Introducción a PostGIS

Bases de datos geográficas

Anna Muñoz Bollas

PID_00153924

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción 5

1. Instalación de PostgreSQL/PostGIS 9

2. Creación de una base de datos espacial 10
2.1. Añadir un nuevo servidor de base de datos PostgreSQL 10
2.2. Visualizar los contenidos de un servidor PostgreSQL 10
2.3. Añadir una nueva base de datos PostGIS 11
2.4. Visualizar el contenido de una tabla 12

3. Ejecutar expresiones SQL 14

4. Inserción de datos en la base de datos PostGIS 15
4.1. Ejecución directa de sentencias SQL 15
4.2. Importación del formato *shape* 15
4.3. Importación desde gvSIG 16

5. Creación de índices espaciales 18
5.1. Construcción de índices espaciales 18
5.2. Uso de índices espaciales 19

6. Consultas espaciales con SQL 20

7. Anexos 21
7.1. Anexo A: ¿Cómo trabajan los índices espaciales? 21
7.1.1. La construcción de consultas espaciales 21
7.1.2. Comprobación del índice espacial 23
7.1.3. Comentarios adicionales 23
7.2. Anexo B. Funciones y operadores espaciales 24
7.2.1. Funciones de medición 24
7.2.2. Funciones de comparación 24
7.2.3. Utilidades 27
7.2.4. Operadores geométricos 28
7.3. Anexo C. Visualización de datos PostGIS 28
7.3.1. Visualización con gvSIG 28
7.3.2. Visualización con uDIG 29

Introducción

PostGIS es una extensión de la base de datos relacional PostgreSQL que permite almacenar objetos geográficos. PostGIS proporciona a PostgreSQL el cumplimiento de la norma OpenGIS *Simple Features Specification for SQL* (SFSQL)* y la capacidad de almacenar información geo-espacial y de realizar operaciones de análisis geográfico.

* http://www.opengeospatial.org/standards/sfs

PostGIS es *OGC Compliant*; es decir, sigue las normas que marca el Open Geospatial Consortium*. Estas normas están definidas en la primera y segunda parte del documento OpenGIS Implementation Specification for Geographic Information – Simple Features Access –. Versión 1.1.0.

* http://www.opengeospatial.org/resource/products/details/?pid=509

- Part 1: Common architecture (SFA). En esta norma se describe, entre otros aspectos, el modelo de geometría utilizado y la forma de representación de las geometrías (WKT, WKB) y de los sistemas de referencia (WKT).

- Part 2: SQL option (SFS). Versión 1.1.0. En esta norma se describe, entre otros aspectos, el esquema SQL para soportar almacenamiento, consulta

y actualización de objetos espaciales basados en la geometría de dos dimensiones.

PostGIS también incluye soporte para todas las funciones y objetos definidos en las especificaciones del OpenGIS *Simple Features for* SQL. Esta especificación, que data de 1999, es el documento en que se basan las normas ISO SFA y SFS. Como consecuencia, PostGIS contiene múltiples funciones espaciales que permiten realizar procesos de análisis espacial avanzado mediante el uso de consultas SQL.

Nota

En la documentación de la página oficial de PostGIS*, aparece de forma detallada la compatibilidad de todas las funciones espaciales con los estándares del OGC.

* http://postgis.refractions.net/docs/ch04.html

Aunque las normas ISO no son de libre distribución, los documentos SFA y SFS se pueden consultar de forma gratuita en la página web del OGC*, ya que también son especificaciones de esta organización.

* www.opengeospatial.org

PostGIS* ha sido desarrollado principalmente por Refractions Research Inc., como proyecto de software libre con licencia GNU General Public License (GPL). Es un producto muy difundido, con importantes referencias a nivel mundial.

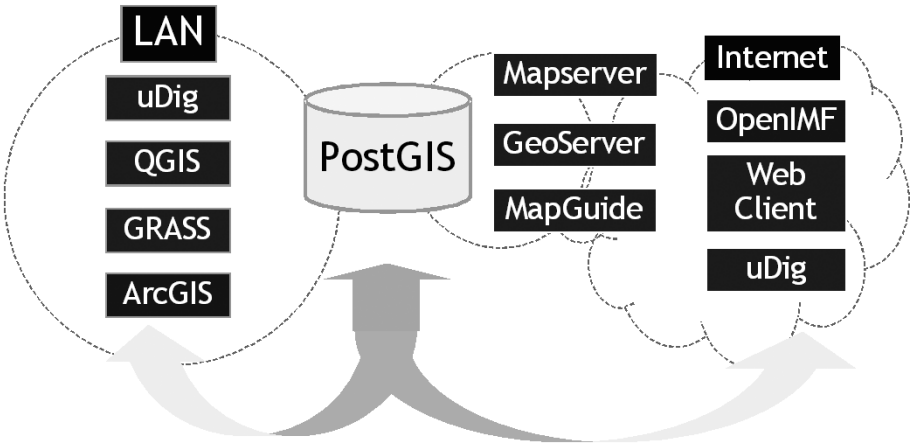
* http://postgis.refractions.net/documentation/casestudies/

Se trata de un proyecto muy activo, en continua evolución*, con recientes incorporaciones, como la segmentación dinámica (LRS) o el cálculo de rutas (todavía bastante básico). Están previstas una serie de mejoras futuras, como la utilización de topología o el soporte de datos *raster*.

* http://www.refractions.net/products/postgis/history/

Aunque visualizar los resultados no es el objetivo principal de una consulta espacial, a menudo ayuda a comprender o explicar un resultado. Para visualizar los datos almacenados en PostGIS, existen diferentes alternativas, como exportar los datos a otro formato, como GML o ESRI *shapefile*, o bien visualizarlos directamente con algún SIG.

Relación de PostGIS con el resto del mundo SIG



Fuente: www.refractions.net

* www.refractions.net

PostGIS está desarrollado en lenguaje C, C++ y PL/pgSQL y utiliza, entre otras, las librerías de software libre GEOS para realizar tests y cálculos geométricos, y Proj4 para realizar las operaciones de reproyección de coordenadas.

Para que os familiaricéis con una base de datos geográficos, en este módulo presentamos las nociones básicas de PostGIS y os proporcionamos una pequeña guía de uso.

Nomenclatura

- PL/PgSQL. Lenguaje procedural del gestor de base de datos PostgreSQL, que permite ejecutar bloques de sentencias SQL, de manera que se proporciona automatismo a las sentencias SQL básicas.
- GEOS*: Geometry Engine Open Source
- PROJ4**: Cartographic projections Library

* http://trac.osgeo.org/geos/

** http://trac.osgeo.org/proj/

1. Instalación de PostgreSQL/PostGIS

La instalación de PostgreSQL y PostGIS se puede realizar sobre MS Windows o Linux.

- Sobre Linux, podéis instalarlo directamente desde Synaptic. Debéis instalar:

– PostgreSQL
– PostGis
– PgAdmin III

- Sobre MS Windows, podéis instalarlo siguiendo cualquiera de las referencias oficiales que se detallan en la página de descarga del software.

Documentación

Podéis consultar la documentación oficial de PostGIS, donde encontraréis un manual muy detallado.

