Análisis vectorial en PostGIS y Oracle Spatial: estado actual y evolución de la especificación *Simple Features for SQL*

Martínez Llario, José Carlos[1]
Coll Aliaga, Eloina[2]
Universidad Politécnica de Valencia (España), jomarlla@cgf.upv.es [1], ecoll@cgf.upv.es [2]

Resumen: *Este artículo pretende abordar los pasos a seguir, así como los problemas surgidos al resolver un ejemplo típico de análisis espacial vectorial utilizando las bases de datos espaciales PostGIS y Oracle Spatial, siguiendo en lo posible la especificación Simple Features for SQL (SFS) del OGC. La mayoría de trabajos realizados en la actualidad que soportan alguna de estas bases de datos espaciales utilizan el sistema únicamente para almacenar objetos geográficos sin aprovechar las capacidades de análisis vectorial. Se estudia también algunos problemas de rendimiento en ciertas operaciones de análisis, todo ello con el objetivo de iniciar una debate sobre el estado actual y la posible evolución de la especificación SFS del OGC, de forma que mediante su utilización se pueda incorporar en una IDE un motor de análisis espacial de forma sencilla y estándar, delegando el cliente esta funcionalidad en el servidor e incorporando en la IDE características de un software SIG de escritorio.*

1.- INTRODUCCIÓN

Tradicionalmente las operaciones de análisis espacial tanto raster como vectorial se han efectuado de forma local utilizando un software SIG de escritorio. Hace ya algunos años que las bases de datos más importantes del mercado han incorporado extensiones espaciales, sobre todo con el impulso de la organización Open GeoSpatial Consortium (OGC) apoyándose en la especificación 'Simple Features for SQL' [5]. Profundizando en esta especificación, nos abordan una serie de preguntas que este artículo junto con su presentación oral tratará de resolver y plantear algunas de ellas para un posible debate:

- ¿Se podría delegar en el sistema gestor de bases de datos, la capacidad de realizar el análisis espacial que necesita un Sistema de Información Geográfica?
- Pero, ¿aporta algún beneficio que el cliente delegue el análisis vectorial en el servidor?
- ¿La especificación SFS define de forma suficientemente amplia las capacidades que debe implementar un SGBD para poder efectuar un análisis espacial vectorial completo?
- ¿Los paquetes informáticos actuales de SIG utilizan estas capacidades?
- ¿Hacia donde debe evolucionar la especificación SFS? ¿Existe alguna normativa o alternativa más actual?

Este artículo trata de describir las herramientas más importantes que existen actualmente en el mercado, que capacidades tienen y cómo se deben utilizar para realizar un análisis espacial vectorial. Para ello, se han seleccionado dos bases de datos espaciales, una comercial y otra libre. PostGIS [2] (basado en PostgreSQL) es la alternativa de software libre más avanzada, por otro lado se ha utilizado el software comercial Oracle Spatial [1] que en su versión 10.2 incorpora un amplio abanico de funcionalidades.

En la primera parte del artículo se describe brevemente un análisis vectorial sencillo y la cartografía que se necesita. A continuación se resuelve el análisis utilizando estos dos SGBD. En una segunda parte se remarcan los puntos y características más importantes a tener en cuenta para realizar el análisis, así como la fidelidad de estos software con la especificación SFS. El artículo finaliza con unas conclusiones que tratan de responder a las cuestiones enunciadas anteriormente.

2.- ANÁLISIS ESPACIAL PROPUESTO

El objetivo de este análisis espacial es mostrar de una forma práctica cómo se realiza el mismo utilizando dos programas diferentes y si es posible realizar este análisis siguiendo de forma rigurosa la especificación SFS. En ningún caso se ha querido realizar cálculos de eficacia de los algoritmos utilizados. Por lo tanto, todas las capas cartográficas utilizadas son muy pequeñas y tienen menos de un centenar de entidades geométricas.

