as thematic data (*dbf*).

PostGIS
The import command creates only one file where the necessary SQL sentences are included for the creation of the table and the data load (with the corresponding sentences 'insert') of each one of the features.

Oracle
The import command creates two files; a file (*.sql*) where the necessary SQL sentences are included for the creation of the table and other file (*.ctl* in text format) with the data of the table for their import with the utility *sqlloader* of Oracle. In the parameters of the command *shp2sdo* it should also be specified certain parameters like the extension and the tolerance of each one of the dimensions of the geometries, information that is used in the definition of the creation of the feature table.

4.2. SFS Specification

Although in the documentation of both products it makes sure that they follow the specification SFS of the OGC (passing with success the OGC conformance test: 'Conformance Test Guidelines for Simple OpenGIS Features Specification for SQL' [4]), it is necessary to carry out important shadings in this aspect:

- Oracle Spatial indeed follows the specification but it uses much more the relaxation (changes allowed by the OGC) of the test mentioned previously, that is to say, the methods defined in SFS (geometry manufacturers, operators and spatial predicates,…) have its correspondence in methods of Oracle Spatial but these have different name, number or type of arguments. This is since a great limitation it is needed to translate the SQL code of Oracle if one wants to export to other systems, even although these support in a strict way the specification of the OGC.

- PostGIS follows in a much faithfully way the specification SFS, respecting the names of the methods (except some exception) and the number of arguments. PostGIS also stores the information of columns of geometry metadata of tables (GEOMETRY_COLUMNS) and spatial reference metadata following the standard SFS (SPATIAL_REFERENCE_SYSTEMS).

- Both software not incorporate methods defined by the OGC in their specification, but necessary if they want to be carried out complex analysis or stored procedures that they increase the functionality of the system. As it will be seen but it advances in the case of PostGIS, some of these methods (that don't follow the standard) they are necessary to even carry out the simplest vectorial analysis.

4.3. Spatial tables

In previous enunciated to the tables 1 and 2 corresponding to the spatial analyses, the different form is shown that they have these two systems of creating a geometry table:

- PostGIS uses the method 'addGeometryColumn' (method proposed in the SFS), it takes the responsibility of adding the geometry field and the selected geometry type to a certain table, to upgrade the information of the metadata in the table GEOMETRY_COLUMNS, and to add the table restriction of the geometry type in the corresponding field.

```
SELECT ADDGEOMETRYCOLUMN ('','tmp1','the_geom',-1,'POLYGON',2);
```

- On the contrary Oracle doesn't implement the method 'addGeometryColumn', that is to say, it defines the table directly (also changing the name of the same one proposed by the OGC).

```
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
    VALUES ('TMP1','GEOM', MDSYS.SDO_DIM_ARRAY
        (MDSYS.SDO_DIM_ELEMENT('X', 0.000000000, 10000.000000000, 0.000001000),
         MDSYS.SDO_DIM_ELEMENT('Y', 0.000000000, 10000.000000000, 0.000001000)), NULL);
```

PostGIS limits the type of geometries stored in a table (table restriction on the geometry column, fixing the type of this), although this form of working (a layer contains geometric entities with the same type of geometries) it is but chord with the traditional form, presents certain problems (problems that Oracle Spatial doesn't have since it doesn't establish this type of restrictions in the tables). For example, if in a Polygon table it is tried to store a Multipolygon (or the other way around), the system will give a violation error of a table restriction. This can cause like it will be seen the impossibility later on of carrying out certain analysis operations provided the user is not helped of methods defined in the SFS and that they can in certain way to solve this problem.

4.4. Incompatibility with SFS specification

As for the incompatibilities of methods that don't follow SFS specification, we can establish two groups according to the importance of the incompatibility. A first group would consist on those methods that although they are defined in the specification SFS they don't coincide in name, number or type of admitted arguments or returned (although the relaxation of the test of the OGC allows to change the name to the functions/tables/views). A second group would be formed by DBMS methods that offer a functionality that SFS doesn't. In this second group the incompatibility takes place hopelessly when some of these methods are indispensable to carry out a common spatial analysis, so that it is necessary to use it, and this way the SQL code generated is not compatible with SFS.

4.4.1. PostGIS

According to table 1:
In the first group we would have the sentences that invoke to methods '**`BUFFER`** `(the_geom,300,32) `' and '**`GEOMUNION`** `(the_geom) `'. The incompatibility in the first method is due to the third argument (number of segments of the curvilinear elements) not considered in SFS, while the second has as original name 'union' in SFS and non 'geomunion' (PostGIS renames it for not coinciding with the SQL word key 'union').