Algunas guías para instalar PostGIS sobre Ubuntu son:

- http://trac.osgeo.org/postgis/wiki/UsersWikiPostgisOnUbuntu
- http://www.paolocorti.net/2008/01/30/installing-postgis-on-ubuntu/
- http://www.merlos.org/documentos/mapserver/34-instalacion-de-postgis-extensiones-gis-de-posrgres.html

Documentación

- Para saber más, se recomienda consultar <http://mapas.topografia.upm.es/geoserviciosOGC/documentacion/WMS/Instalacion_PostgreSQL_PostGIS.pdf>, un excelente documento sobre PostgreSQL y PosGIS, que, además, describe los pasos básicos para la instalación sobre Windows XP.
- Para la instalación sobre Windows Vista, podéis consultar: <http://trac.osgeo.org/postgis/wiki/UsersWikiWinVista>

2. Creación de una base de datos espacial

Para crear una base de datos espacial PostGIS, utilizaremos **pgAdmin III**, que proporciona una interfaz gráfica para la gestión de las bases de datos almacenadas en servidores PostgreSQL.

En este apartado, veremos cómo conectarnos a las bases de datos PostgreSQL, cómo crear una instancia de base de datos postGIS y cómo visualizar el contenido de un SGBD PostgreSQL: bases de datos, tablas, etcétera, utilizando la herramienta pgAdmin III.

2.1. Añadir un nuevo servidor de base de datos PostgreSQL

Al abrir este programa, aparecerá una ventana con el listado de todos los servidores PostgreSQL registrados en pgAdmin III hasta el momento. Si partimos de una nueva instalación, sólo aparece un servidor (*localhost* o máquina local).

En caso de que no aparezca ningún servidor de bases de datos, deberéis añadir uno desde el menú **File** > **Add Server**.

2.2. Visualizar los contenidos de un servidor PostgreSQL

Nos conectaremos al servidor *localhost* haciendo doble clic sobre PostgreSQL Database Server; entonces el sistema solicita la contraseña de usuario *postgres*:

Al validarnos, veremos una serie de objetos que se han creado en la instalación. Los más importantes son los siguientes:

- **Bases de datos**. Son las instancias de bases de datos creadas. Por defecto, se crea la base de datos "postgres" y la base de datos con soporte geo-espacial

"template_postgis". Esta instancia de base de datos contiene las funciones, procedimientos, tipos de datos y demás objetos necesarios para manejar información vectorial geo-referenciada.

- **Esquemas**. Son los contenedores de información de una instancia de base de datos. Una base de datos puede tener varios esquemas. Entre bases de datos no se puede ver la información, pero sí entre esquemas dentro de la misma BD. Por defecto, se crea un esquema *public*.

- **Tablas**. Son las unidades contenedoras de información dentro de un esquema. Por defecto, se habrán creado dos tablas, tal como especifica la norma SFS del estándar OGC.

– "geometry_columns": tabla con información sobre las tablas que contienen alguna columna con geometrías geo-referenciadas,
– "spatial_ref_sys": tabla con información sobre los sistemas de referencia empleados en la geo-referenciación de las geometrías.

Las tablas "geometry_columns" y "spatial_ref_sys" no deberían eliminarse nunca.

Nota

Si no existen, podemos generarlas manualmente mediante la ejecución de los siguientes *scripts* SQL:

- "lwpostgis.sql". Este *script* instala las funciones y los tipos PostGIS en la base de datos creada, así como el segundo lee los sistemas de referencia EPSG en la tabla de sistemas de referencia PostGIS.
- "spatial_ref_sys.sql". Estos *scripts* dependen de la instalación de PostGIS; es decir, no es posible usar los de Linux en Windows o viceversa.

Encontraremos estos *scripts* en: <C:\Archivos de programa\PostgreSQL\8.2\share\contrib>

2.3. Añadir una nueva base de datos PostGIS

Tras instalar el software y comprobar su correcto funcionamiento, crearemos una instancia de base de datos PostGIS mediante la plantilla que viene por defecto: "template_postgis". El uso de esta plantilla garantiza que la instancia de base de datos creada tendrá todos los objetos de PostGIS (las tablas "geometry_columns" y "spatial_ref_sys", funciones, tipos de datos, *casts*…).

Para crear esta instancia, nos situaremos sobre el icono **Bases de Datos** e iremos a **Editar** > **Nuevo Objeto** > **Nueva Base de Datos…** (También podemos hacerlo desde el menú contextual.) Aparecerá el siguiente cuadro de diálogo:

En este cuadro de diálogo, cumplimentaremos los siguientes parámetros:

- **Nombre**. Nombre de la instancia de base de datos.
- **Codificado**. Codificación de los caracteres de la base de datos, en nuestro caso, UTF8, que es la más recomendable.
- **Plantilla**. Emplearemos la plantilla **template_postgis** que viene con la instalación para la generación de base de datos PostGIS.

Una vez introducidos los parámetros, hacemos clic sobre el botón **OK** y ya habremos creado la base de datos.

Un apunte sobre los *table-space*

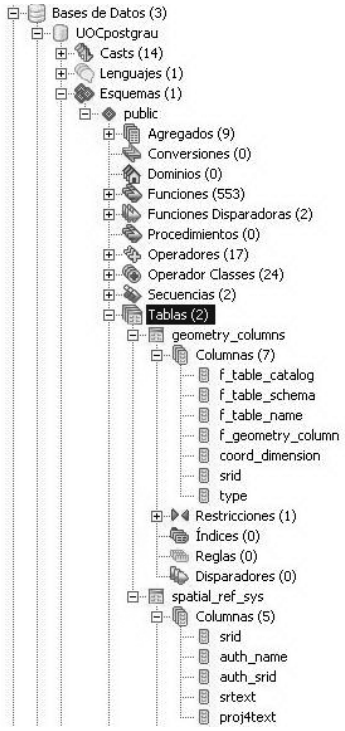
PostgreSQL proporciona dos *table-spaces* por defecto:

- "pg_default"
- "pg_global"

La "pg_global" se usa para tablas del sistema que necesitan estar visibles para el resto del *cluster*. Es conveniente usar "pg_default" para las tablas usuales de trabajo. En caso de no escoger ninguno, por defecto se asigna "pg_ default".

2.4. Visualizar el contenido de una tabla

Para visualizar el el contenido de una tabla seleccionamos **Tablas**. En la siguiente imagen, podemos ver el contenido de las tablas "geometry_columns" y "spatial_ref_sys":



- La tabla "geometry_columns" contiene información relativa a los campos geométricos de nuestra base de datos. Esta tabla aparecerá vacía de contenidos porque aún no hemos creado ningún campo con geometría. La tabla "geometry_columns" se actualiza por las funciones *AddGeometryColumn* y *DropGeometryColumn*.

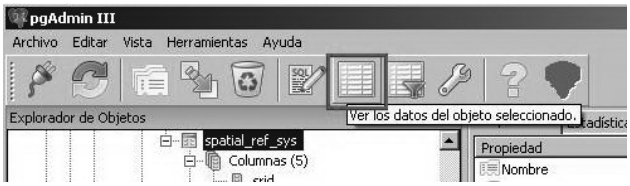
- La tabla "spatial_ref_sys" contiene información descriptiva sobre los sistemas de referencia espacial (SRE) soportados. Cada SRE tiene un identificador único (SRID), que usaremos en nuestras consultas siempre que tengamos que especificar un sistema de referencia. El valor SRID = -1 se utiliza para especificar que los datos no presentan ningún SRE.

Esta tabla también contiene los parámetros que definen cada uno de esos sistemas de referencia, de modo que podemos hacer cambios de proyecciones, *datums* y elipsoides fácilmente.

El estándar utilizado se especifica en el campo "auth_name", que, en el caso de PostGIS, es el EPSG. De esta manera, los campos SRID y AUTH_SRID son coincidentes para todos los SRE definidos.

Además, PostGIS especifica los SRE en el campo PROJ4TEXT, de manera que los entienda la librería PROJ4, que se utiliza para los cambios de proyección.