2.1.- Criterios análisis espacial

Localización de una determinada infraestructura bajo los siguientes criterios:

- Criterio zonal (capa 'suelos': tsuelo = 300)
- Criterio zonal (capa 'usos': tuso > 0)
- Proximidad (capa 'alcanta': < 300 metros)
- Proximidad (capa 'rios': > 20 metros (trio = 1) > 40 metros (trio = 2))
- Área final > 5 000 m^2

2.2.- Cartografía

- Capa 'suelos' (Figura 1). Capa de polígonos formada por 43 entidades describiendo los tipos de suelos. Esquema: (shape: polygon, tsuelo: short int {0,1,2,3})
- Capa 'usos' (Figura 2). Capa de polígonos formada por 76 entidades describiendo los tipos de usos de suelo. Esquema: (shape: polygon, tuso: short int {100,200,300,400,500,600,700})
- Capa 'rios' (Figura 3). Capa de líneas formada por 106 entidades describiendo los ríos existentes. Esquema (shape: line, trio: short int {1,2})
- Capa 'alcanta' (Figura 4). Capa de líneas formada por 6 entidades describiendo la red de alcantarillado. Esquema: (shape: line, id: short int {0}).

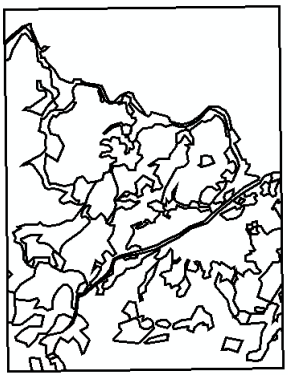

Figura 1: Capa de suelos

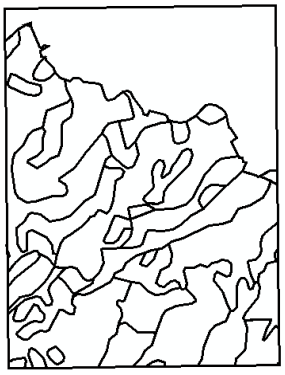

Figura 2: Capa de usos

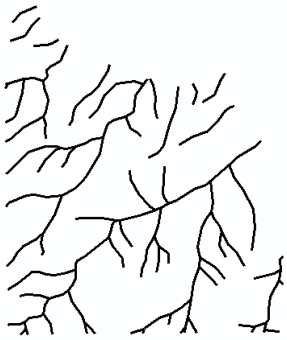

Figura 3: Capa de ríos

Figura 4: Capa de alcantarillado

Esta cartografía está depurada para que no existan vértices repetidos, polígonos con solape, vértices más próximos que la tolerancia establecida en el análisis con Oracle Spatial, o elementos no válidos según el OGC.

3.- ANÁLISIS ESPACIAL

3.1.- Importación de la cartografía con PostGIS

La importación de la cartografía se puede realizar utilizando el comando 'shp2pgsql' de PostGIS:

```
#!/bin/bash

#Ficheros necesarios:
#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 # nueva base de datos
createlang plpgsql test # lenguaje plpgsql
psql -f lwpostgis.sql -d test #PostGIS
psql -f spatial_ref_sys.sql -d test #EPSG

#Conversión de las capas shape a PostGIS
shp2pgsql rios.shp rios > rios.sql
shp2pgsql suelos.shp suelos > suelos.sql
shp2pgsql usos.shp usos > usos.sql
shp2pgsql alcanta.shp alcanta > alcanta.sql

#Carga de las capas
psql -d test -f rios.sql
psql -d test -f alcanta.sql
psql -d test -f suelos.sql
psql -d test -f usos.sql
```

Listado 1: Importación de datos espaciales en PostGIS

3.2.- Resolución con PostGIS

En el siguiente listado aparece las sentencias SQL del análisis, en negrita aparecen aquellas sentencias no consideradas en la especificación SFS y que son necesarias para realizar el análisis. Para simplificar el código del ejemplo, se ha sustituido las expresiones de creación de tablas de geometría por la sentencia 'crearTabla (tipo, nombre)'.

```
crearTabla (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);
```

