SERVEI DE SISTEMES
D'INFORMACIÓ GEOGRÀFICA
I TELEDETECCIÓ
Universitat de Girona

V JORNADAS DE SIG LIBRE

Desarrollo de un índice espacial para la extensión JASPA sobre H2.

J.A. Calvillo Ardila[(1)], *J.M. De Diego Alarcón* [(1)] *y A. Pérez-Navarro* [(1)]

[(1)] Estudis d'Informàtica, Multimèdia i Telecomunicació, Universitat Oberta de Catalunya, Rambla Poblenou 156, 08018 Barcelona, [jcalvilloa|jde_diegoa|aperezn]@uoc.edu

RESUMEN

El presente trabajo tiene por objetivo ofrecer una solución para la creación de un índice espacial para la extensión JASPA (Java SPAtial) sobre la base de datos H2. Esta propuesta está limitada a operaciones espaciales en dos dimensiones. El algoritmo de indexación elegido para la implementación del índice espacial ha sido el Rtree.

La implementación se ha realizado con el lenguaje de programación Java lo que ha facilitado su integración con la extensión JASPA y la base de datos H2. El índice es persistente en memoria secundaria en una tabla de la propia base de datos H2.

La solución que se propone es lo suficientemente flexible como para que no se haya necesitado modificar ni el código fuente de JASPA, ni de H2. Además, se ha previsto que el algoritmo de indexación se pueda mejorar o sustituir fácilmente.

Por último, se ha tenido en cuenta que el proceso de creación y manipulación de los índices espaciales sea intuitivo y fácil de usar.

Palabras clave: *Índice espacial, Rtree, Árbol R, JASPA, H2.*

ABSTRACT

This paper aims to provide a solution for creating a spatial index for the extension JASPA (Java Spatial) on the H2 database. The indexing algorithm chosen to implement the spatial index has been the Rtree.

The implementation was done with the Java programming language which has facilitated its integration with the extension JASPA database and H2, since both projects are coded in the same language. The index is persistent in secondary memory in a table in the database itself H2.

The proposed solution is limited to two-dimensional space operations and is flexible enough to not have needed to modify or JASPA source code, or H2. In addition, it is expected that the indexing algorithm can easily upgrade or replace.

Finally, we have taken into account that the process of creation and manipulation of spatial indexes is intuitive and easy to use.

Key words: *Spatial index, Rtree, JASPA, H2.*

INTRODUCCIÓN

El uso de Bases de Datos (*BBDD*) ligeras para el almacenamiento de información espacial y su posterior explotación en Sistemas de Información Geográfica *SIG*) constituye una tendencia en los últimos años en algunas de las aplicaciones *SIG* propietarias más extendidas. Este tipo de Bases de Datos resuelven varios problemas en la gestión de información geográfica:

- permiten el modelado relacional de información geográfica, no soportado por los formatos tradicionales basados en archivos como shapefile (*shp*).
- facilitan el intercambio de información geográfica entre usuarios.
- reducen al máximo las necesidades de administración y consumo de recursos, en comparación con las *BBDD* corporativas.

En el mundo de software libre, sin embargo, aún no existe una alternativa clara para este tipo de bases de datos.Existen diversos proyectos de software libre que intentan dar soporte espacial a *BBDD* ligeras; por ejemplo, *SpatialDBBox* y *HatBox* proveen de almacenamiento de objetos geográficos y funcionalidad de análisis espacial a *BBDD Java* como *H2*, *Derby* y *HSQLDB,* pero todas ellas adolecen de índices espaciales que mejoren el rendmiento de las consultas.

Examinando las características de extensiones espaciales de bases de datos como *Oracle Spatial* y *PostGIS*, se ha realizado en estudios anteriores una comparativa entre las bases de datos mencionadas (*H2*, *Derby* y *HSQLDB*) que ha permitido identificar *H2* como la mejor candidata para la implementación de una extensión espacial.[1,2]

Recientemente se ha presentado la extensión *JASPA*[1] (*Java Spatial*), soportada sobre *PostgreSQL* y *H2*. *JASPA* recoge funcionalidad espacial sobre *PostgreSQL* y *H2* e implementa el estándar del *Open Geospatial Consortium SQL Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option* .

En este trabajo se implementa un índice espacial en la extensión *JASPA (JAva SPAtial)* para acelerar las operaciones sobre campos con información espacial en la base de datos *H2*[2].

Los objetivos son:

- Implementar una solución tecnológica que permita crear un índice espacial sobre la base de datos *H2* .
- Dar soporte a las funcionalidades espaciales implementadas en *JASPA*.