Para ver el contenido de estas tablas, podéis seleccionar la tabla en la ventana de la izquierda y pulsar el icono **Ver los datos del objeto seleccionado**:



Documentación

El documento OpenGIS Implementation Specification: Coordinate Transformation Services* muestra cómo describir un sistema de referencia mediante WKT.

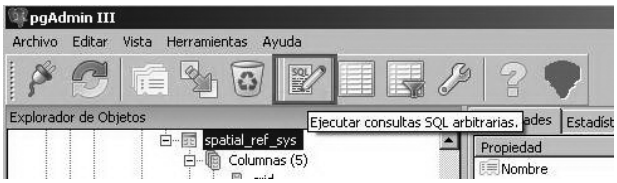
* www.opengeospatial.org

La tabla "spatial_ref_sys" de PostGIS describe, mediante WKT, los sistemas de referencia establecidos por el grupo EPSG** (*european petroleum survey*).

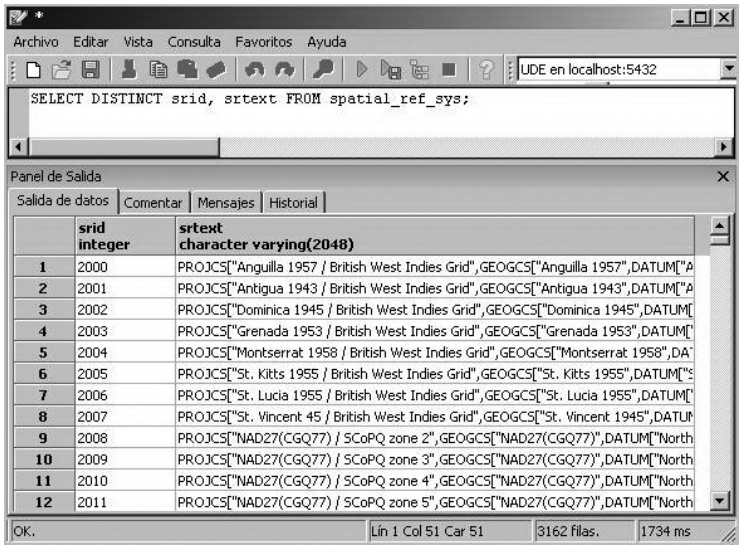
** www.epsg.org

3. Ejecutar expresiones SQL

Para ejecutar expresiones SQL, se ha de hacer clic sobre el icono de la barra de utilidades que lleva por nombre **Ejecutar consultas SQL arbitrarias**.



En la ventana que aparece a continuación, el marco superior se usará para escribir nuestras expresiones SQL y el cuadro inmediatamente inferior servirá para mostrar los resultados de la consulta o, en su defecto, un mensaje de error, describiendo la razón por la cual no se pudo ejecutar el SQL introducido.



En esta ventana, podemos escribir múltiples expresiones SQL, asegurándonos de que cada una de ellas acabe con punto y coma, y pgAdmin III las ejecutará de modo secuencial. En este caso, debemos tener en cuenta que, si tenemos más de una expresión que devuelve una tabla de datos, pgAdmin III sólo mostrará los últimos resultados.

Si sólo queremos ejecutar una de las expresiones que hemos introducido en la ventana, deberemos seleccionar el texto que contiene dicha expresión SQL y hacer clic sobre el icono **Ejecutar consulta**.

Por último, recordad que podemos guardar nuestras expresiones SQL dentro de un archivo en el disco duro para recuperarlas posteriormente.

4. Inserción de datos en la base de datos PostGIS

Tras la creación de una instancia de base de datos PostGIS, realizaremos la inserción de los datos. Esta inserción se puede realizar de diversas maneras:

- mediante la ejecución directa de sentencias SQL,
- mediante el uso de la utilidad "shp2pgsql" de PostGISm
- mediante la exportación de datos desde gvSIG a una base de datos PostGIS.

A continuación se describe cada una de ellas.

4.1. Ejecución directa de sentencias SQL

Una manera sencilla de insertar datos en una base de datos PostGIS es mediante los comandos estándar SQL para bases de datos relacionales. Estos comandos se pueden ejecutar, por ejemplo, desde la ventana de ejecución de comandos SQL de pgAdmin III, en cuyo caso sólo se ha de tener en cuenta que uno de los campos de la tabla contendrá las geometrías, y, por tanto, se ha de especificar en el formato adecuado; es decir, el WKT que ya hemos comentado anteriormente.

4.2. Importación del formato *shape*

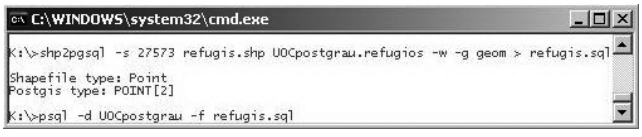
PostGIS dispone de la utilidad "shp2pgsql" para convertir capas en formato *shape* a tablas PostGIS (y viceversa con "pgsql2shp"). Esta utilidad se invoca desde la consola del sistema.

Documentación

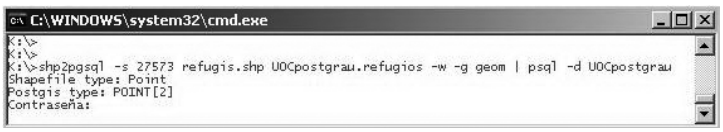
Para ver los detalles de este comando, basta con ejecutarlo en línea de comando sin ningún parámetro, o bien consultar la ayuda en línea para SHP2PGSQL Command Line Options*.

* http://postgis.refractions.net/docs/ch04.html

El comando "shp2pgsl", que se encuentra en el directorio /bin de vuestra instalación de PostgreSQL, crea el código SQL PostGIS para crear la tabla espacial y añadir las geometrías del formato *shape*. Para cargar los datos en PostGIS, usaremos la función "psql" con el archivo SQL generado anteriormente:



Estos dos pasos se pueden realizar en uno sólo, redireccionando la salida, para utilizarla directamente como entrada en el comando "psql" y cargar directamente los datos mediante un *pipe*.



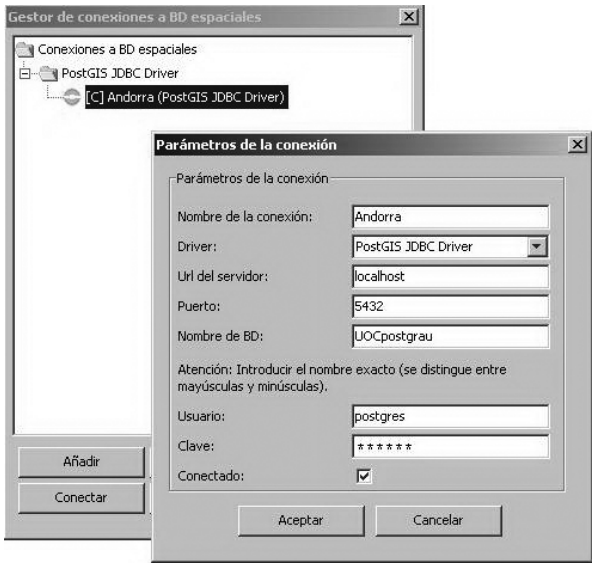
4.3. Importación desde gvSIG

Podemos exportar cualquier tema vectorial que tengamos cargado en un proyecto gvSIG a una BD espacial PostGIS con la que hayamos establecido conexión. A continuación se describen los pasos que hay que seguir.