Área de influencia de 'alcanta'

crearTabla (POLYGON, tmp1)

1: insert into tmp1(the_geom) select **buffer(the_geom,300,32)** from alcanta;

crearTabla (POLYGON, alcantabuf)

2: insert into alcantabuf(the_geom) select **geomunion(the_geom)** from tmp1;

Área de influencia de 'rios'

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

crearTabla (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;

crearTabla (MULTIPOLYGON, riosbuf)

7: insert into riosbuf(the_geom) select **geomunion (the_geom)** from tmp2;

Intersección de 'suelos' y 'alcanta'

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);

crearTabla (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;

Diferencia entre las áreas de influencia

crearTabla (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);

crearTabla (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);

Resultado final

20: select gid, area(the_geom) as area from final order by area desc;

Listado 2: Sentencias del análisis espacial en PostGIS

3.3.- Importación de la cartografía con Oracle Spatial

La importación de la cartografía se puede realizar utilizando el comando 'shp2sdo', que viene en el paquete de utilidades espaciales de Oracle Spatial.

```
#Conversión a formato sql espacial de Oracle
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

#Creación del esquema de las tablas (ejecución dentro de isqlplus)
start suelos.sql
start usos.sql
start rios.sql
start alcanta.sql
```

```
#Carga sql utilizando sqlloader
Sqlldr user/contraseña suelos
Sqlldr user/contraseña suelos
Sqlldr user/contraseña suelos
Sqlldr user/contraseña suelos
```

Listado 3: Importación de datos espaciales en Oracle Spatial

3.4.- Resolución con Oracle Spatial

En el siguiente listado aparecen las sentencias SQL del análisis. Para simplificar el código del ejemplo, se ha sustituido las expresiones de creación de tablas de geometría por la sentencia 'crearTabla (nombre)'.

```
crearTabla (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);
```

Área de influencia de 'alcanta'

```
crearTabla (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;

crearTabla (alcantabuf)

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

Área de influencia de 'rios'

```
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);

crearTabla (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;

crearTabla (riosbuf)

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

Intersección de 'suelos' y 'alcanta'

```
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;

crearTabla (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;
```

Diferencia entre las áreas de influencia

```
crearTabla (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;

crearTabla (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');
```

Resultado final

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

Listado 4: Sentencias del análisis espacial en Oracle Spatial

4.- Funcionalidades de los sistemas

4.1.- Importación de datos espaciales

Ambos sistemas disponen de un comando para la importación de ficheros en formato *shape* (*shp2pgsql* en PostGIS y *shp2sdo* en Oracle), estos comandos se invocan desde la consola del sistema e importan tanto los datos espaciales como los temáticos asociados (*.dbf*).

PostGIS
El comando de importación crea un único fichero *.sql* donde se incluyen las sentencias SQL necesarias para la creación de la tabla y la carga (con las correspondientes sentencias 'insert') de cada uno de los registros.

Oracle
El comando de importación crea dos ficheros; un fichero *.sql* con las sentencias SQL necesarias para la creación de la tabla y un fichero *.ctl* (fichero en formato de texto) con los datos de la tabla para su importación con la utilidad *sqlloader* de Oracle. En los parámetros del comando *shp2sdo* además se debe especificar ciertos parámetros como la extensión y la tolerancia de cada una de las dimensiones de las geometrías, información que se utiliza en la definición de la creación de la tabla.

4.2.- Especificación SFS

Aunque en la documentación de ambos productos se asegura que siguen la especificación SFS del OGC (pasando con éxito el test de comprobación del OGC 'Conformance Test Guidelines for OpenGIS Simple Features Specification for SQL' [4]), hay que realizar importantes matizaciones en este aspecto:

- Oracle Spatial efectivamente sigue la especificación pero utiliza mucho más la relajación (cambios permitidos por el OGC) del test mencionado anteriormente, es decir, los métodos definidos en SFS (constructores de geometría, operadores y predicados espaciales,…) tienen su correspondencia en métodos de Oracle Spatial pero éstos tienen diferente nombre, número o tipo de argumentos. Esto es una gran limitación ya que se necesita traducir el código SQL de Oracle si se quiere exportar a otros sistemas, incluso aunque éstos soporten de una manera estricta la especificación del OGC.