PostGIS has near 100 methods that offer a extra spatial functionality, but it is for example the method 'multi', **`MULTI`** `(intersection (u.the_geom,s.the_geom))`, the one that is necessary to use in a common spatial analysis to solve the problem that is introduced next.

<u>Geometries types returned in the spatial intersection operation in PostGIS</u>

Many of the free solutions that use spatial analysis are based on the library JTS [6]. This is the case of PostGIS, it is based on the library GEOS that is a wrapper of JTS in C++. To carry out the software test, JTS Test Builder has been used [5]. In this example we only use the spatial operator 'intersection(A, B) ', this operator returns the intersection among geometries A and B. To simplify the case we will consider that A and B are features of 'POLYGON' type. As you it can appreciate in the figures, the intersection of two polygons can produce a multipolygon (Figure 5) or a geometry collection (Figure 6) formed by points, lines and polygons.

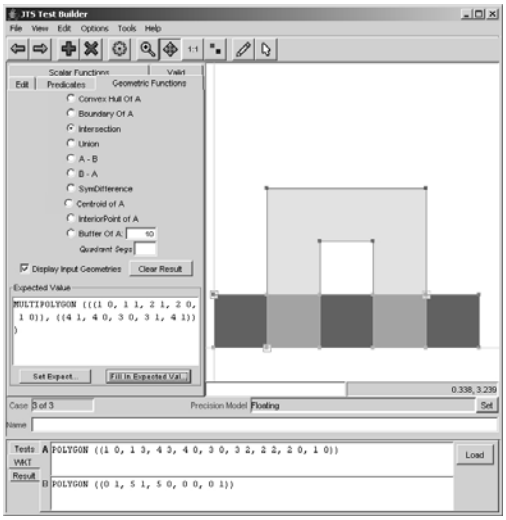

Figure 5: JTS Test Builder Software. Example 1

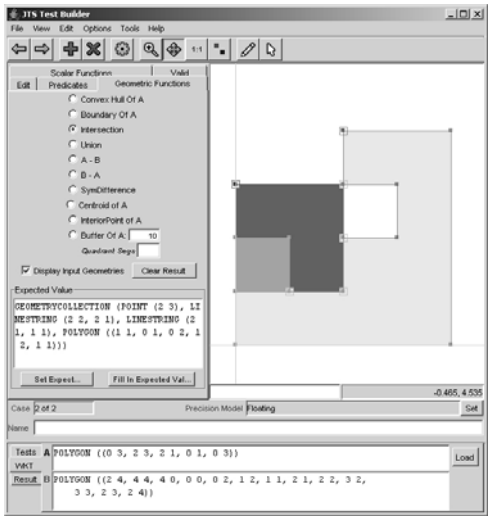

Figure 6: JTS Test Builder Software. Example 2

When using the intersection spatial operator (between polygons) it will be kept in mind the following ones characteristics:

- An intersection of two polygons returns an only geometry.
- If this geometry is formed by several type elements POINT, LINESTRING or POLYGON it will be of type 'multi', that is to say, MULTIPOINT, MULTILINESTRING or MULTIPOLYGON.
- If the result is a mixed geometry of several simple types, it will be of type GEOMETRYCOLLECTION.

These characteristics can cause a **violation in the table restrictions in PostGIS** [3], since PostGIS cannot store in a table different types of geometries. Evidently you can apply a clause WHERE in the corresponding one SQL sentence to filter the type of geometry return of intersection, but this will produce a lost of elements that it can be valid in the final result. The only possible solution is to use the method 'multi' of PostGIS, to convert the entities returned by intersection to entities of type 'multi' (MULTIPOLYGON, MULTIPOINT or MULTILINESTRING), and to store them for example in a layer of multipolygon type. Even this way if one had a case as the one showed in Figure 6, the problem could not be solved unless it will be carried out a program in some server language like **pgplsql** and even more methods will be used not considered in the SFS specification.

4.4.2. Oracle Spatial

As you it can observe in table 2, practically all methods of Oracle Spatial would be framed inside what we have denominated first group of incompatibility. For example, besides renaming the methods, many of them use an additional argument for the tolerance of the coordinates: `SDO_BUFFER (r.geom,d.dist, 0.000001)`, and others use types defined by the system: `SDO_AGGR_UNION (SDOAGGRTYPE(a.geom, 0.000001).`

On the other hand we don't find with one of the problems that prevented PostGIS to follow SFS (command *multi*). indeed, if we notice how a geometry table is defined in Oracle, we realize that the geometry type is not specified, and in a polygon table they can also be stored multipolygons, what eliminates the problem that had PostGIS.