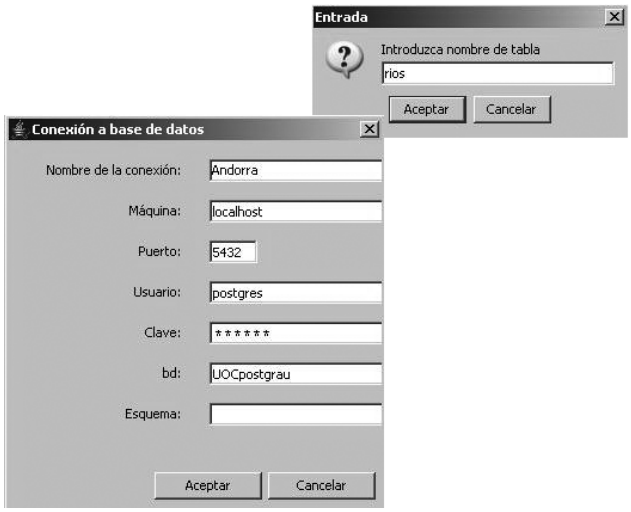
Nota

El gvSIG se verá en la asignatura *Sistemas de información geográfica*. Sin embargo, se incluye este capítulo con el fin de que pueda utilizarse más adelante.

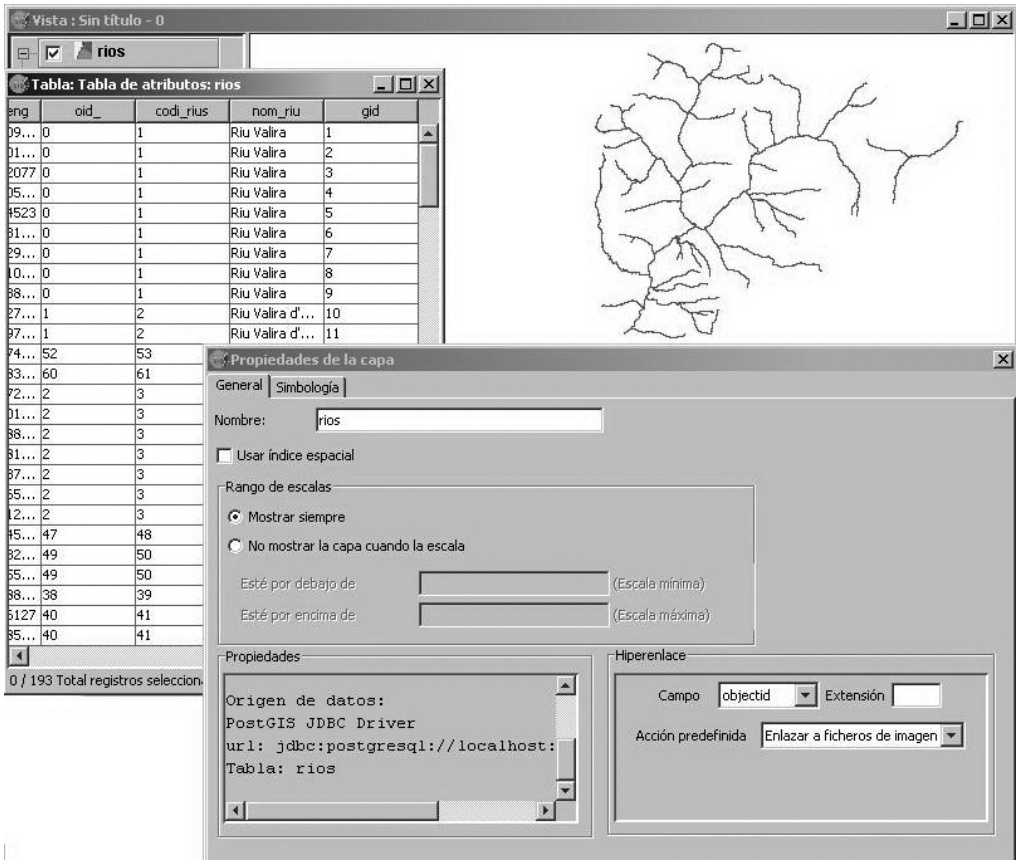
- Establecer conexión con una BD espacial. Desde la ventana principal de gvSIG, usaremos la opción del menú **Ver Gestor de conexiones a BD espaciales**. En la siguiente, imagen se muestran los parámetros que se deben definir para establecer conexión con la BD PostGIS que hemos creado anteriormente.



- Exportar datos vectoriales a PostGIS. Aunque gvSIG es un SIG de escritorio con capacidad para trabajar con datos *raster*, PostGIS no permite almacenar este tipo de datos, por lo que sólo exportaremos datos de tipo vectorial. Comenzaremos creando una nueva vista en gvSIG: cargaremos los *shapefiles* que nos interesen (**Vista** > **Añadir capa** > **Archivo**, **gvSIG SHP driver**) y seleccionaremos la que queramos exportar mediante la utilidad de exportación de gvSIG: **Capa** > **Exportar a**… > PostGIS. Indicaremos el nombre de la tabla que contendrá la información geográfica y alfanumérica del *shapefile*.



- Verificar la exportación. Tras cargar los datos en PostGIS, el programa nos preguntará si deseamos cargar la capa recién creada en la vista. Responderemos afirmativamente a esta cuestión con la finalidad de ver la tabla recién creada, tal como se muestra en la siguiente imagen.



5. Creación de índices espaciales

La creación de índices para una tabla de una base de datos espacial tiene gran importancia, debido a que, normalmente, estas tablas tienen un número de registros considerable y mediante los índices se optimiza el acceso a los mismos: Sin índices, cualquier búsqueda en una base de datos de gran tamaño seria excesivamente lenta (minutos, horas o incluso días), por tener que hacerse de forma secuencial.

Lectura recomendada

Se recomienda una lectura más exhaustiva del concepto de índice espacial en los materiales teóricos y las referencias de la asignatura.

PostgreSQL soporta tres tipos de índices por defecto.

- **B-Tree**. Estos índices son adecuados para datos que se pueden ordenar sobre un eje, como por ejemplo: números, fechas o letras. Los datos GIS no pueden ordenarse sobre un eje.

- **R-Tree**. Estos índices organizan los datos en rectángulos anidados.

- **GIST** (Generalized Search Trees). Estos índices dividen los datos en "cosas que están a un lado", "cosas que se superponen" y "cosas que están dentro". Además, soporta un amplio rango de tipos de datos, incluidos los datos GIS. En PostGIS, esta ordenación es más robusta que la R-Tree.

PostGIS implementa índices R-Tree sobre GIST para indexar los datos GIS.

Organiza los datos en rectángulos anidados para agilizar las búsquedas.

5.1. Construcción de índices espaciales

GIST proporciona un árbol de búsqueda generalizado y una forma generalizada de indexación. Además de proporcionar indexación sobre datos GIS, GIST aumenta la velocidad de las búsquedas con toda clase de estructuras irregulares (datos espectrales, *arrays* de enteros, etcétera) que no se pueden tratar con la indexación basada en B-Tree.

Cuando la tabla espacial excede unos pocos cientos de filas, y si las búsquedas no sólo están basadas en atributos, es recomendable crear un índice para incrementar la velocidad de las búsquedas. La sintaxis para crear este tipo de índices en PostGIS es la siguiente:

```
CREATE INDEX [nombreIndice] ON [nombreTabla]
USING GIST ([campoGeom] GIST_GEOMETRY_OPS);
```

```
CREATE INDEX rios_gidx ON rios USING GIST (the_geom);
CREATE INDEX municipios_gdix ON parroquias USING GIST (the_geom);
```

La creación de un índice es una tarea intensiva a nivel computacional, por lo que es importante que, después de construirlo, PostgreSQL guarde las estadísticas de las tablas. Los índices utilizan estas estadísticas para optimizar los planes de consulta. Para generar y guardar estas estadísticas se utiliza el comando:

```
VACUUM ANALYZE;
```

5.2. Uso de índices espaciales

Para mejorar aún más el rendimiento de los índices espaciales, debemos conocer cómo se implementan y cómo se utilizan en las consultas espaciales.