- PostGIS sigue de una manera mucho más fiel la especificación SFS, respetando los nombres de los métodos (salvo alguna excepción) y el número de argumentos. PostGIS almacena también la información de metadatos de las columnas de geometría de las tablas (GEOMETRY_COLUMNS) y de los sistemas de referencia espacial siguiendo el estándar SFS (SPATIAL_REFERENCE_SYSTEMS).

- Ambos sistemas incorporan métodos no definidos por el OGC en su especificación, pero necesarios si se quieren realizar análisis complejos o procedimientos almacenados que incrementen la funcionalidad del sistema. Como se verá mas adelante en el caso de PostGIS, algunos de estos métodos (que no siguen el estándar) son necesarios para realizar incluso el análisis vectorial más sencillo.

4.3.- Tablas espaciales

En los enunciados previos a los listados 2 y 4 correspondientes a los análisis espaciales, se muestra la diferente forma que tienen estos dos sistemas de crear una tabla de geometría:

- PostGIS utiliza el método 'addGeometryColumn' (método propuesto en la SFS), que se encarga de añadir el campo de geometría del tipo seleccionado a una determinada tabla, actualizar la información de los metadatos contenidos en la tabla GEOMETRY_COLUMNS, y añadir la restricción de tabla, del tipo de geometría en el campo correspondiente.

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

- Por el contrario Oracle no implementa el método 'addGeometryColumn', es decir, define la tabla directamente (cambiando además el nombre de la misma propuesto por el 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 limita el tipo de geometrías almacenadas en una tabla (restricción de tabla sobre la columna de geometría, fijando el tipo de ésta), aunque esta forma de trabajar (una capa agrupa entidades geométricas con el mismo tipo de geometrías) está mas acorde con la forma tradicional, presenta ciertos problemas (problemas que Oracle Spatial no tiene ya que no establece este tipo de restricciones en las tablas). Por ejemplo, si en una tabla de Polígonos se intenta almacenar un Multipolígono (o al revés), el sistema dará un error de violación de una retricción de tabla. Esto puede ocasionar como se verá más adelante la imposibilidad de realizar ciertas operaciones de análisis siempre y cuando el usuario no se ayude de métodos no definidos en el SFS y que pueden en cierta manera solucionar este problema.

4.4.- Incompatibilidad con la especificación SFS

En cuanto a las incompatiblidades de métodos que no siguen la norma SFS, podemos establecer dos grupos según la importancia de la incompatibilidad. Un primer grupo consistiría en aquellos métodos que aunque están definidos en la especificación SFS no coinciden en nombre, número o tipo de argumentos admitidos o devueltos (aunque la relajación del test del OGC permite cambiar el nombre a los métodos/tablas/vistas). Un segundo grupo estaría formado por métodos del SGBD que ofrecen una funcionalidad que no existen en la norma SFS. En este segundo grupo la incompatibildad se produce irremediablemente cuando alguno de estos métodos es indispensable para realizar una análisis espacial común, de forma que es necesario utilizarlo y de esta manera el código SQL generado no es compatible con la norma SFS.

4.4.1.- PostGIS

Según el Listado 2:
En el primer grupo tendríamos las sentencias que invocan a los métodos '***`BUFFER (the_geom,300,32)`***' y '***`GEOMUNION (the_geom)`***'. La incompatibilidad en el primer método se debe al tercer argumento (número de segmentos de los elementos curvilíneos) no considerado en la norma SFS, mientras que el segundo tiene como nombre original en la norma SFS *union* y no *geomunion* (PostGIS lo renombra para no coincidir con la palabra clave *union* de SQL).

PostGIS tiene cerca de 100 métodos que ofrecen una funcionalidad espacial extra, pero es por ejemplo el método 'multi', **`MULTI (`**`intersection (u.the_geom,s.the_geom)`**`)`**, el que resulta necesario utilizar en un análisis espacial común para solucionar el problema que se introduce a continuación:

Tipos de geometrías devueltas en una operación de intersección espacial en PostGIS

Muchos de las soluciones libres que utilizan análisis espacial se basan en la biblioteca JTS [6]. Este es el caso de PostGIS que se basa en la librería GEOS, que es un *wrapper* de JTS en C++. Para realizar el test se ha utilizado el software JTS Test Builder [5]. En este ejemplo se va utilizar únicamente el operador espacial 'intersection(A, B)', este operador devuelve la intersección entre las geometrías A y B. Para simplificar el caso vamos a considerar que A y B son de tipo POLYGON. Como se puede apreciar en las figuras, la intersección de dos polígonos puede producir un multipolígono (Figura 5) o una colección de geometrías (Figura 6) formada por puntos, líneas y polígonos.

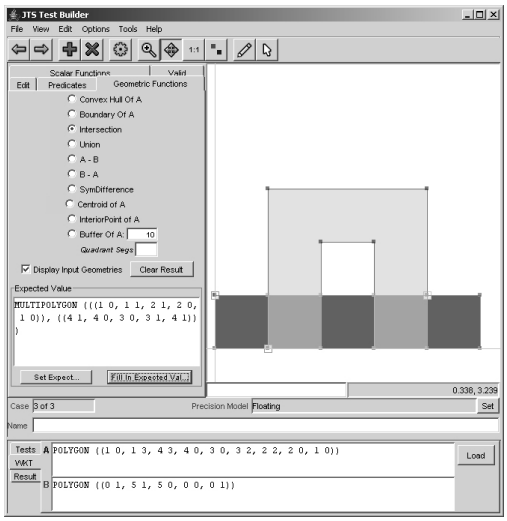

Figura 5: JTS Test Builder Software. Ejemplo 1

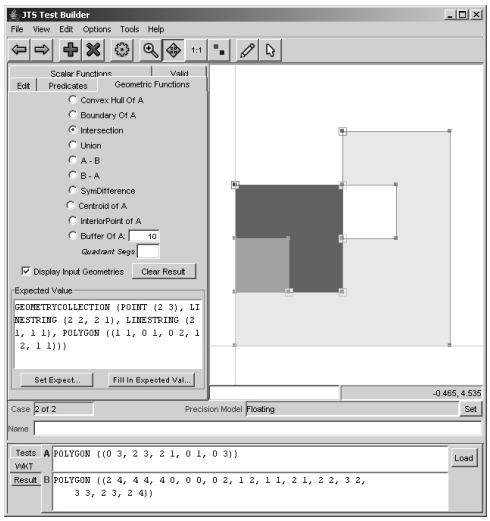

Figura 6: JTS Test Builder Software. Ejemplo 2

Al utilizar el operador espacial de intersección (entre polígonos) se deberá tener en cuenta las siguientes características:

- Una intersección de dos polígonos devuelve una única geometría.
- Si esta geometría está formada por varios elementos de tipo POINT, LINESTRING o POLYGON será de tipo 'multi', es decir, MULTIPOINT, MULTILINESTRING o MULTIPOLYGON.
- Si el resultado es una geometría mixta de varios tipos sencillos, será de tipo GEOMETRYCOLLECTION.

Estas características pueden ocasionar una **violación en las restricciones de tabla de PostGIS** [3], ya que PostGIS no puede almacenar en una tabla diferentes tipos de geometrías (restricción de la norma SFS). Evidentemente se puede aplicar una cláusula WHERE en la correspondiente sentencia SQL para filtrar el tipo de retorno de las intersecciones, pero esto producirá una perdida de elementos que pueden ser válidos en el resultado final. La única posible solución es utilizar el método 'multi' de PostGIS, para convertir las entidades devueltas en la intersección a entidades de tipo *multi* (MULTIPOLYGON, MULTIPOINT o MULTILINESTRING), y almacenarlas en una capa de tipo multipolígono por ejemplo. Aún así si se tuviera un caso como el de la figura 6, el problema no se podría resolver a menos que se realizará un programa en algún lenguaje de servidor como *pgplsql* y se utilizarán aún más métodos no considerados en la especificación SFS.

4.4.2.- Oracle Spatial

Como se puede observar en el Listado 4, todos los métodos prácticamente de Oracle Spatial se encontrarían encuadrados dentro de lo que hemos denominado primer grupo de incompatibilidad. Por ejemplo, además de renombrar los métodos, muchos de ellos utilizan un argumento adicional para la tolerancia de las coordenadas: `SDO_GEOM.SDO_BUFFER (r.geom,d.dist, 0.000001)`, y otros utilizan tipos definidos por el sistema: `SDO_AGGR_UNION (SDOAGGRTYPE(a.geom, 0.000001).`

En cambio no nos encontramos con uno de los problemas que impedía a PostGIS seguir la norma SFS (orden *multi*). En efecto, si nos fijamos en cómo se define una tabla de geometría en Oracle nos damos cuenta que no se especifica el tipo de geometría y en una tabla de polígonos se pueden almacenar también multipolígonos, lo que elimina el problema que tenía PostGIS.

4.5.- Problemas de eficiencia en las áreas de influencia

Como se puede ver en los listados de código SQL de los análisis espaciales, las áreas de influencia se realizan en dos pasos:

1. Primero se realiza el buffer de cada una de las geometrías de la capa
2. Segundo se disuelven las fronteras adyacentes entre los polígonos creados por el buffer

Estas dos operaciones normalmente en un SIG de escritorio se realizan de forma transparente en un solo paso para el usuario, pero en realidad el procedimiento es similar.

Si las áreas de influencias se realizan con sistemas que no tengan topología explícita (al contrario que por ejemplo la famosa estructura Arco-Nodo implementada en el formato cobertura de ArcInfo Workstation), la disolución de los límites entre los polígonos será una operación muy costosa para el sistema. Esto es especialmente significativo si se utilizan agregados SQL como es el caso de los sistemas estudiados que van acumulando tantas uniones de elementos como entidades existen en la capa. En efecto, la creación de las áreas de influencia sobre capas con varios miles de elementos colapsa a estos sistemas y a otros paquetes SIG tan conocidos como por ejemplo ArcGIS de ESRI.

Además al utilizar un agregado el resultado consiste en un único macropolígono, con lo cual se pierde todo el beneficio de la indexación espacial en operaciones posteriores.

Soluciones parciales pasarían por realizar *buffers* segmentados lo cual incrementaría la eficacia del sistema (Listado 5), o realizar procedimientos almacenados que calcularan las áreas de influencia únicamente de polígonos disjuntos, o con un número máximo de polígonos. Estas dos últimas soluciones aumentan enormemente la eficiencia de estos algoritmos (aunque no pueden en ningún caso igualar a sistemas con topología explícita) haciendo posible el cálculo de áreas de influencia de otra forma inviable en capas con un alto número de elementos geométricos. Pero otra vez el coste pagado sería el apartarse de la especificación SFS.

```
INSERT INTO alcantabuf (gid, geom)
SELECT 1,sdo_aggr_union(mdsys.sdoaggrtype(aggr_geom,0.5)) aggr_geom
FROM (SELECT sdo_aggr_union(mdsys.sdoaggrtype(aggr_geom,0.5)) aggr_geom
  FROM (SELECT sdo_aggr_union(mdsys.sdoaggrtype(aggr_geom,0.5)) aggr_geom
      FROM (SELECT sdo_aggr_union(mdsys.sdoaggrtype(aggr_geom,0.5)) aggr_geom
            FROM (SELECT sdo_aggr_union(mdsys.sdoaggrtype(geom,0.5)) aggr_geom
                  FROM tmp1 GROUP BY mod(rownum,16))
            GROUP BY mod (rownum, 8))
      GROUP BY mod (rownum, 4))
  GROUP BY mod (rownum, 2));
```

Listado 5. Segmentación en la creación de áreas de influencia

5.- CONCLUSIONES

En cuanto a la compatibilidad de los dos SGBD espaciales estudiados con la especificación SFS del OGC, PostGIS sigue mucho más fielmente la especificación que Oracle Spatial, pero este último es más fácil de manejar al no tener que comprobar constantemente los tipos de las geometrías devueltas para evitar errores de violación de las restricciones de tabla. De esta forma, para realizar un análisis espacial común y sencillo se necesita utilizar funcionalidades extra no recogidas en la especificación SFS como el método 'multi' de PostGIS o directamente no seguir las especificaciones en la definición de las tablas como hace Oracle.

Aún queda pues trabajo por realizar para poder delegar la funcionalidad de análisis espacial en el servidor, por lo menos si se quiere seguir la especificación SFS. Aunque existen opiniones a favor del sistema tradicional de efectuar el análisis en local (en el cliente), personalmente nosotros pensamos que el futuro pasa por implementar de forma óptima esta funcionalidad en el servidor, lo cual aportaría muchas ventajas como: mantenimiento de equipos, coste de licencias, incremento en la separación lógica de la estructura cliente-servidor en un SIG, partir de una base común más amplia en la elaboración de SIG de escritorio, etc. Todo esto repercutiría en dotar de servicios de análisis espacial a un cliente SIG, y en acercar la posibilidad de crear un Web SIG totalmente operativo utilizando protocolos estándar. La aplicación quizás más espectacular consiste en dotar a una IDE de un completo análisis espacial y uno de los objetivos finales es quizás obtener un SIG profesional a través de la Web.

Pero todo esto aún no es realidad ya que como se ha comentado anteriormente queda trabajo por hacer (sobre todo si hablamos de protocolos estándar y software libre). En efecto, actualmente los software SIG no utilizan estas capacidades de análisis espacial en la parte del servidor (a excepción de unos pocos productos comerciales, de muy alto costo, y bajo sistemas no compatibles con OGC), como mucho, son capaces de leer directamente de PostGIS (otros solo se limitan a importar o exportar la información a PostGIS) y realizar operaciones de análisis vectorial en el cliente (convirtiendo los datos a un formato interno o al formato de las bibliotecas que utilicen como las JTS).

Existen ciertas funcionalidades espaciales que se pueden beneficiar de un modelo de topología explícito si están correctamente implementadas, como las áreas de influencia. La topología explícita debe ser obligatoria ya que además de evitar redundancia de datos y añadir conectividad, acelera enormemente ciertas operaciones de análisis espacial vectorial que de otra forma podrían llegar a ser inviables. Actualmente no existen bibliotecas libres que implementen un modelo de topología explícito aunque ya se está trabajando en bibliotecas como GeoTools. En cuanto a PostGIS, actualmente se está implementando un modelo de topología basado en la norma ISO 13249 (SQL/MM parte 3). Aunque una base de datos espacial con un modelo de topología explícito beneficia en gran medida las operaciones de análisis espacial, puede perjudicar de forma también apreciable la lectura de un gran número de elementos cartográficos (operación común en el renderizado de los servidores de cartografía) ya que las entidades se deben formar a partir de las topologías integrantes cada vez que se solicita un elemento. En estas operaciones intervienen relaciones transversales entre elementos que se deben implementar bajo un gestor de bases de datos objeto-relacional (SQL3) que implemente de forma eficaz estas características.

REFERENCIAS

1. Manuales de Oracle Spatial 10g. http://www.oracle.com (último acceso: Octubre, 2005)
2. Manual de PostGIS. http://www.postgis.org/ (último acceso: 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.