Geoprocesamiento con SQL en OrbisGIS

Fernando González, Erwan Bocher, Thomas Leduc

Geoprocesamiento con SQL en OrbisGIS.

F. González Cortés[(1)], *E. Bocher*[(2)] *y T. Leduc*[(3)]

[(1)] OrbisGIS - Freelance, fergonco@gmail.com.
[(2)] IRSTV - FR CNRS 2488, erwan.bocher@ec-nantes.fr
[(3)] Laboratorio CERMA, IRSTV – FR CNRS 2488, thomas.leduc@cerma.archi.fr

RESUMEN

OrbisGIS dispone de un lenguaje que permite la manipulación de datos de forma independiente al formato. Éste se ajusta al estándar SQL92 (Lenguaje de Consulta Estructurado) y se extiende espacialmente según el OGC simple features SQL specification, lo que permite un alto nivel de compatibilidad con sistemas de bases de datos tradicionales como PostgreSQL/PostGIS. Habitualmente el SQL es procesado en servidores, sin embargo este artículo presentará cómo puede aplicarse en el cliente, principalmente para la definición y utilización de geoprocesos. La aplicación del SQL más inmediata es la manipulación de las fuentes de datos especificando una serie de instrucciones o script. Aparte, la implementación de SQL de OrbisGIS permite la parametrización de estos scripts de manera que pueden ser reutilizados con diferentes fuentes de datos y sin necesidad de conocer el proceso internamente. La reutilización puede darse tanto como ejecución del script como inclusión en un constructor de modelos que permite encadenar múltiples scripts dando lugar a un nuevo proceso más complejo. Por otra parte, el uso de disparadores permite la definición de reglas de validación mediante instrucciones SQL. Es posible definir reglas topológicas (entre otras) que definan relaciones entre dos o más fuentes de datos y que controlarán el proceso de edición tanto espacial como alfanumérico. Para finalizar, la especificación de vistas (en el sentido de los SGBD) permite reducir la redundancia en los datos y realizar algunas aplicaciones interesantes como la visualización de la evolución de los distintos imperios a lo largo de la historia.

Palabras clave: *SIG, SQL, edición, manipulación de datos, geoprocesos*

INTRODUCCIÓN

Recientemente, se han realizado muchos esfuerzos en el desarrollo de Infraestructuras de Datos Espaciales a diferentes niveles (local, nacional, etc.) y en distintos campos de aplicación (social, económico, medioambiental, ...)[1][2]. Todos estos esfuerzos comparten los mismos objetivos: maximizar el acceso a la información por parte del usuario y minimizar la redundancia de esfuerzos e inversiones[3].

A medida que el volumen de la información disponible aumenta, lo hace también la necesidad de obtener herramientas capaces de procesar dicha información para

obtener resultados de interés. La aparición del estándar WPS (Web Processing Services) define una interfaz que permite la publicación de geoprocesos de manera estándar y por tanto, es de esperar que vayan apareciendo distintas soluciones para su creación y publicación de forma sencilla.

Entre las soluciones existentes en el campo de geoprocesamiento podemos encontrar implementaciones de SQL espacial como PostGIS[4], el proyecto JEQL[5] que define un nuevo lenguaje para la manipulación de datos y Sextante[6] que permite la creación de nuevos procesos en java.

En el presente artículo describiremos la solución adoptada por OrbisGIS: una implementación del estándar SQL extendido espacialmente y que permite la definición y ejecución de geoprocesos con distintas fuentes de datos.

Comenzaremos por describir la arquitectura de la solución implementada en OrbisGIS y la motivación que llevó al desarrollo de dicha solución. Después explicaremos las distintas aplicaciones que el SQL puede tener en los clientes SIG en general y en OrbisGIS en particular.

MOTIVACIÓN

El IRSTV (Institut de Recherche en Sciences et Techniques de la Ville) es una federación de laboratorios de investigación creada por el CNRS (Centre National de la Recherche Scientifique) en la que se llevan a cabo investigaciones multidisciplinares aplicadas al entorno urbano. En el seno del proyecto de puesta en marcha de la IDE del instituto se plantea el desarrollo de una herramienta que utilizarían los investigadores del instituto para apoyarse y plasmar el resultado de sus trabajos. Por otra parte, dentro del instituto existen aplicaciones propias que usan formatos también propios y que es necesario integrar también en la IDE.