Nota

Dado que éste no es un tema básico ni imprescindible para el seguimiento de la asignatura, os remitimos al anexo 1 si queréis profundizar en él.

6. Consultas espaciales con SQL

PostGIS implementa la especificación SFA que define un esquema SQL para el acceso a información geo-referenciada en una base de datos relacional. En esta especificación, se definen una serie de operaciones básicas sobre geometrías, almacenamiento, etc.

PostGIS, entre otras, dispone de funciones propias de medición y comparación, utilidades para la consulta de elementos o la validación de geometrías y operadores geométricos, como ST_Area, ST_Perimeter, ST_Length y ST_Distance. También dispone de un buen número de funciones de comparación, como ST_Intersects, ST_Contains, ST_Crosses, ST_DWithin, etcétera, de las que siempre se obtiene como resultado un booleano.

Nota

Para profundizar en el funcionamiento de algunas de las funciones más comunes de PostGIS, podéis consultar el anexo B de este documento. Sin embargo, si queréis un listado exhaustivo y detallado de las mismas, tendréis que consultar el capítulo 7 (secciones 7 y 8) del manual de usuario* de PostGIS.

* http://postgis.refractions.net/docs/ch07.html

Lista de la longitud de los ríos en cada uno de los países por los que pasa:

```
SELECT r.name, p.name, length(intersection(r.the_geom, p.the_geom))
FROM rios r, pais p WHERE crosses(r.the_geom, p.the_geom)='T';
```

7. Anexos

Debido a que el tema de la bases de datos en general, y las bases de datos geográficas en particular, es un tema en el que se puede profundizar desde muchos ámbitos, y teniendo en cuenta a su vez que estos materiales son de propósito general, se incluye en este anexo una serie de temas que son más específicos que el resto de los temas de este módulo, o que, por su temática, se considera que son de un nivel avanzado.

No son, por tanto, de obligada lectura, pero esperamos que sean de ayuda para aquellos que deseéis profundizar en los temas que aquí se exponen.

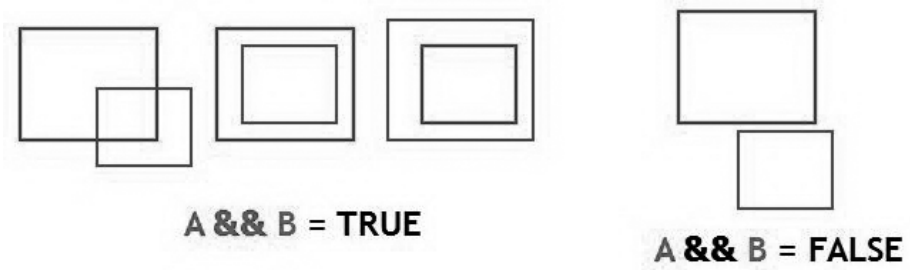
7.1. Anexo A: ¿Cómo trabajan los índices espaciales?

Es importante recordar que los índices espaciales no se usan de forma automática en todas las comparaciones ni en todos los operadores espaciales. De hecho, debido a la naturaleza "rectangular" de los índices R-Tree, los índices espaciales son realmente útiles cuando se realizan comparaciones entre las cajas contenedoras (*bounding box*) de los objetos geométricos implicados en la consulta.

7.1.1. La construcción de consultas espaciales

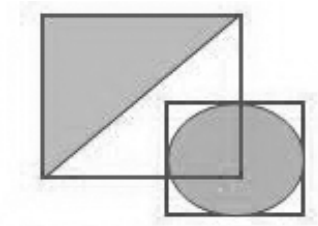
Las bases de datos espaciales implementan las consultas espaciales en dos fases.

- En una primera fase, se indexan los *bounding box* de todos los elementos de las tablas que intervienen en una consulta para reducir candidatos:



  Mediante el uso del operador &&, en una consulta SQL, indicamos a PostgreSQL que estamos utilizando los índices espaciales.

- En una segunda fase, se refina el proceso espacial sólo sobre aquellos elementos obtenidos como resultado de la primera fase. Como podemos ver en la imagen de ejemplo, la consulta sobre los *bounding box* **no** es suficiente.

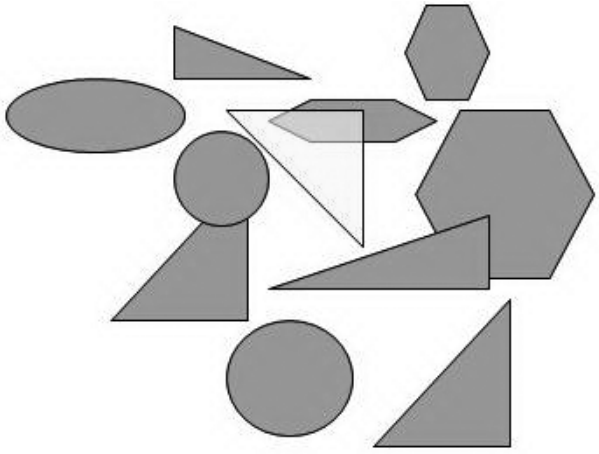


A && B = TRUE
_ST_Intersects(A && B) = FALSE

Las consultas espaciales se llevan a cabo en dos pasos:

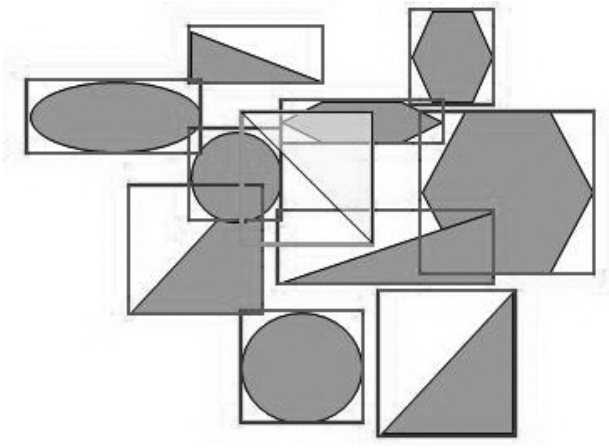
- se reducen los candidatos mediante el uso de los *bounding box*,
- se hace un test topológico para obtener el resultado final.

Ejemplo de intersección en que se usan índices espaciales

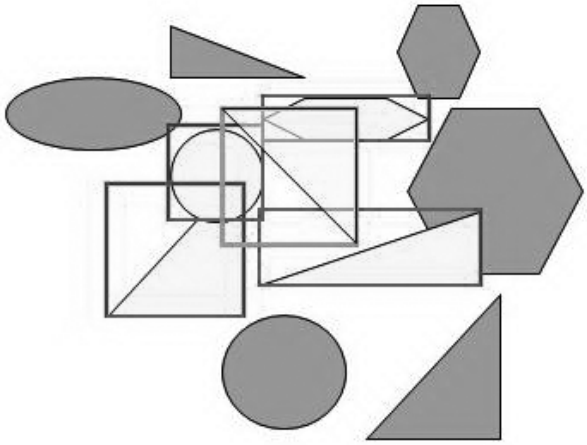


Veremos el proceso de búsqueda de la intersección del triángulo amarillo con el resto de las geometrías. Concretamente, ejecutaremos la siguiente sentencia SQL: `A && B AND _ST_Intersects(A, B)`, o lo que es lo mismo: `ST_Intersects(A, B)`.