Este artículo se ha dividido en los siguientes apartados, en el primero de ellos se realiza una introducción. Posteriormente en el segundo apartado se exponen algunos conceptos teóricos básicos que servirán para comprender la solución final.A continuación, en el tercer apartado se describen las principales características de la extensión *JASPA* y *H2*. Más adelante, en el cuarto apartado se describe la solución implementada, se profundiza en la integración de *JASPA* con *H2* y se muestran ejemplos de uso de la aplicación realizada. Por último, en el quinto apartado se realiza un conjunto de pruebas para demostrar la eficiencia de la implementación y en el apartado sexto se recogen las conclusiones.

INDEXACIÓN DE DATOS GEOGRÁFICOS

Los índices permiten acelerar la realización de consultas sobre los datos mediante la eliminación de la necesidad de recorrer de forma secuencial todos los datos. Desde el punto de vista de la base de datos son estructuras que quedan almacenadas de forma independiente de las tablas que contienen los datos pero que a su vez dependen de ellas.

1 JASPA: http://jaspa.forge.osor.eu/
2 H2: http://www.h2database.com/

Básicamente un índice consiste en almacenar en una tabla parejas de elementos: por un lado el elemento que se desea indexar y, por el otro, su posición en la base de datos. Cuando hay que localizar un elemento previamente indexado, únicamente hay que buscar dicho elemento en el índice, y una vez localizado, acceder al registro indicado por la posición definida por el índice.

Una base de datos relacional presenta deficiencias para manejar objetos de tipo geométrico y no puede indexar datos espaciales. Para cubrir estas necesidades se definen nuevas estructuras de almacenamiento para datos espaciales. Los índices espaciales están diseñados para optimizar las búsquedas basadas en criterios espaciales sobre la información geográfica.

A continuación profundizaremos en el funcionamiento de un índice espacial.

2.1 Principio de Indexación Espacial

La construcción de un índice espacial no está basada directamente en los objetos, sino en el mínimo rectángulo que engloba a cada uno de ellos (Fig. 1).

Al rectángulo mínimo que engloba a un objeto se le denomina *REM* (*Rectángulo Espacial Mínimo*), en inglés, *MBR (Minimum Bounding Rectangle)*. De esta manera se simplifican las operaciones sobre un índice espacial.

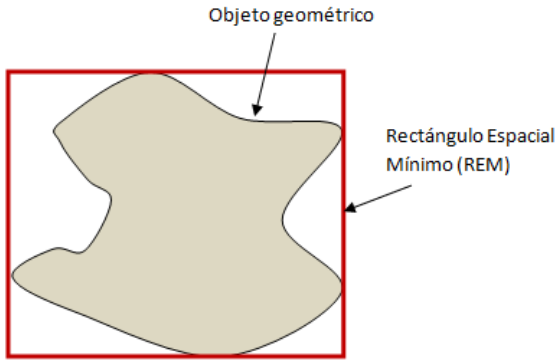

Figura 1: Ejemplo de rectángulo espacial mínimo (REM o MBR)

Para poder maximizar su eficiencia, los índices espaciales usan una estructura de datos en forma de árbol (Fig. 2). Un árbol es una estructura no lineal y dinámica de datos:

- *Dinámica*: puede cambiar durante la ejecución de un programa.
- *No lineal*: a cada elemento del árbol pueden seguirle varios elementos

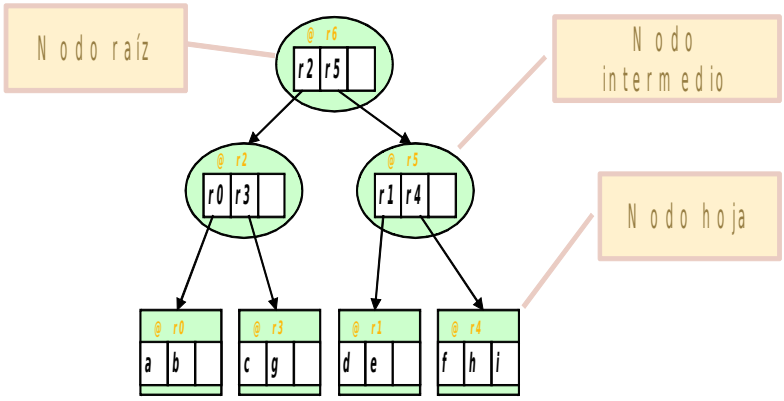

Figura 2: Estructura de datos en árbol

En esta estructura se definen tres tipos de nodos:

- *Nodo raíz* es aquel del que parten todas las conexiones a los demás nodos.