Todo ésto, junto con la efectividad del SQL en el procesado de datos hizo que se desarrollara una implementación de dicho lenguaje capaz de operar con distintos tipos de ficheros y bases de datos. Se tomó como base el proyecto GDBMS, que formaba parte de gvSIG[7], y se procedió a su desarrollo y optimización. Adicionalmente se implementó una interfaz gráfica que permitiera la explotación amigable de GDMS: OrbisGIS.

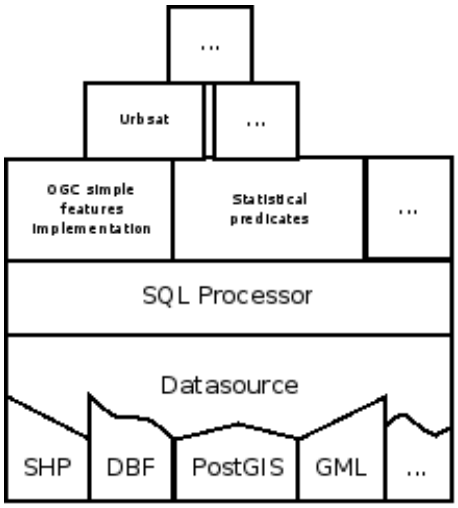

Figura 1: Arquitectura de GDMS.

La arquitectura de GDMS consiste en una capa de abstracción para el acceso a la información vectorial sobre la cual se construye un motor SQL. La implementación del lenguaje se ajusta en gran parte al estándar SQL92 y se extiende espacialmente según el estándar OGC simple features SQL specification.

Ésto hace que GDMS pueda ejecutar instrucciones SQL de forma independiente al formato y a la ubicación donde se encuentran los datos. Por ejemplo, es posible ejecutar instrucciones SQL que procesen shapefiles, tablas de PostgreSQL, una mezcla de ambas o cualquier otra fuente de datos para la que GDMS tenga un driver que permita su acceso.

En la Figura 2 podemos ver un ejemplo de GDMS ejecutado interactivamente tras efectuar una intersección entre una capa de edificios llamada *bati* y otra capa conteniendo solamente la geometría que queremos intersectar (*fence*). El resultado son los edificios color calabaza.

Figura 2: Intersección ejecutada interactivamente.

¿Por qué SQL?

SQL (Structured Query Language) es un lenguaje de consulta que permite la manipulación y creación de tablas en sistemas de gestión de bases de datos relacionales. La elección de SQL como lenguaje de manipulación vino determinada por varios factores. Principalmente, el hecho de que SQL haya demostrado su eficacia procesando datos durante más de 30 años era una garantía de que una buena implementación de dicho lenguaje iba a ser igualmente eficaz. Ésto, unido a las soluciones ya existentes que añaden tipos de datos espaciales a motores SQL (PostGIS, MySQL spatial, ...) eliminaban cualquier duda sobre la aplicabilidad de la solución.

Más concretamente, el SQL tiene una serie de ventajas sobre la programación imperativa. Por una parte, los índices, tanto espaciales como alfanuméricos, son usados de forma automática por el optimizador de consultas. Por otra parte, la complejidad en una instrucción SQL es mucho menor que la existente en el código imperativo que realiza la misma operación, por lo que la probabilidad de introducir un *bug* en una instrucción SQL es también mucho menor.

Desde el punto de vista didáctico, el SQL es un lenguaje de consulta con una curva de aprendizaje bastante suave en comparación con lenguajes de programación

imperativa. Su complejidad es menor y existen además infinidad de recursos para su aprendizaje, aparte de ser ya conocido por muchas personas, notablemente aquellas dedicadas a la gestión de bases de datos.

Sin embargo, no todo son ventajas. El lenguaje SQL implementa una serie de operadores relacionales que son muy adecuados para resolver determinados problemas pero que son ineficientes o insuficientes para otros. Se podría decir que el SQL está más limitado que un lenguaje imperativo. Para solucionar ésto se ha dotado a GDMS de la posibilidad de ser extendido de tal manera que es posible escribir un proceso con código java e invocarlo con una simple instrucción SQL.

GEOPROCESAMIENTO EN ORBISGIS

A continuación veremos en detalle las capacidades de geoprocesamiento que GDMS puede dar en un cliente SIG. En concreto se verá en el contexto de OrbisGIS por lo que previamente describiremos su interfaz gráfica.

Interfaz gráfica de OrbisGIS

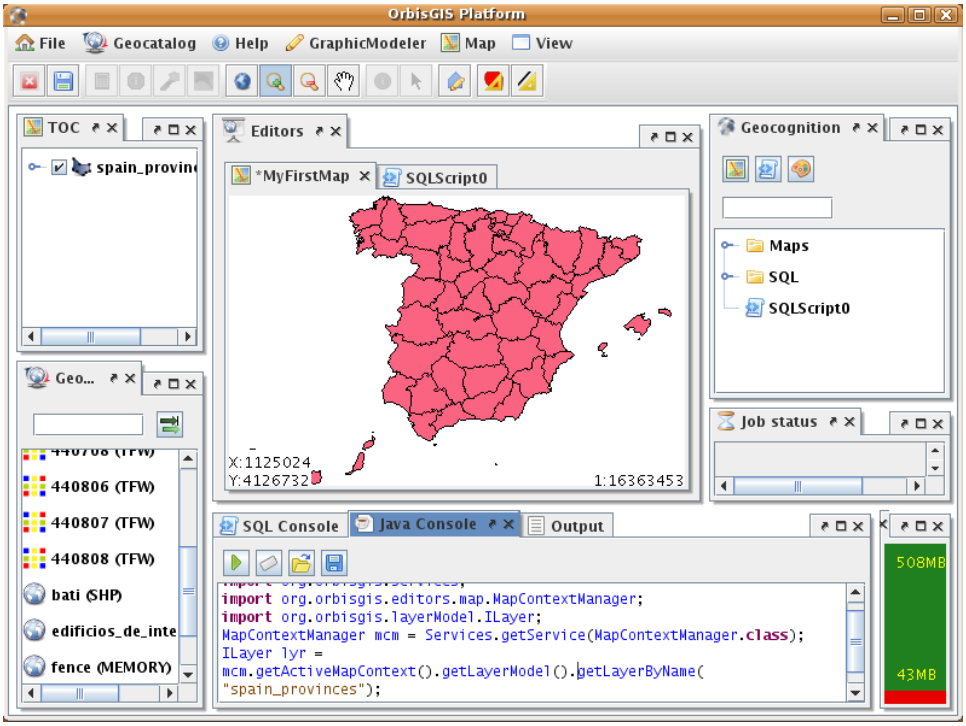

Figura 3: Interfaz gráfica de OrbisGIS.

Los componentes principales de OrbisGIS que podemos ver en la figura 3 son: la *consola SQL*, que permite la ejecución interactiva de código SQL contra las fuentes de datos; el *geocatálogo (abajo a la izquierda)*, donde se encuentran todas las fuentes registradas que pueden ser usadas en la *consola SQL*; el *geocognition, que* contiene el resultado de la productividad del usuario: mapas, scripts, etc; y los *editores*, que permiten la modificación de los elementos del *geocognition*.

Manipulación de datos de forma interactiva

OrbisGIS permite la interacción con el mapa activo mediante la *consola SQL*. Podemos por ejemplo intersectar dos capas entre sí y añadir el resultado al mapa activo (Figura 2).

De manera similar, podemos crear fuentes de datos mediante la interfaz gráfica de OrbisGIS e interactuar con ellas con la *consola SQL*. En concreto la herramienta *fence* permite crear una fuente de datos conteniendo la geometría dibujada por el usuario. Posteriormente es posible intersectar esta nueva fuente con una capa existente (Figura 2).

Por último, aprovechando las capacidades de extensión del SQL podemos interactuar directamente con la interfaz gráfica. OrbisGIS proporciona un panel llamado *geomark* en la que podemos almacenar marcadores espaciales. Además del panel, proporciona una función SQL con el mismo nombre, que añade las geometrías que recibe como parámetro al panel *geomark*. Así, podemos ver en las figuras 4 y 5 cómo se pueden añadir a los marcadores las posiciones de las geometrías inválidas de una capa para visitarlos posteriormente.

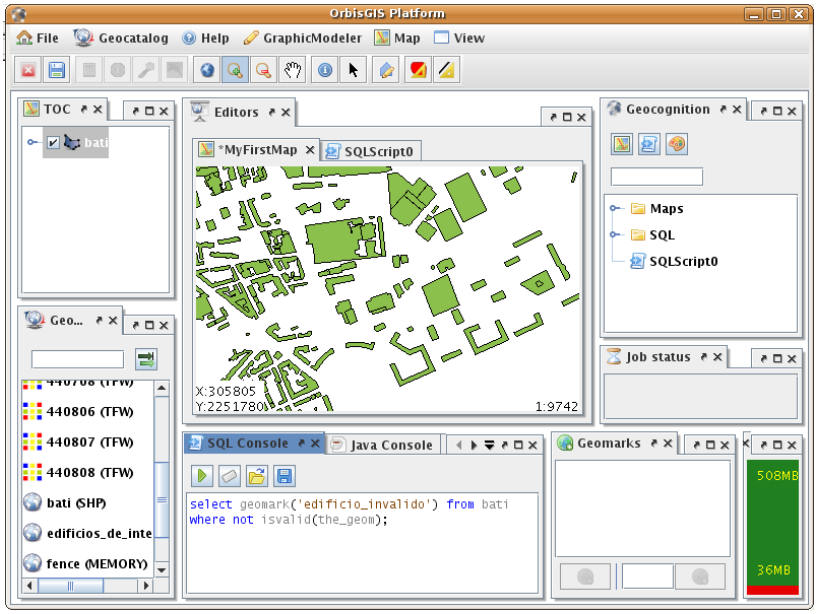

Figura 4: Marcado de geometrías no validas mediante SQL.

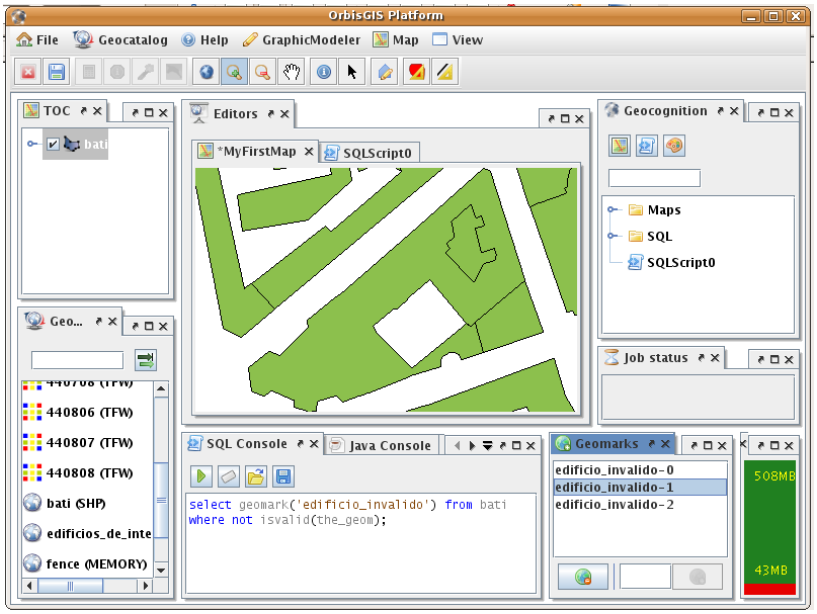

Figura 5: Visitando una geometría no válida con el panel geomark (abajo a la derecha se solapa consigo misma).

Ejecución de scripts parametrizados

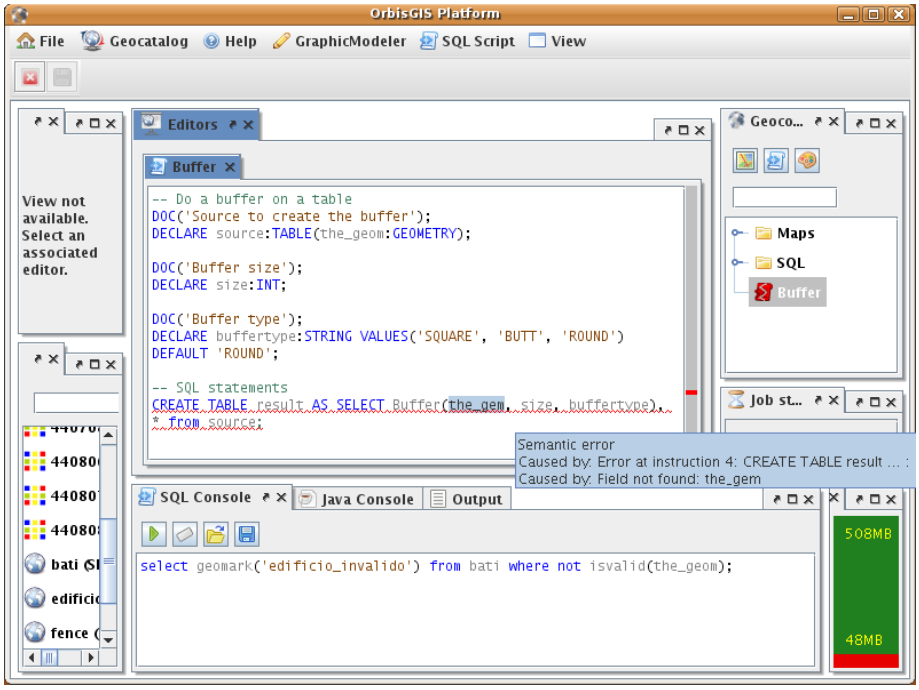

Figura 6: Script parametrizado.

Hemos visto como podemos ejecutar secuencias de instrucciones de forma interactiva. Ésto es útil para la manipulación de datos, pero no permite la reutilización fácil del código SQL creado. Por ejemplo, el código de la figura 2 fallará cuando la capa no se llame *bati* y para hacerlo funcionar habría que leer el script y substituir manualmente las distintas referencias a *bati* por el nombre de la capa con la que se quiere interactuar.

Para dar solución a ello se ha extendido la sintaxis SQL dando la posibilidad de especificar los parámetros de entrada de un script. El código SQL es precedido por una sección de declaración de parámetros en las que se especifican los parámetros escalares que recibe el script así como los parámetros con estructura tabular y los campos que éstos deben de tener. Podemos ver un ejemplo de dicha sintaxis en la figura 6, donde se define primero un parámetro con estructura tabular que debe tener al menos un campo geométrico y segundo un par de parámetros escalares, uno de los cuales es un número y el otro un valor de cadena a elegir de una lista.

Esta definición nos permite generar al vuelo una interfaz gráfica para la introducción de los parámetros de un script justo antes de la ejecución, permitiéndonos de esta manera realizar distintas ejecuciones de un mismo script con distintos valores de entrada (distintas fuentes de datos).

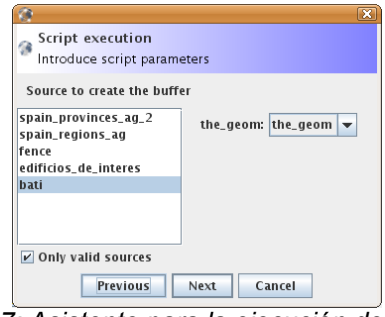

Figura 7: Asistente para la ejecución de scripts.

En el momento de la ejecución, GDMS substituye los parámetros escalares por los valores especificados y las referencias a los parámetros tabulares y sus campos por las fuentes de datos y campos seleccionados en la interfaz gráfica. Como consecuencia de ésto, el SQL definido en OrbisGIS puede ser considerado un lenguaje para la definición de geoprocesos, ya que el código es independiente de la fuente de datos de entrada. Como prueba de concepto, se han desarrollado de forma satisfactoria scripts para las operaciones más habituales en geoprocesamiento: join, split, dissolve, geometría más cercana, etc. y que están incluidos en OrbisGIS a disposición del usuario. El resultado para estos procesos suele ser un script de dos o tres declaraciones de parámetros y una o dos instrucciones SQL.

Constructor de modelos

En el punto anterior hemos visto cómo se pueden construir scripts que definen geoprocesos y que éstos pueden usarse a modo de *cajas negras* sin necesidad de ver su código. Hemos comentado también la implementación de operaciones de geoprocesamiento básicas, como *dissolve*, *buffer*, etc.

Sin embargo, habitualmente encontramos problemas más grandes de lo que puede resolverse con un *dissolve* o un *buffer*. Aunque sería posible crear scripts más complejos que resolvieran estos problemas, lo ideal es poder encadenar la ejecución de elementos más simples para obtener un geoproceso más complejo.

Orbisgis permite ésto mediante la creación de modelos en los que se pueden especificar gráficamente las fuentes de datos de entrada y salida así como la cadena de procesos a aplicar (Figura 8). Existen dos tipos de procesos que pueden incluirse en los modelos. Por una parte están los scripts de los que ya hemos hablado y por otra parte es posible incluir modelos enteros como procesos dentro de otros modelos.

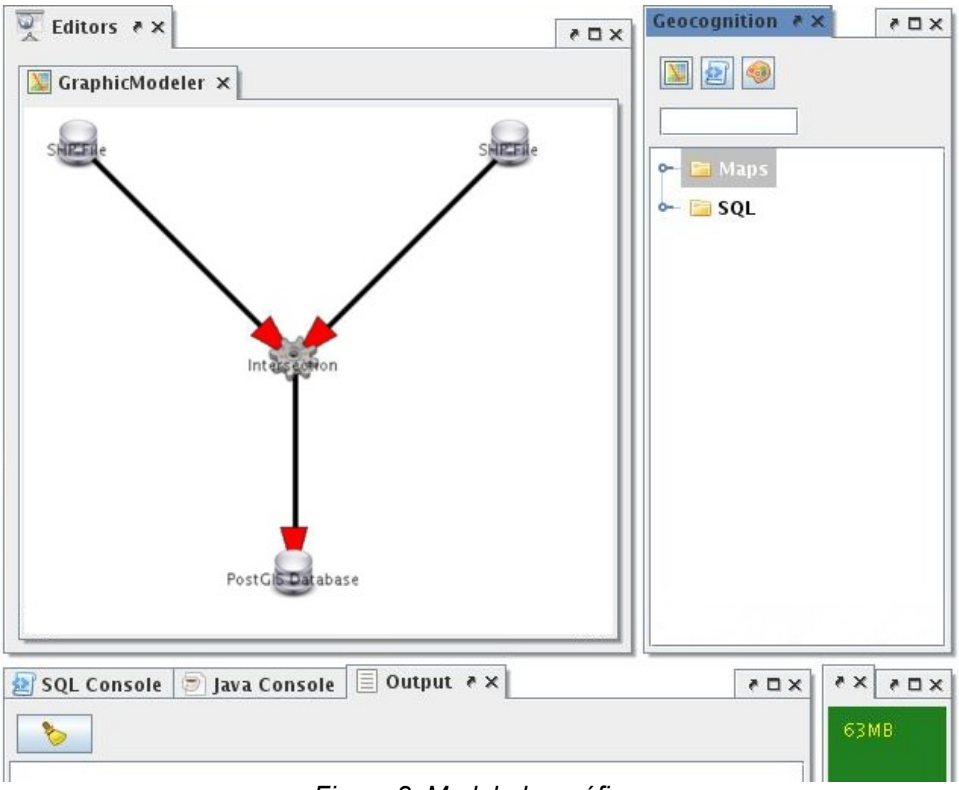

Figura 8: Modelador gráfico.

REGLAS DE VALIDACIÓN

Aparte de la definición de procesos, el SQL definido en OrbisGIS permite otras aplicaciones, como la definición de *triggers* o disparadores. En los sistemas de bases de datos tradicionales, un disparador es código que se ejecuta en respuesta a determinado evento de la base de datos (típicamente modificaciones en alguna tabla con la finalidad de validarlas). En OrbisGIS se implementa un disparador que comprueba la validez de los datos en base a unas instrucciones SQL.