- Paso 1 `A && B`. Este paso corresponde al primer paso de la indexación comentado anteriormente, donde se realiza la intersección de los *bounding box* de las geometrías para establecer una primera selección de posibles candidatos.

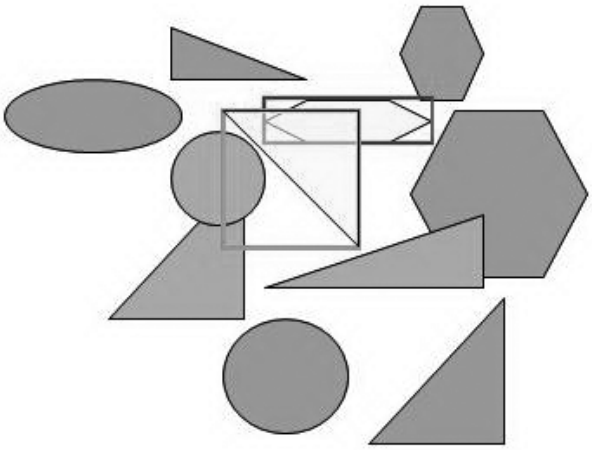


Como resultado de la consulta anterior, obtenemos un subconjunto de las geometrías.



Sobre este resultado, se aplicará el test topológico correspondiente.

- Paso 2 `_ST_Intersects(A,B)`. Se aplica el test topológico para obtener el resultado esperado



Se ha utilizado la función `_ST_Intersects(A,B)`, que no implementa internamente el índice espacial a modo de ejemplo del proceso.

La función `ST_Intersects(A, B)`, al igual que el resto de las funciones de comparación espacial (`ST_DWithin(A, B)`, `ST_Touches(A, B)`, `ST_Contains(A, B)`, etcétera) incluyen la implementación del operador `&&`.

7.1.2. Comprobación del índice espacial

Una buena manera de verificar la utilidad de los índices espaciales y de comparar la mejora del rendimiento de las consultas SQL que los utilizan es la ejecución de una misma sentencia SQL; primero, usando funciones espaciales que implementen los índices; posteriormente usando la misma función que no los implemente (por ejemplo: `ST_Crosses` y `_ST_Crosses`).

Podéis encontrar múltiples ejemplos en la web, pero os aconsejo que hagáis la prueba con el juego de datos que tenéis a vuestra disposición.

7.1.3. Comentarios adicionales

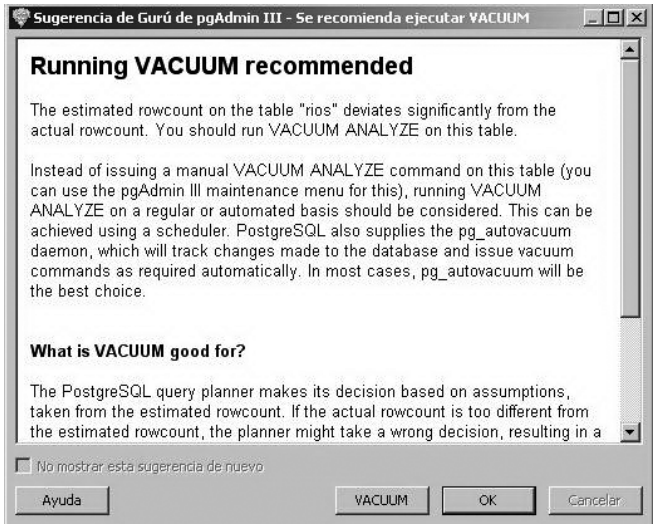
Una vez construido el índice, el planificador de consultas decide cuándo usar el índice de forma transparente. Pero el planificador de consultas de postgreS-

QL no está optimizado para los índices GIST y realiza búsquedas secuenciales aun teniendo el índice.

Si no se están usando los índices GIST en la base de datos, podemos hacer dos cosas:

- Asegurarnos de que hemos ejecutado `VACUUM ANALIZE [nombreTabla]` en las tablas que nos dan problemas. V*acuum analyze* recoge estadísticas sobre el número y distribución de los valores en la tabla; esto ayuda al planificador de consultas a tomar la decisión de usar el índice.

Es una buena costumbre ejecutar *vacuum analyze* de forma regular.



Si el uso de *vacuum analyze* no funciona, debemos forzar al planificador a usar la información del índice con el comando `''SET=OF''`. Este comando ha de usarse con moderación y sólo con consultas espaciales. Una vez terminada la consulta, tenemos que volver a poner: *on* `''ENABLE_SEQSCAN''`, para que se use de forma normal en otras consultas.

Nota

Para ver el rendimiento de vuestras consultas, tenéis la utilidad **Explain** en pgAdmin III, que muestra la representación del planificador.

7.2. Anexo B. Funciones y operadores espaciales

A continuación, se listan las funciones espaciales específicas más usuales de PostGIS.

Dirección web

Encontraréis más detalles sobre las funciones espaciales de PostGIS, así como ejemplos de uso en la web de PostGIS*.

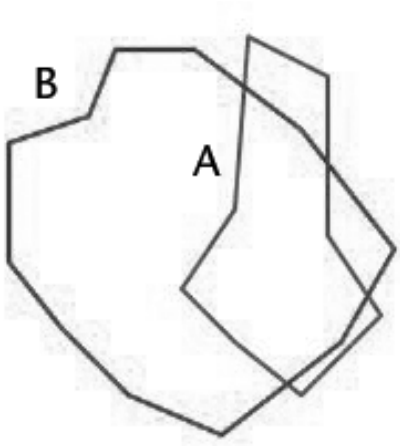
* http://www.postgis.org/documentation/manual-1.4

7.2.1. Funciones de medición

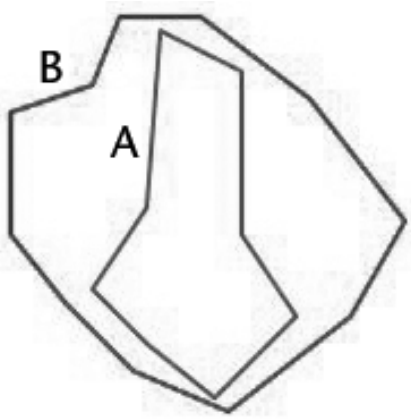
- `ST_Area (geometry) returns Numeric`. Área de la geometría, en las unidades de la proyección.
- `ST_Length (geometry) returns Numeric`. Longitud de la geometría, en las unidades de la proyección.
- `ST_Perimeter (geometry) returns Numeric`. Perímetro de la geometría, en las unidades de la proyección.
- `ST_Distance (geometryA, geometryB) returns Numeric`. Distancia entre la geometría A y la B

7.2.2. Funciones de comparación

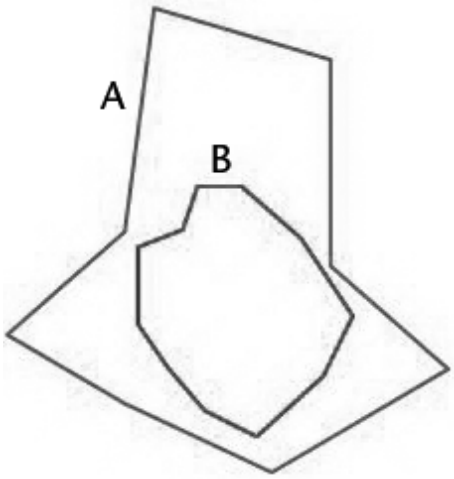
- `ST_Intersects (geometryA, geometryB) returns Boolean`. TRUE si las geometrías hacen intersección.