5. CONCLUSIONS

As for the compatibility of the two spatial DBMS studied with the specification SFS of the OGC, PostGIS continues much more faithfully the specification that Oracle Spatial, but this last one is easier of managing when not having to constantly check the types of the geometries returned to avoid errors of violation of the table restrictions. This way, to carry out a common and simple spatial analysis it is needed to use functionalities extra non collected in the specification SFS like the method *multi* of PostGIS.

It is still work to carry out, to be able to delegate the functionality of spatial analysis in the server side at least if one wants to follow the SFS specification. Although opinions exist in favor of the traditional system of making the analysis in local way (in the client), personally we think that the future goes to implement in a good way this functionality in the server, which would contribute many advantages like: maintenance of teams, cost of licenses, increment in the logical separation of the structure client-server in a GIS, to start of a wider common base in the elaboration of a Desktop GIS, etc. all this would rebound in endowing from services of spatial analysis to a GIS client and in bringing near the possibility to create a GIS Web completely operative using standard protocol . The application maybe more spectacular it consists on endowing an SDI of a complete spatial analysis and one of the final objectives is maybe to obtain a professional GIS through the Web.

But all this is not still since reality like it has been commented previously it is work to make (mainly if we speak of standard protocols and free software). indeed, at the moment the GIS software doesn't use these characteristics on spatial analysis in the server side (to exception of some few commercial products, of very high cost, and using non compatible systems with OGC), as a lot, they are able of reading directly of PostGIS (others ones are limited to import or to export the information to PostGIS) and to carry out operations of vectorial analysis in the client (converting the data to an internal format or to the format of the used libraries like JTS).

There are certain spatial functionalities that they can benefit of an explicit topology model if they are correctly implemented, as the influence areas. The explicit topology should be obligatory since besides to avoid redundancy of data and to add connectivity. This explicit topology speed up certain operations of vectorial spatial analysis that could end up being unviable in another way. At the moment free libraries that implement an explicit topology model don't exist although one is already working in libraries like GeoTools. As for PostGIS, at the moment a topology model is implementing based on the ISO 13249 (SQL/MM leaves 3). Although a spatial database with an explicit topology

model benefits in great measure the operations of spatial analysis, it can also harm in way appreciable the reading of a great number of cartographic elements (common operation in the render of cartography map servers) since the entities should be formed starting from the integral topologies every time that an element is requested. In these operations traverse relationships intervene among elements that should be implemented using a object-relational database management system (SQL3) that implements in an effective way these characteristics.

REFERENCES

1. Manuals de Oracle Spatial 10g. http://www.oracle.com (last access: Octubre, 2005)
2. Manual de PostGIS. http://www.postgis.org/ (last access: Noviembre, 2005)
3. Martinez-Llario, J. C., Coll, E. (2005): Spatial Analysis using OpenGIS Specifications: 'Simple Features for SQL'. WSEAS Transactions on Information Science and Applications. ISSN 1790-0832
4. Open GIS Consortium, Inc. (1998): Conformance Test Guidelines for OpenGIS Simple Features Specification for SQL, Rev. 1.0.
5. Open GIS Consortium, Inc. (1998): Simple Features Specification for SQL (SFS). Version 1.1.
6. Vivid Solutions. (2005): JTS Technical Specifications, Victoria, Canada.

PhD. Jose Martinez-Llario
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022, Valencia, Spain
Tef.: +34 96 3877550, Fax: +34 96 387759, email: jomarlla@cgf.upv.es

PhD. Eloina Coll Aliaga
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022, Valencia, Spain
Tef.: +34 96 3877550, Fax: +34 96 387759, email: ecoll@cgf.upv.es

Jose Martinez received his M.Sc. degree in Geodesy and Cartography from the Polytechnic University of Valencia in Spain in 1999, and his PhD in GIS (Geographical Information System) and Cartography in 2003. His main research interests include free GIS software, Spatial Databases, 3D Visualization and Photogrammetry. He has collaborated with several industries and research centers, in particular on GIS, interoperability tools and open source projects. He is author of more than 20 papers in journals and international conferences

Eloina Coll received her laurea degree in Computer Engineering in 1992 and his PhD in Geographical Information System, Geodesy and Photogrammetry in 2001 at Polytechnic University of Valencia, Spain. Her research interests are the videogrammetry, camera calibration and GIS. She is author of more than 30 papers in journals and international conferences. She serves the program committees of workshops and conferences in the software GIS fields, such as the CISCI conferences. Nowadays, she is the main researcher of the project BIA2003-07914 SIGMUN. Research state grant of CICYT cofinanced with funds FEDER of the European Union