Para validar una capa mediante una instrucción SQL hay que especificar una instrucción que devuelva los registros no deseados. Cuando el disparador sea ejecutado, comprobará que todas las instrucciones devuelven un conjunto de resultados vacío y en caso contrario cancelará o notificará al usuario la modificación que causó la inconsistencia.

Por ejemplo, si se están creando datos sobre yacimientos arqueológicos en la provincia de Valencia se podría introducir la siguiente instrucción:

SELECT y.* FROM provincias p, yacimientos y WHERE p.code='valencia' AND NOT contains(p.the_geom, y.the_geom);

la cual obtendría los yacimientos cuya geometría no está contenida en la geometría de la provincia de Valencia.

Obviamente realizar dicha comprobación cada vez que el usuario modifique los datos es demasiado pesado. El uso de índices espaciales y alfanuméricos permitiría ejecutar las comprobaciones en tiempos relativamente bajos pero si la edición se hace de forma interactiva, la experiencia para el usuario sigue sin ser agradable.

Para evitar ésto, OrbisGIS permite que los disparadores sean ejecutados tanto al modificar una fuente de datos como al guardar esas modificaciones. El guardado es una operación que se realiza mucho menos frecuentemente que la edición y por tanto la validación en ése instante no es tan molesta como durante la edición. El hecho de que las comprobaciones se realicen al guardar no supone ningún inconveniente puesto que las propias instrucciones de validación devuelven los registros causantes de las inconsistencias y por tanto éstas pueden localizarse fácilmente.

VISTAS

La última aplicación del SQL que trata este artículo es la definición de vistas. En los sistemas de bases de datos tradicionales, una vista es una tabla virtual definida en base a otras tablas, ya sean físicas o virtuales. Paralelamente, OrbisGIS permite definir vistas mediante instrucciones SQL que involucren distintas fuentes de datos físicas (shapefiles, tablas de postgresql,...) o virtuales (otras vistas).

De este modo, si por ejemplo se tiene por un lado un shapefile con las parcelas catastrales y con un campo alfanumérico *ref* conteniendo la referencia catastral, y por otro una tabla en postgresql con las columnas *ref* y *propietario* conteniendo respectivamente la referencia catastral y el propietario, sería posible crear una vista con el resultado de incorporar el *propietario* a los contenidos catastrales. Para ello habría que utilizar una instrucción como la siguiente:

SELECT * FROM parcelas, propietarios WHERE parcelas.ref=propietarios.ref;

Ésto haría posible, por ejemplo, la creación de un mapa temático visualizando las parcelas con un color en función del propietario sin necesidad de crear una nueva fuente de datos físicamente y actualizándose automáticamente cada vez que hay un cambio en las parcelas o en los propietarios. La ventaja más importante con respecto a una solución en la que se unieran ambas tablas de forma física es la eliminación de la redundancia en los datos y por tanto la reducción en la complejidad de tener todos los datos actualizados.

Por último, se ha desarrollado una pequeña aplicación para la visualización de datos en distintos períodos de tiempo gracias al concepto de vista. La aplicación consiste en un deslizador temporal que gestiona el contenido de una fuente de datos llamada *ogtime* con una única fila y una única columna y que contiene el instante de tiempo que se quiere visualizar. Cada vez que el deslizador se mueve, el valor de *ogtime* es actualizado al instante de tiempo que marca el deslizador. Mediante una vista es posible filtrar cualquier juego de datos que contenga un par de campos definiendo el intervalo de tiempo en el que es válida cada entidad de manera que el resultado sólo contenga entidades válidas en el instante definido por el deslizador. La vista se definiría con una instrucción muy similar a la siguiente:

```
CREATE VIEW history_instant AS SELECT * FROM history h, ogtime t where h.start <
t."time" and h.end > t."time";
```

En las figuras 9 y 10 podemos ver este ejemplo con los distintos imperios que han existido a lo largo de la historia.

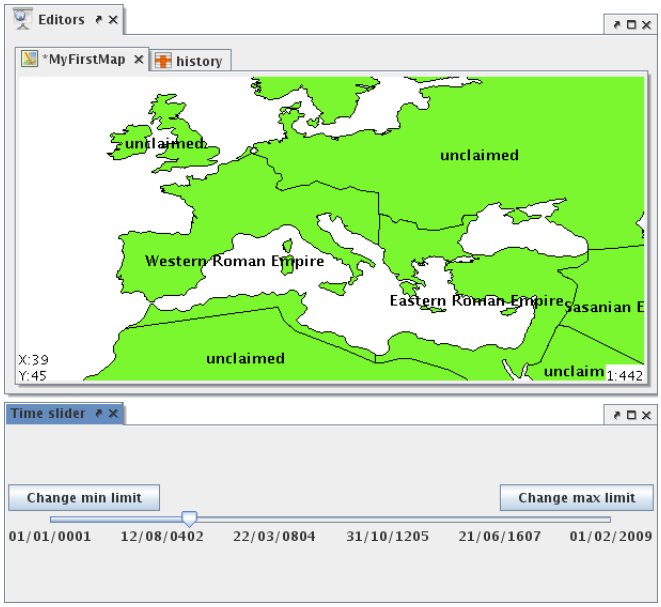

Figura 9: La vista cambia al cambiar el deslizador temporal (I)

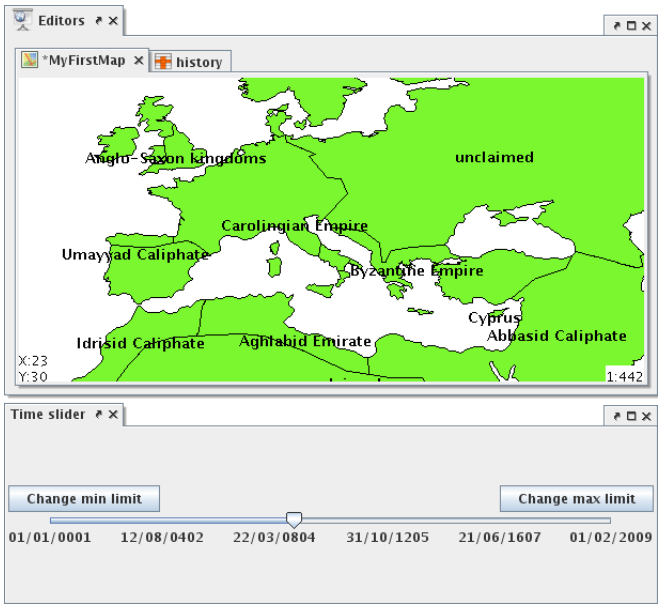

Figura 10:La vista cambia al cambiar el deslizador temporal (II)

TRABAJOS FUTUROS

La reciente aparición del estándar WPS (Web Processing Services) poco a poco está suponiendo un incremento en la disponibilidad de geoprocesos para el público. Una de las implementaciones más extendidas es la realizada por la iniciativa de 52ºNorth[8] la cual, además de hacer una implementación del estándar lo extiende añadiendo la posibilidad de publicar geoprocesos desde el cliente[9].

Está previsto extender dicho servidor para la publicación de geoprocesos definidos en SQL en OrbisGIS. Esta funcionalidad permitiría la creación y publicación de procesos muy ágilmente.

REFERENCIAS

- [1] Clinton W. Executive Order 12906 (apr. 1994): Coordinating geographic data acquisition and access: The National Spatial Data Infrastructure.
- [2] INSPIRE (2007). Directive of the European Parliament and of the Council establishing an Infrastructure for SpatialInformation in the European Community, http://www.ec-gis.org/inspire/.
- [3] Nebert D. Editor (2004) Developping Spatial Data Infrastructures: the SDI Cookbook. Online book, www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf.
- [4] PostGIS spatial database extension for PostgreSQL. http://postgis.refractions.net/.
- [5] JEQL Query Languaje. http://tsusiatsoftware.net/jeql/main.html
- [6] SEXTANTE (Sistema Extremeño de Analisis Territorial). http://www.sextantegis.com/.
- [7] gvSIG Project. http://www.gvsig.gva.es/.
- [8] 52ºNorth WPS server. http://52north.org/maven/project-sites/wps/52n-wps-webapp/.
- [9] Bastian Schaeffer (2008). 66th OGC Technical Committee . Atlanta, Georgia USA .