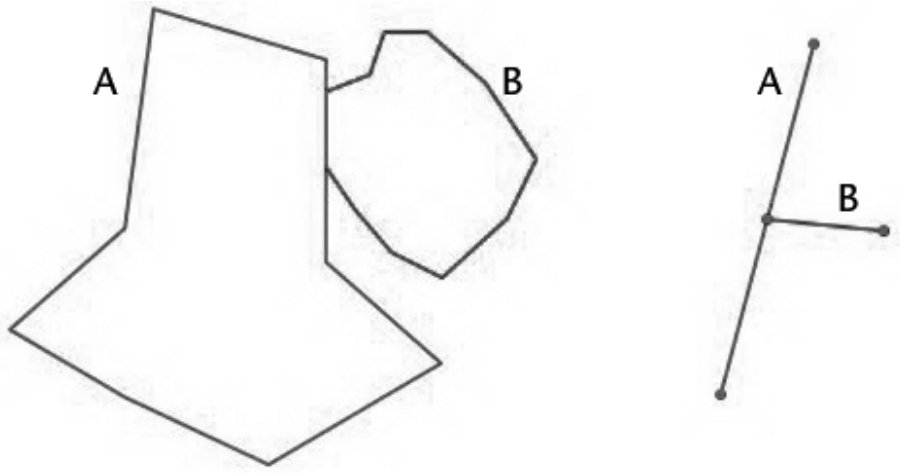
- `ST_Contains (geometryA, geometryB) returns Boolean`. TRUE si la geometría A contiene a la geometría B.



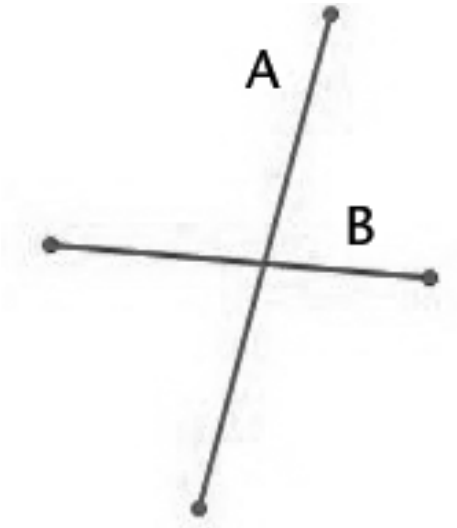
- `ST_Within (geometryA, geometryB) returns Boolean`. TRUE si la geometría A está contenida en la geometría B.



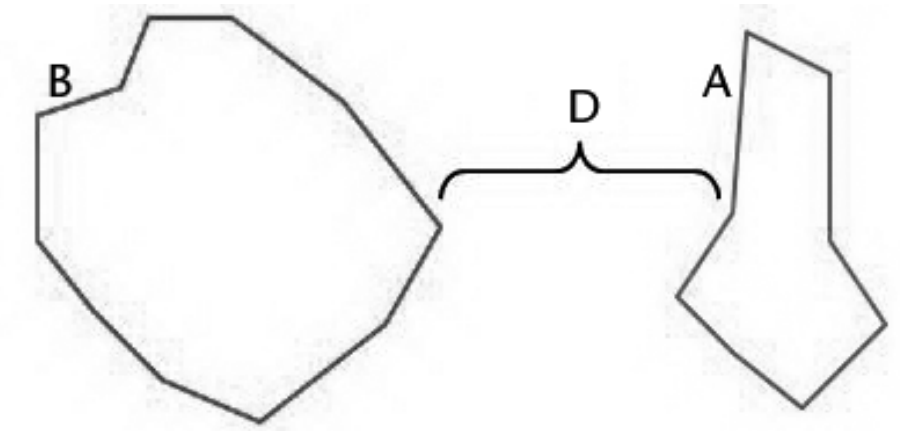
- `ST_Touches (geometryA, geometryB) returns Boolean`. TRUE si las geometrías se tocan sólo en el contorno.



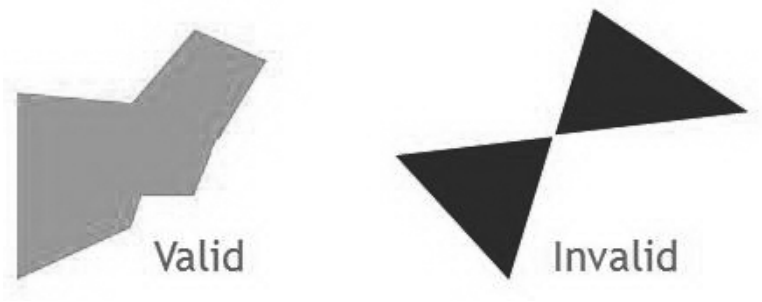
- `ST_Crosses (geometryA, geometryB) returns Boolean`. TRUE si las geometrías se cruzan (muy útil para líneas).



- `ST_DWithin (geometryA, geometryB, distance) returns Boolean`. TRUE si la geometría A y la B están a una distancia < *distance*.



- `ST_IsValid (geometry) returns Boolean`. TRUE si la geometría es válida según las especificaciones OGC.



7.2.3. Utilidades

- `ST_SRID (geometry) returns Integer`. Devuelve el SRID de la geometría.
- `ST_SetSRID (geometry, srid) returns Geometry`. Devuelve la geometría a la que se ha asignado un valor de *srid*.
- `ST_GeometryType (geometry) returns String`. Devuelve el tipo de geometría de un objeto.
- `ST_AsText (geometry) returns String`. Devuelve la geometría en formato WKT; es decir, como texto legible.
- `ST_AsBinary (geometry) returns ByteA`. Devuelve la geometría en binario, como WKB.
- `ST_AsGML (geometry) returns String`. Devuelve la geometría en formato XML.
- `ST_GeometryFromText (geometry) returns Geometry`. Devuelve una geometría que ha leído en formato texto.
- `ST_X (geometry), ST_Y(geometry), ST_Z(geometry) returns Numeric`. Lectura de las coordenadas de un punto.
- `ST_PointN (geometry) returns Point`. Lectura del enésimo punto de un "LineString" o un "LinearRing".
- `ST_NumPoints (geometry) returns Integer`. Devuelve el número de puntos de la geometría.
- `ST_RingN (geometry) returns LineString`. Lectura del enésimo anillo de un polígono.
- `ST_GeometryN (geometry) return Geometry`. Lectura de la enésima componente de una geometría compuesta, como por ejemplo, MULTIPOINT o MULTIPOLYGON.

Nota

Se trata de un proceso de asignación de un valor *srid*, no de transformación o cambio del *srid* de la geometría.

7.2.4. Operadores geométricos

ST_Intersection (geometryA, geometryB) returns Geometry. Devuelve la geometría que se obtiene como intersección de A y B.

ST_Union (geometryA, geometryB) returns Geometry. Devuelve la geometría que se obtiene como unión de A y B.

ST_Buffer (geometryA, distance) returns Geometry. Devuelve la geometría expandida según la distancia especificada.

ST_Expand (geometryA, distance) returns BBOX. Devuelve la caja contenedora de la geometría (*bounding box*) expandida según el valor de distancia especificada.

ST_Centroid (geometry) returns Geometry. Devuelve un punto cercano al centro de la geometría.

Nota

En el caso de geometrías de tipo polígono, este punto no tiene por qué estar contenido dentro de la geometría.

ST_Transform (geometry, srid) returns Geometry. Devuelve una geometría con las coordenadas transformadas al nuevo *srid*.

ST_Simplify (geometry, tolerance) returns Geometry. Devuelve una geometría simplificada según el algoritmo de Douglas/Peucker.