- *Nodo Hoja* son aquellos nodos que no se subdividen más.
- Al resto de nodos se les denomina *Nodos intermedios*.

Para cubrir todo el espacio ocupado por una *capa o feature*, se parte de un nodo raíz, que es el nodo superior. Su *MBR* contiene lógicamente todos los objetos del espacio de búsqueda.

Para encontrar los datos de un objeto dentro de la capa, habría que recorrerlos de forma secuencial. Para acelerar la búsqueda, se subdivide el conjunto de objetos en subconjuntos más pequeños, cada uno de los cuales están contenido en un *MBR*. Cada objeto estará en un solo subconjunto, y los subconjuntos se crean de forma que sus *MBR* se solapen lo menos posible. Si los subconjuntos tienen un número de objetos mayor que una cantidad prefijada dentro del algoritmo, se subdividen. Para que los *MBR* hijos de un *MBR* determinado se solapen lo menos posible, se insertan en el árbol objetos de uno en uno, de forma que al subdividirse en subgrupos éstos contengan cada uno de los objetos más cercanos.

Se tiene entonces que las entradas (*MBR*, *uid*[4]) son los nodos hoja del árbol, ordenado de manera que se permite seleccionar en unos pocos pasos los *MBRs* que contienen los objetos involucrados en la consulta. Por tanto, toda operación que utiliza el índice es ejecutada en dos partes:

- Se recorre el índice para seleccionar los *MBR* que corresponden a la consulta (apuntador, ventana, join, etc.) y extrae los identificadores de los registros que contienen datos.
- Se aplican las operaciones sobre los objetos (intersección y adyacencia de polígonos).

2.2 Algoritmo Árbol-R (R-Tree)

Los índices espaciales pueden clasificarse según la estructura del arbol en que se basan: *QuadTree*, *Árbol-R* (*R-Tree*), *K-d-tree* o Índices *GIST* (*Generalized Search Tree*). A continuación se describirá el tipo de algoritmo incluido en el desarrollo de este trabajo, en concreto, el *Árbol-R*.

El *Árbol-R* no es el algoritmo de indexación más eficiente para datos espaciales [3] ya que su rendimiento se deteriora a medida que aumenta la cantidad de datos y no previene el solapamiento. A pesar de esta limitación, sigue siendo el índice más representativo y está considerado como el más paradigmático y se encuentra disponible en la mayoría de las bases de datos con soporte de indexación de datos espaciales.

Vamos a profundizar a continuación en algunos conceptos relativos al *Árbol-R* que será necesario conocer para el desarrollo del índice. Para una capa con objetos geométricos, un índice *Árbol-R* consiste en un índice jerárquico sobre *REM* de las geometrías de la capa [4]. Veamos ahora cómo se construye un *R-Tree* [5]:

- Cada objeto se representa en el árbol por el identificador del registro (*SRID*) y su *MBR*, en la Fig. 3 los rectángulos rojos.
- Los *MBR* se van agrupando en rectángulos cada vez mayores, por ejemplo el rectángulo punteado en negro de la figura, que contiene los objetos “b” y “c”, que constituyen los nodos internos del árbol. Estos a su vez se agrupan en rectángulos cada vez más grandes, hasta que sólo queda uno, que contiene todos el espacio considerado, o sea, la figura entera (Fig. 3).

4 Identificador

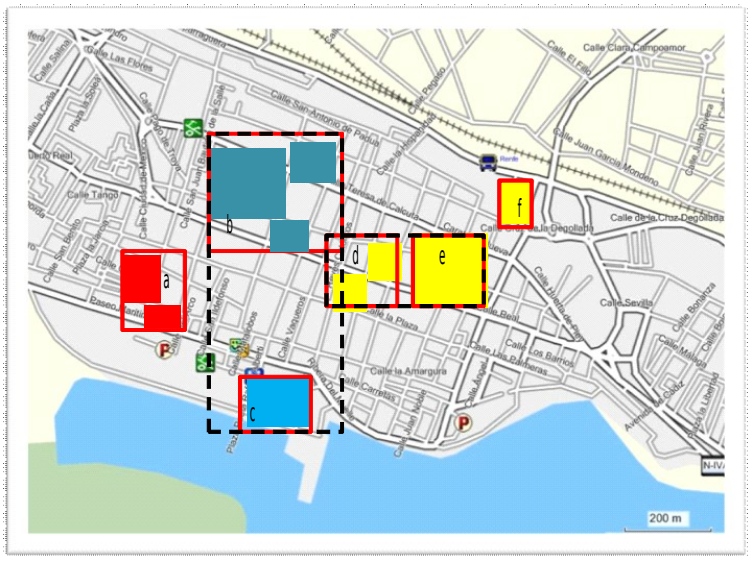

Figura 3: Representación del proceso de creación de un índice jerárquico sobre los MBRs de las geometrías de una capa.

Todos los rectángulos se pueden solapar entre ellos, es decir, pueden tener en común uno o más puntos.

Como el *Árbol-R* es equilibrado, el coste para recuperar el objeto es igual a la altura del árbol. El inconveniente es que a veces es necesario recorrer varios caminos desde la raíz del árbol a las hojas debido a los solapamientos de los rectángulos asociados a los objetos. En estos casos, el rendimiento de la operación de búsqueda se degrada.

3. CARACTERÍSTICAS Y FUNCIONAMIENTO DE H2 Y DE LA EXTENSIÓN JASPA

Una vez descritos los conceptos básicos relativos a la indexación de información espacial, se pasará a describir dos de los componentes externos con los que se integrará la aplicación. En primer lugar, se profundizará en el estudio de *H2* para identificar los recursos que aporta para alojar el índice espacial que se pretende implementar. En segundo lugar, se estudiará la extensión *JASPA* para descubrir las posibilidades de integración con *H2*.

3.1 H2

H2 es una base de datos relacional programada completamente en *Java*. Esto implica que se puede integrar con otras aplicaciones *Java* y acceder a ella ejecutando *SQL* directamente, sin tener que pasar por una conexión a través de *sockets*, como ocurre con otras bases de datos.*H2* también permite crear tablas en memoria y en disco. Además todas las operaciones de manipulación de datos son transaccionales.

3.1.1 Características más destacables

Algunas de las características más interesantes de H2 de cara al presente trabajo son:

- Es compatible con conexiones ODBC y JDBC.
- Dispone de optimizador de consultas.

- Permite columnas calculadas.
- Permite la definición de tipos de datos por el usuario.
- Permite crear tablas en memoria.
- Admite secuencias.

3.1.2 Desarrollo con H2

En cuanto al desarrollo de aplicaciones, H2, las principales características son:

- Soporta la creación de nuevas clases y métodos.
- Permite la definición por parte del usuario de funciones *Java* que a su vez pueden usarse como procedimientos almacenados.
- Permite declarar (registrar) una función antes de usarla y puede definirse usando código fuente, o como una referencia a una clase compilada siempre que se encuentre disponible en el *CLASSPATH*.
- Permite la creación de nuevos tipos de datos.

3.2 Extensión JASPA

JASPA (JAva SPAtial) es una extensión con licencia *GNU GPL* que añade la funcionalidad espacial a bases de datos relacionales que admitan procedimientos almacenados en *Java*. Actualmente *PostgreSQL* y *H2* aunque pueden migrarse a otras bases de datos.

3.2.1 Características más destacables

Las características más destacables de *JASPA* son:

- Está completamente escrita en *Java*.
- Puede usarse con *PostgreSQL* y *H2*.
- Ofrece la posibilidad de migración a otras Bases de datos *Java*.
- Dispone de más de 200 funciones espaciales.
- Es fácilmente extensible usando procedimientos almacenados en Java.
- Utiliza librerías *JTS*[6](*Java Topology Suite*) y *GeoTools*[7]
- Tiene un sistema de referencia espacial.
- Ofrece soporte para los estándares *SHP*, *KML* y *GML* como formatos de entrada.
- Dispone de funciones topológicas propias.

3.2.2 Limitaciones de JASPA

JASPA para *PostgreSQL* provee indexación espacial basada en *PostgreSQL Gist* pero en cambio para *H2* no está disponible esta funcionalidad.

La extensión *JASPA* cuenta con un modelo de datos que da soporte al manejo de datos espaciales desde *H2*. Para la implementación de esta aplicación ha sido necesario ampliar este modelo para que incluya la nueva funcionalidad de indexación

6 JTS:http://www.vividsolutions.com/jts/jtshome.htm
7 Geotools:http://www.geotools.org/

de datos espaciales. Para ello se ha tenido que crear una nueva tabla de datos para almacenar los MBRs de la tabla que contiene los datos geométricos.

4. DESCRIPCIÓN DE LA SOLUCIÓN

El objetivo que se plantea en este trabajo consiste en ofrecer una solución para la creación de un índice espacial para la extensión *JASPA* sobre la base de datos *H2*. Esta solución está limitada a operaciones espaciales en dos dimensiones.

Se ha implementado una solución que permite crear, activar y eliminar índices espaciales con una forma de uso similar a las funciones espaciales proporcionadas por la extensión *JASPA*.

La aplicación permite realizar dos tipos de consultas espaciales que son:

- **De contenido:** dado un rectángulo devuelve los objetos incluidos completamente dentro de él.
- **De intersección:** dado un rectángulo devuelve aquellos objetos incluidos parcial o completamente dentro de él.

Se ha tratado de que la solución que se propone sea lo suficiente flexible como para que no se haya necesitado modificar ni el código fuente de *JASPA*, ni el de *H2*.

Para la implementación del índice espacial se necesitará crear una tabla auxiliar por cada uno de los índices. Esta tabla estará relacionada a través de un identificador con la tabla que contiene los objetos geométricos (Fig. 4).

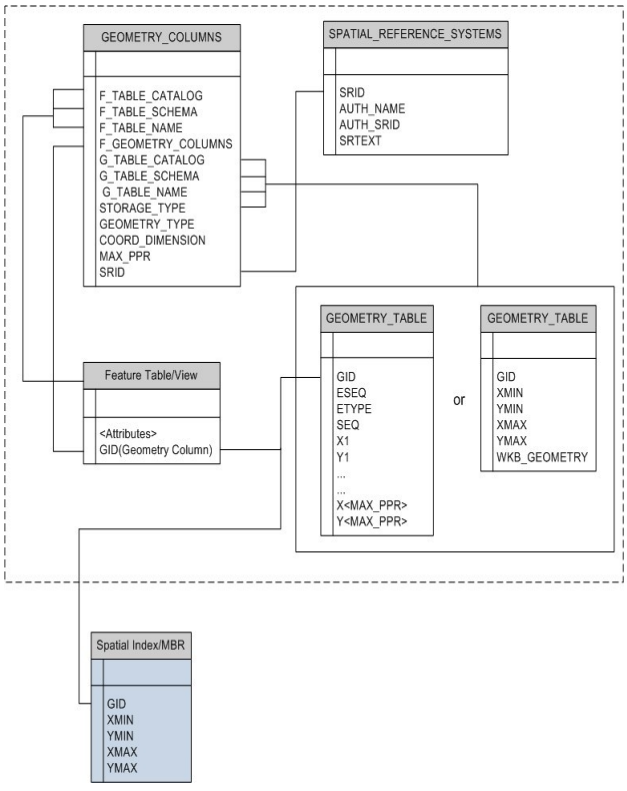

Figura 4: Estructura de tablas de la extensión JASPA (según estándar OpenGIS) que incluye la tabla índice creada.

Esta tabla auxiliar tendrá la siguiente estructura:

- Un identificador que se definirá como clave primaria y que permitirá relacionarla con la tabla que contiene los datos geométricos a través de un *identificador*, clave primaria, que relacionará uno a uno los registros en la tabla original con los datos geométricos y los registros en la tabla auxiliar.

- Cuatro campos que alojarán la definición del rectángulo mínimo (*MBR*) que contiene el objeto espacial. Este rectángulo vendrá definido a partir de las coordenadas *x* e *y* de los puntos que identifican el vértice inferior izquierdo y el vértice superior derecho del rectángulo.

Además, esta nueva tabla auxiliar se creará automáticamente al crear el índice espacial, y deberá estar sincronizada de forma automática con la tabla que contiene los objetos espaciales por medio de los disparadores (*triggers*) que la base de datos *H2* permite construir. De esta forma se persigue que el índice siempre esté actualizado, es decir, que refleje todas las variaciones que la tabla de datos espaciales pueda sufrir (modificaciones, inserciones y borrados de registros).

El *Árbol-R* se generará en memoria a partir de la tabla auxiliar descrita anteriormente cada vez que el usuario introduzca desde consola de *H2* la orden adecuada para activar el índice.

Se optó por no implementar desde cero el algoritmo *Árbol-R*. Después de analizar diferentes implementaciones en *Java* del algoritmo *Árbol-R*, se opta por usar la *JSI (Java Spatial Index) RTree Library*[1]

Se trata de un proyecto de código abierto y liberado bajo la licencia *GNU Lesser General Public License* que tiene como objetivo mantener un alto rendimiento sobre la implementación en *Java* del algoritmo de indexación espacial *Árbol-R*. Esta librería realiza deliberadamente un conjunto reducido de funciones pero lo hace de manera muy eficiente.

Esta biblioteca contiene métodos que permiten:

- Añadir y eliminar entradas en un *Árbol-R*.
- Realizar consultas de intersección y contenido.

La documentación de esta librería asegura la estabilidad de la implementación con diferentes tipos de tests.

El *Árbol-R* se genera en memoria cada vez que se activa un índice, en vez de cargarlo desde la base de datos, que es lo habitual. Esto se ha hecho así por dos motivos:

1. Extraer la información del árbol y transformarla en un formato adecuado para almacenarla en la base de datos consume tiempo de cálculo y la tabla almacenada es más grande que la tabla índice usada.
2. Dado que la tabla índice sólo almacena el *SRID* y el *MBR*, y que además está indexada, la recreación del índice es muy rápida y no necesita cálculos adicionales. Además una vez cargada en memoria se puede utilizar para sucesivas consultas.

Por último, mencionar que se ha implementado una función *Java* en *H2* que incluye los métodos externos que podrá invocar el usuario desde la consola *Sql* de *H2* para el manejo de los índices espaciales. Para facilitar al usuario el uso de éstos métodos se han creado en *H2* unos seudónimos o alias, con un formato similar a las funciones espaciales definidas por la extensión *JASPA*.

4.1 Integración con JASPA y H2

A continuación se expone un ejemplo de cómo la aplicación está integrada con la extensión *JASPA* y *H2*.

La aplicación permite realizar dos tipos de consultas espaciales que son:

- **De contenido:** dado un rectángulo devuelve los objetos incluidos completamente dentro de él.
- **De intersección:** dado un rectángulo devuelve aquellos objetos incluidos parcial o completamente dentro de él.

1 JSI: http://sourceforge.net/projects/jsi/files/jsi/1.0b6/jsi-1.0b6.zip/download?use_mirror=kent

La consulta espacial de intersección desde *JASPA* se realizará en dos fases. En la actualidad sólo está implementada la primera fase en la que se comprueba si intersectan los *MBR* de los objetos geométricos con el rectángulo inicial, es decir, por ahora el índice actúa de filtro. Esto reducirá el número de objetos candidatos sobre los que tendrán que actuar las funciones de *JASPA*, que actuarán en una segunda fase.

Estos identificadores de las geometrías candidatas se almacenan en un *array* que posteriormente se recorre para devolver su contenido como resultado.

Integrando este tipo de consulta espacial con *JASPA* se pueden obtener aquellos objetos que realmente intersectan con la ventana dada. Para ello habría que hacer lo siguiente:

- Una vez que se ha comprobado que el *MBR* de la geometría intersecta con el rectángulo definido, es decir, justo antes de almacenar el *SRID* en el array, sería necesario realizar una consulta para obtener sus correspondientes datos geométricos
- Una vez obtenidos los datos geométricos, se puede invocar a la función *JASPA* **ST_Intersects(bytea GeomA, bytea GeomB)** donde *GeomA* es el rectángulo dado y *GeomB* las geometrías obtenidos a partir de su *SRID*.
- La función devuelve un valor booleano *cierto*" en caso afirmativo y en ese caso se en el array el valor del SRID correspondiente.
- Recorriendo este array se obtendrán los identificadores de los objetos que realmente intersectan con el rectángulo dado originalmente.

4.2 Ejemplos de uso

En los subapartados siguientes se describirán los pasos a realizar para la creación y activación de un índice espacial. Posteriormente se usarán éstos índices para la realización de consultas espaciales y también se mostrará cómo se elimina un índice.

Para la realización de los ejemplos se usarán dos tablas con datos geométricos, llamadas *SOILS* y *PAISES*. Ambas contienen datos geométricos de tipo multipolígonos.

Creación en H2 de un índice espacial

En el ejemplo siguiente se creará una tabla índice para el campo geométrico GEOM de la tabla SOILS:

```
SELECT ST_CREATE_SPIDX('SOILS','GEOM')
```

Activación en H2 del índice espacial

Una vez que se dispone de la tabla índice, se procederá a cargar el Árbol-R en memoria con los datos del fichero índice, eso se realizará desde la consola de H2 con la instrucción siguiente:

```
SELECT ST_LOAD_SPIDX('SOILS','GEOM')
```

La función que activará el índice espacial necesitará como argumentos el nombre de la tabla de datos y el nombre del campo geométrico que se indexó. Es importante tener en cuenta que el nombre de la tabla que se pasa como primer argumento es el nombre de la tabla que contiene los datos y no la tabla que contiene el índice.

Realización de consultas espaciales haciendo uso del índice espacial

Una vez que se ha activado el índice espacial se está en disposición de realizar consultas espaciales desde la consola de H2.

El formato de la función que permite realizar consultas espaciales de ventana es el siguiente:

```
ST_CONTAIN_IDX('nombredelatabladedatos','nombredelcampoindexado',XMIN,YMIN,XMAX
,YMAX))
```

Donde:

- *nombredelatabladedatos*: es el nombre de la tabla de datos que se desea consultar espacialmente.
- *nombredelcampoindexado*: es el nombre del campo sobre el que se desea consultar espacialmente.
- *XMIN,YMIN*: coordenadas *x* e *y* del vértice inferior izquierdo del rectángulo sobre el que se desea localizar los datos geométricos que indique la operación espacial.
- *XMAX,YMAX*: coordenadas *x* e *y* del vértice superior derecho del rectángulo sobre el que se desea localizar los datos geométricos que indique la operación espacial.
- *ST_CONTAIN_IDX* devuelve los identificadores de los datos geométricos que se encuentran dentro del rectángulo delimitados por los pares de coordenadas que definen el vértice inferior izquierdo (*XMIN,YMIN*) y el vértice superior derecho (*XMAX,YMAX*) que se pasan como argumentos a la función *ST_USESPINDEX*.
- Esta primera consulta espacial de ventana se realizará sobre los datos multipolígonos que contiene la tabla SOILS.

A continuación, se muestra un ejemplo de este tipo de consulta espacial que hace uso del índice espacial activado anteriormente:

```
SELECT ST_CONTAIN_SPIDX('SOILS','GEOM',1,1,9000,9000)
```

Esta orden devuelve los identificadores de aquellos datos geométricos de la columna *GEOM* perteneciente a la tabla *SOILS*, que estén contenidos dentro del rectángulo definido por *XMIN,YMIN (1,1)* y *XMAX,YMAX (9000,9000)*.

Eliminación de un índice espacial

El formato de la orden que permite realizar eliminar un índice espacial es el siguiente:

```
ST_DROP_SPIDX('nombredelatabladedatos','nombredelcampoindexado')
```

La orden siguiente eliminaría el índice creado anteriormente, para ello eliminará la tabla con los MBRs, los alias y los disparadores.

```
SELECT ST_DROP_SPIDX('SOILS','GEOM');
```

5. COMPARATIVA ENTRE UNA CONSULTA ESPACIAL QUE USA UN ÍNDICE ESPACIAL Y UNA EQUIVALENTE QUE NO LO USA

En este apartado se tratará de demostrar la eficiencia de la aplicación. Para ello se realizará una misma consulta espacial de dos formas diferentes: la primera de ellas se realizará sin hacer uso del índice y la segunda haciendo uso del índice.

Teniendo en cuenta que el propio gestor *H2* dispone de su propia *caché*, para evitar que haga uso de ella en las consultas que se van a realizar se desactivará previamente.

Para esta demostración se usará una nueva capa llamada *PAISES* de tamaño considerablemente mayor (tanto en número de registros cómo en espacio ocupado) que las usadas anteriormente.

La primera consulta a realizar se muestra a continuación y obtiene los identificadores incluidos dentro de un rectángulo dado, en concreto, el definido por el vértice inferior izquierdo *(1,1)* y el vértice superior derecho *(9000,9000)*.

```
SELECT GID FROM PAISES where st_xmin(GEOM) >1 and st_ymin(GEOM) > 1 and
st_xmax(GEOM)<9000 and st_ymax(GEOM)<9000
```

Es necesario recordar que antes de la ejecución de esta consulta es necesario crear el índice espacial, en este caso se ha omitido este paso ya que se ha descrito con detalle anteriormente.

A continuación se muestra la sentencia equivalente a la consulta anterior pero esta vez haciendo uso del índice espacial.

```
SELECT ID FROM ST_CONTAIN_SPIDX('PAISES','GEOM',1,1,9000,9000)
```

Se ha podido comprobar como la primera consulta consume *301 ms* y la segunda consulta consume *41 ms*. Hay que tener en cuenta que en la segunda consulta se ha despreciado el tiempo que lleva crear la tabla que contiene los *MBR* y el tiempo que se tarda en regenerar el *Arbol-R* en memoria. Ambas operaciones se harán una única vez y servirán para poder realizar un número indefinido de consultas sobre el índice activo.

Tras repetir las mismas consultas en diferentes sesiones, se observa como los resultados son similares y la diferencia de tiempo entre una consulta y otra son del mismo orden.

En la tabla 1 se puede comprobar los resultados de la ejecución de las dos consultas anteriores en cinco sesiones diferentes:

	Tiempo consumido por la consulta 1 (sin usar índice espacial)	*Tiempo consumido por la consulta 2 (usando índice espacial)*
Sesión 1	*301ms*	*41ms*
Sesión 2	*456ms*	*45ms*
Sesión 3	*465ms*	*54ms*
Sesión 4	*442ms*	*41ms*
Sesión 5	*443ms*	*40ms*

Tabla 1: Comparativa de tiempos de respuesta de una consulta espacial con y sin índice espacial.

6. CONCLUSIONES

En este artículo se ha mostrado una aplicación que permite crear, activar, eliminar y utilizar índices espaciales en *H2* con un formato de uso similar a las funciones espaciales de la extensión *JASPA*.

La aplicación, desarrollada completamente con software libre, implementa las siguientes funcionalidades:

- Para los índices espaciales: creación, activación, borrado y uso
- Sincronización automática de la tabla de datos y de la tabla auxiliar del índice.
- Realización de dos tipos de consultas espaciales: búsqueda de objetos que están contenidos o intersectan una ventana dada.

Para la realización de la aplicación ha sido necesario:

- Crear los componentes que implementan las funciones previstas.
- Integrar todos los componentes anteriores, tanto los externos cómo los nuevos, estudiando y creando las interfaces necesarias para ello.

- Implementar los disparadores para que la tabla del índice permanezca siempre sincronizada con la tabla que contiene los datos. Para ello se ha necesitado crear una tabla que almacene los *MBRs* de los datos geométricos y se ha precisado de la implementación de unos *triggers* para que dicha tabla permanezca siempre sincronizada con la tabla que contiene los datos.

Una característica importante de esta solución es que para su desarrollo no ha sido necesario modificar ni el código fuente de *JASPA*, ni de *H2* lo que la hubiera hecho totalmente dependiente de ambas tecnologías y también menos flexible.

Desde el punto de vista de la usabilidad de la aplicación y considerando el contexto de la implementación enmarcado entre diferentes componentes externos, se ha buscado que las funciones creadas sigan el mismo patrón de uso para el usuario final que las ya existentes en la extensión *JASPA*.

También cabe mencionar la posibilidad que incorpora la aplicación para crear uno o varios índices espaciales desde la consola del sistema operativo, sin tener que acceder obligatoriamente a la consola de *H2* y con la posibilidad de hacer un seguimiento al proceso de creación de los índices desde la propia consola a través de los mensajes de *logs* emitidos.

Por último, en el futuro se prevé llevar a cabo las siguientes ampliaciones:

- Adaptar la solución para que trabaje con datos en tres dimensiones y con capas.
- Desarrollar la posiblidad de tener varios índices activo simultáneamente. Actualmente sólo se puede tener uno activo.
- Añadir nuevos tipos de consultas, en lugar de los dos únicos tipos implementados.
- Implementar en H2 la funcionalidad que permita obtener la dirección física de un registro.

REFERENCIAS

- [1] RUIZ GARCÍA, J.M. (2010) Comparativa para la implementación de una extensión espacial en una Base de Datos Java. Proyecto de Fin de Carrera. Dirigido por Jesús Manuel de Diego. Universitat Oberta de Catalunya.
- [2] ZUBIAUR ABRISQUETA, J.G. (2010) Comparativa para la implementación de una extensión espacial en una Base de Datos Java. Proyecto de Fin de Carrera. Dirigido por Jesús Manuel de Diego. Universitat Oberta de Catalunya
- [3] BRISABOA N., LUACES M., NAVARRO G., SECO D. (2006) Indexación espacial de puntos empleando wavelet trees. [en línea]Disponible en: http://www.dcc.uchile.cl/~gnavarro/ps/jisbd09.pdf (última consulta: 08/03/2011)
- [4] PÉREZ A., BOTELLA A., MUÑOZ A., OLIVELLA R., OLMEDILLAS J.C., RODRÍGUEZ J. (2009). Sistemas de Información Geográfica y GeoTelemática.. Universitat Oberta de Cataluña.
- [5] RIGAUX P., SCHOLL M. Y VOISARD A.(2002) Spatial Databases - with Application to GIS [Capítulos 5, 6 y 8]. Ed. Morgan Kaufmann Publishers.