7.3. Anexo C. Visualización de datos PostGIS

Para visualizar las capas de datos geográficos contenidas en una base de datos PostGIS, tenemos diferentes opciones, tanto de software libre como de software propietario.

- Algunas de las soluciones de software libre son: gvSIG, QGIS, uDIG, OpenJump o Kosmo, en cuanto a GIS Desktop, o bien Mapserver, Mapguide o Geoserver, para WebGIS.

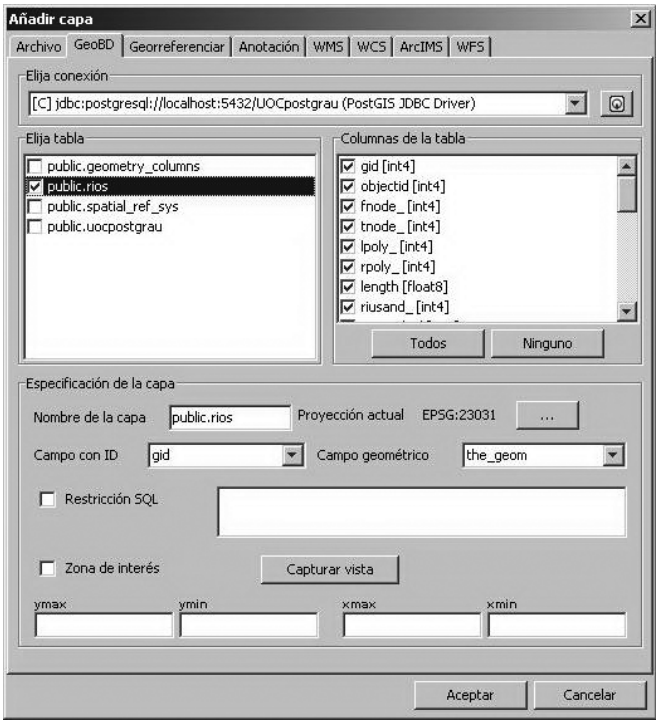
- Por lo que se refiere a software propietario, tenemos arcSDE 9.3.1 o Geomedia 6.1.

A continuación, se describen los pasos para las visualización de los datos PostGIS con dos productos que nos han parecido particularmente interesantes: gvSIG y uDIG.

7.3.1. Visualización con gvSIG

Suponemos que ya hemos establecido conexión con la BD PostGIS, tal como se explicó en el apartado "Inserción de datos en una base de datos PostGIS. Im-

portación desde gvSIG". Para añadir un tema procedente de una fuente de datos JDBC, dentro de la vista en la que estamos trabajando, hacemos clic sobre el icono **Añadir capa** y seleccionamos la pestaña **GeoBD**.



Al seleccionar una conexión, se muestran las capas o tablas disponibles en el catálogo seleccionado. También podemos seleccionar sólo una porción definida por una zona de interés.

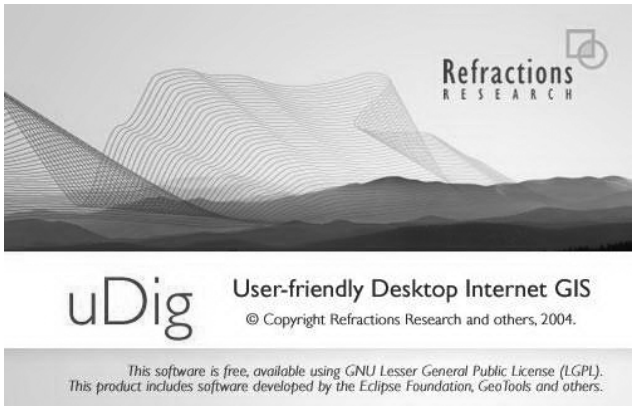
Al seleccionar una tabla, se muestran los campos disponibles. Podemos seleccionar uno, varios o todos los campos.

> Los campos GID y el campo que contiene la geometría son de obligada definición.

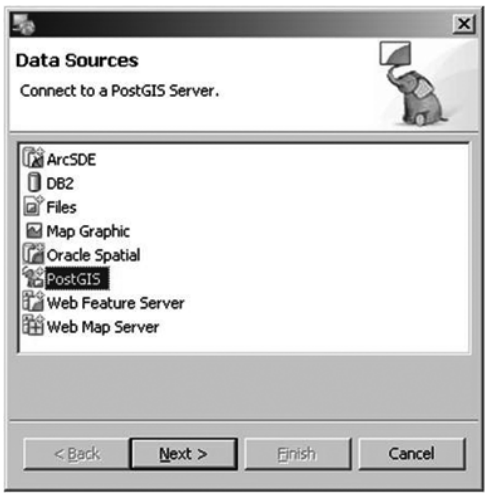
Al pulsar el botón **Aceptar**, se mostrará la cobertura procedente de la base de datos, como una capa más en la TOC de la vista activa.

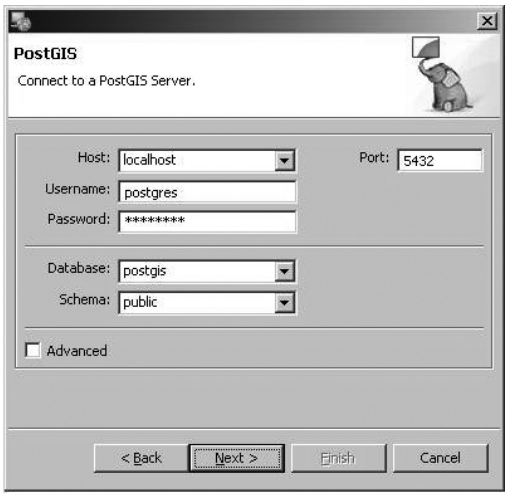
7.3.2. Visualización con uDIG

Para visualizar una capa de datos PostGIS con uDIG, lanzaremos la aplicación desde **Program Files** > **uDig 1.1-RC11** > **uDig**.

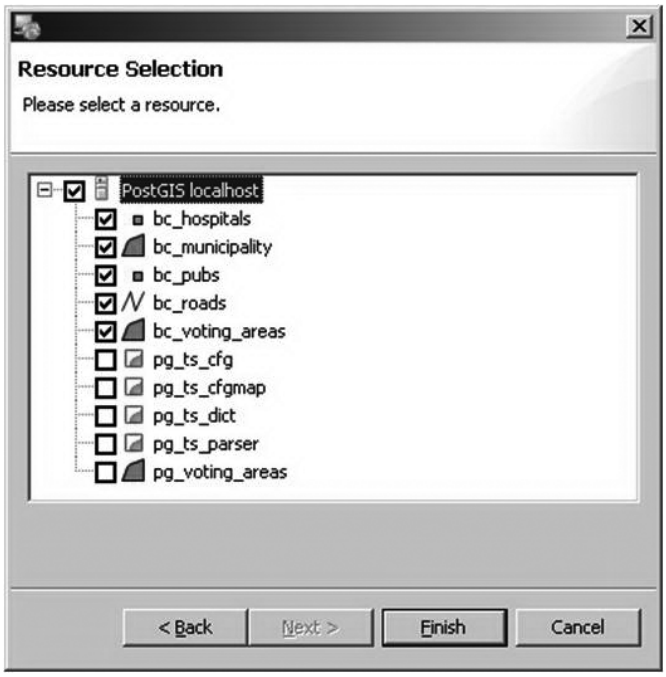


Desde el menú **Files** > **New** > **Layer** seleccionamos la conexión a PostGIS y definimos los parámetros para establecer la conexión con nuestra base de datos.





Al seleccionar una conexión, se mostrarán las capas o tablas disponibles en el catálogo seleccionado.



Finalmente, se mostraran gráficamente las capas seleccionadas: