

CLASE
2

Bases de datos espaciales

República Argentina
Ministerio de Educación
Programa Nacional Mapa Educativo

Curso de capacitación
Bases de datos espaciales

CLASE 2
Material de lectura. Versión 1

**Temas de esta clase:**

Extensión PostGIS.

Tablas de PostGIS: sistemas de referencia y columnas geométricas.

Tipos de geometrías.

Normativas de Open Geospatial Consortium (OGC).

Migración de datos espaciales: desde GVSIG y PGAdmin.

SQL: comandos de actualización. Insertar, modificar y eliminar datos de una tabla.

SQL espacial: consulta de datos espaciales: superficie, longitud, SRID, tipo de geometría, cantidad de vértices y distancias.

Referencias:

A lo largo del documento encontraremos íconos y recuadros que requieren de una especial atención de los lectores:



ACTIVIDADES: son consignas de actividades para realizar la práctica con gvSIG y PGAdmin acompañando la lectura. En la presente clase hay 5 actividades para resolver.



¡IMPORTANTE! Indica una actividad que no debe omitirse para poder desarrollar correctamente la práctica de la clase.

1 Extensión PostGIS

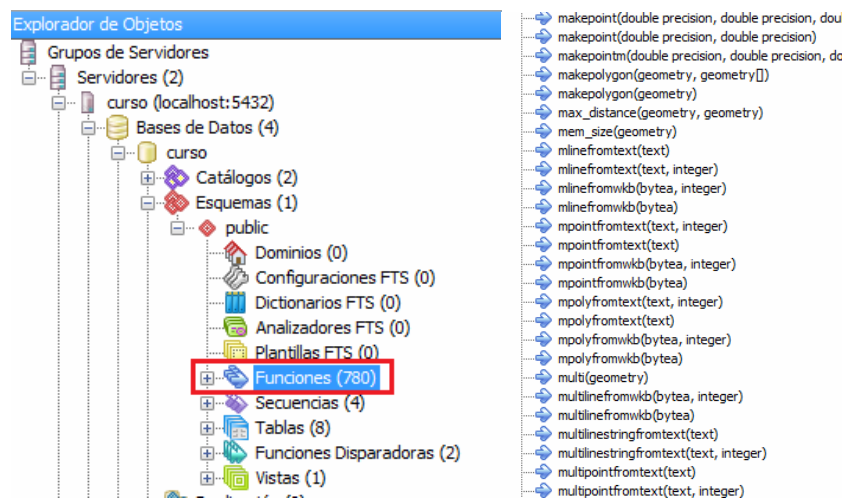
PostGIS es un módulo que se añade al motor de bases de datos Postgres para crear bases de datos espaciales que pueden ser utilizadas para trabajar con SIG. Este

módulo fue creado por la empresa Refraction Research de Canadá, y fue puesto a disposición de la comunidad bajo la licencia pública general de GNU.

En el mundo de las bases de datos existen varios productos similares a PostGIS, como la extensión Spatial de Oracle, o la extensión geográfica de MySQL, pero PostGIS demostró una amplia superioridad en el manejo de datos espaciales. Esto se comprueba al ver que la mayoría de los nuevos productos de SIG incluyen la conexión a PostGIS (ArcGIS).

Además del soporte de datos geométricos (y recientemente, también datos geográficos), PostGIS también añade numerosas funciones relacionadas con el manejo de los datos espaciales, como el cálculo de distancia, superficie, perímetro, reproyección, relaciones topológicas y procedimientos más avanzados de análisis espacial.

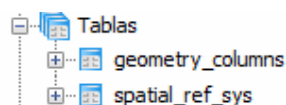
Se pueden explorar estas funciones dentro de la base de datos espaciales que hemos creado en la clase anterior “curso”. Dentro del esquema “public” abrimos el ícono denominado Funciones, y vemos que se despliegan más de 700 objetos.



Cada uno de ellos es una función relacionada con el manejo de datos espaciales. Más adelante veremos como aprovechar este tipo de recursos.

2 Tablas de PostGIS: sistemas de referencia y columnas geométricas

En la clase pasada vimos que al crear la base de datos “curso” usando como plantilla “postgis” o “template_postgis”, se creó con ella un esquema “public” con dos tablas: “spatial_ref_sys” y “geometry_columns”.



La tabla spatial_ref_sys es un repositorio de los sistemas de referencia más utilizados en los Sistemas de Información Geográfica. Recordemos que en la clase 2

del curso de Introducción a los SIG vimos que en el mundo existen diversas formas de proyectar la superficie terrestre sobre un plano, y numerosos modelos matemáticos que representan la forma de la tierra, y que combinados con los sistemas geodésicos de cada región del planeta, dan como resultado infinidad de sistemas de referencia. Recordemos también que existen instituciones que se erigieron como autoridades en la recopilación y codificación de estos sistemas de referencia, y que entre ellos una de las codificaciones más habituales es la impuesta por la EPSG, la Autoridad del Petróleo de Europa.

| | srid [PK] integer | auth_name character varying(255) | auth_srid integer | srttext character varying(2048) | proj4text character varying(2048) |
|------|----------------------|-------------------------------------|----------------------|--|---|
| 2453 | 22174 | EPSG | 22174 | PROJCS["POSGAR 98 / Argentina 4",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-63 +k=1 +x_0=4500000 +y_0=0 +ellps=GRS80 +units=m +no_defs |
| 2454 | 22175 | EPSG | 22175 | PROJCS["POSGAR 98 / Argentina 5",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-60 +k=1 +x_0=5500000 +y_0=0 +ellps=GRS80 +units=m +no_defs |
| 2455 | 22176 | EPSG | 22176 | PROJCS["POSGAR 98 / Argentina 6",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-57 +k=1 +x_0=6500000 +y_0=0 +ellps=GRS80 +units=m +no_defs |
| 2456 | 22177 | EPSG | 22177 | PROJCS["POSGAR 98 / Argentina 7",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-54 +k=1 +x_0=7500000 +y_0=0 +ellps=GRS80 +units=m +no_defs |
| 2457 | 22181 | EPSG | 22181 | PROJCS["POSGAR 94 / Argentina 1",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-72 +k=1 +x_0=1500000 +y_0=0 +ellps=WGS84 +towgs84=0 |
| 2458 | 22182 | EPSG | 22182 | PROJCS["POSGAR 94 / Argentina 2",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-69 +k=1 +x_0=2500000 +y_0=0 +ellps=WGS84 +towgs84=0 |
| 2459 | 22183 | EPSG | 22183 | PROJCS["POSGAR 94 / Argentina 3",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-66 +k=1 +x_0=3500000 +y_0=0 +ellps=WGS84 +towgs84=0 |
| 2460 | 22184 | EPSG | 22184 | PROJCS["POSGAR 94 / Argentina 4",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-63 +k=1 +x_0=4500000 +y_0=0 +ellps=WGS84 +towgs84=0 |
| 2461 | 22185 | EPSG | 22185 | PROJCS["POSGAR 94 / Argentina 5",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-60 +k=1 +x_0=5500000 +y_0=0 +ellps=WGS84 +towgs84=0 |
| 2462 | 22186 | EPSG | 22186 | PROJCS["POSGAR 94 / Argentina 6",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-57 +k=1 +x_0=6500000 +y_0=0 +ellps=WGS84 +towgs84=0 |
| 2463 | 22187 | EPSG | 22187 | PROJCS["POSGAR 94 / Argentina 7",GEOGCS["P... | +proj=tmerc +lat_0=-90 +lon_0=-54 +k=1 +x_0=7500000 +y_0=0 +ellps=WGS84 +towgs84=0 |
| 2464 | 22191 | EPSG | 22191 | PROJCS["Campo Inchauspe / Argentina 1",GEOG... | +proj=tmerc +lat_0=-90 +lon_0=-72 +k=1 +x_0=1500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2465 | 22192 | EPSG | 22192 | PROJCS["Campo Inchauspe / Argentina 2",GEOG... | +proj=tmerc +lat_0=-90 +lon_0=-69 +k=1 +x_0=2500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2466 | 22193 | EPSG | 22193 | PROJCS["Campo Inchauspe / Argentina 3",GEOG... | +proj=tmerc +lat_0=-90 +lon_0=-66 +k=1 +x_0=3500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2467 | 22194 | EPSG | 22194 | PROJCS["Campo Inchauspe / Argentina 4",GEOG... | +proj=tmerc +lat_0=-90 +lon_0=-63 +k=1 +x_0=4500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2468 | 22195 | EPSG | 22195 | PROJCS["Campo Inchauspe / Argentina 5",GEOG... | +proj=tmerc +lat_0=-90 +lon_0=-60 +k=1 +x_0=5500000 +y_0=0 +ellps=intl +units=m +no_defs |

En esta tabla encontraremos un primer campo de tipo auto numérico en donde cada valor es único, y por lo tanto sirve para identificar unívocamente a cada registro. Luego, aparece el campo "srid" que como ya sabemos se corresponde con el número identificador único de los sistemas de referencia. A continuación el campo "srid_authority" almacena el nombre de la autoridad a la que corresponde la codificación del SRID. En los SRID que usaremos en este curso, la autoridad es siempre la EPSG.

Finalmente se presentan dos campos en donde se describen los parámetros que definen a cada uno de los sistemas de referencia, pero escritos según dos normas diferentes. Una de ellas corresponde a la librería de sistemas de proyecciones PROJ4, la más utilizada en los SIG.

| | | | | |
|------|-------|------|-------|---|
| 2457 | 22181 | EPSG | 22181 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-72 +k=1 +x_0=1500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0 +units=m +no_defs |
| 2458 | 22182 | EPSG | 22182 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-69 +k=1 +x_0=2500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0 +units=m +no_defs |
| 2459 | 22183 | EPSG | 22183 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-66 +k=1 +x_0=3500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0 +units=m +no_defs |
| 2460 | 22184 | EPSG | 22184 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-63 +k=1 +x_0=4500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0 +units=m +no_defs |
| 2461 | 22185 | EPSG | 22185 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-60 +k=1 +x_0=5500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0 +units=m +no_defs |
| 2462 | 22186 | EPSG | 22186 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-57 +k=1 +x_0=6500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0 +units=m +no_defs |
| 2463 | 22187 | EPSG | 22187 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-54 +k=1 +x_0=7500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0 +units=m +no_defs |
| 2464 | 22191 | EPSG | 22191 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-72 +k=1 +x_0=1500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2465 | 22192 | EPSG | 22192 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-69 +k=1 +x_0=2500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2466 | 22193 | EPSG | 22193 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-66 +k=1 +x_0=3500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2467 | 22194 | EPSG | 22194 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-63 +k=1 +x_0=4500000 +y_0=0 +ellps=intl +units=m +no_defs |
| 2468 | 22195 | EPSG | 22195 | PROJCS["proj=tmerc +lat_0=-90 +lon_0=-60 +k=1 +x_0=5500000 +y_0=0 +ellps=intl +units=m +no_defs |

proyección latitud de referencia meridiano central factor escala falso este falso norte elipsoide unidades

Es altamente recomendable utilizar los sistemas de referencias y sus codificaciones más difundidos en el mundo de los SIG. Pero puede darse el caso en que necesitemos agregar un sistema de referencia que no fue incluido por Postgres, o bien un sistema de referencia completamente nuevo generado para un proyecto en particular. En estos dos casos, solo tenemos que agregar un nuevo registro a la tabla "spatial_ref_sys" con un SRID que no esté siendo utilizado.

Por ejemplo, el Gobierno de la Ciudad Autónoma de Buenos Aires utiliza un sistema de referencia con las siguientes características:

Sistema de Coordenadas y Proyección: Gauss Krugger Buenos Aires
Datum: Campo Inchauspe; Esferoide: Internacional 1924; Proyección: Transversa

Mercator; Falso Este: 100000; Falso Norte: 100000; Meridiano Central: -58.4627; Latitud de Origen: -34.6297166; Unidad: Metros; Factor De Escala: 0.999998.

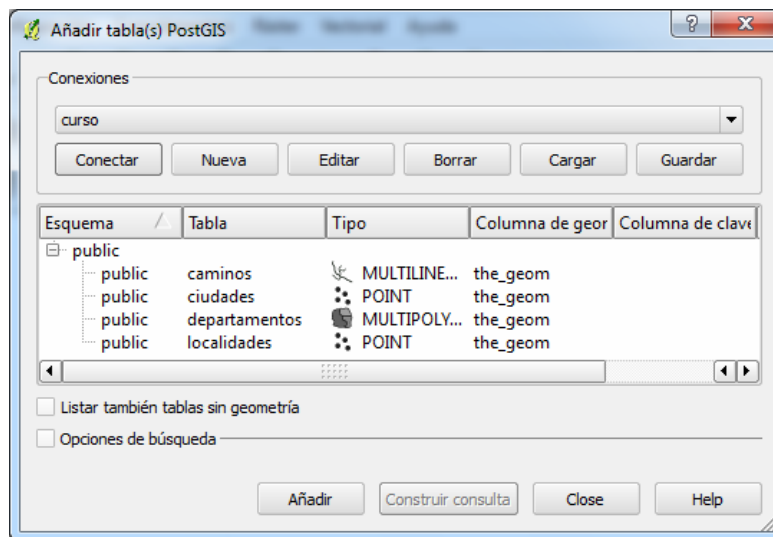
Por su parte, la tabla “geometry_columns”, es empleada como un *registro* de todas las columnas de tipo geometría que existen en la base de datos. Esto es muy útil para que los clientes de la base de datos espaciales (como gvSIG, un servidor de mapas o una aplicación de mapas navegables) accedan rápidamente a los datos geográficos sin explorar toda la base; y además conozcan las características de esos sin tener que analizarlos.

La tabla está compuesta por una primera columna que contiene un identificador único automático de tipo numérico para cada registro. Luego presenta la columna “f_table_schema” en donde se registra el nombre del esquema de la base de datos en donde al que pertenece la tabla.

| | oid | f_table_catalog | f_table_schema | f_table_name | f_geometry_column | coord_dimension | srid | type |
|---|-------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------|---------|-----------------------|
| | | [PK] character varying(256) | [PK] character varying(256) | [PK] character varying(256) | [PK] character varying(256) | integer | integer | character varying(30) |
| 1 | 17409 | " | public | autopistas | the_geom | 2 | 4326 | MULTILINESTRING |
| 2 | 17450 | " | public | caminos | the_geom | 2 | 22183 | MULTILINESTRING |
| 3 | 17425 | " | public | ciudades | the_geom | 2 | 4326 | POINT |
| 4 | 17466 | " | public | departamentos | the_geom | 2 | 22183 | MULTIPOLYGON |
| 5 | 17484 | " | public | localidades | the_geom | 2 | 22183 | POINT |
| * | | | | | | | | |

Luego se registra el nombre de la tabla en “f_table_name”, y el nombre del campo que contiene la geometría: “f_geometry_columns”.

Hasta aquí vemos que con esos datos, por ejemplo un SIG de escritorio nos podría traer sólo esas tablas para poder visualizarlas como capas, en lugar de traernos todas¹. A continuación vemos una imagen de Quantum GIS, otro SIG de escritorio que ha implementado esta funcionalidad.




A continuación veremos el resto de los campos que registran algunas características muy importantes de cada campo geométrico.

¹ GvSIG aún no implementó esta mejora en su versión 1.11. Quantum GIS sí lo hace y esto resulta sumamente práctico.

El campo “srid” almacena el número correspondiente con el sistema de referencia en que se encuentran proyectados los datos. El campo “dimension” es en donde se figura si la geometría corresponde a una representación en 2 o 3 dimensiones. Finalmente, el campo “geometry_type” indica el tipo de geometría que hay en ese campo. De este último hablaremos más adelante en esta misma clase.

Generalmente, las tablas de información geométrica provienen de la migración de archivos shapefile a la base de datos, y el mismo proceso de migración añade el registro de la nueva columna geométrica a la tabla “geometry_columns”. Pero en caso de que la nueva columna de información geométrica no sea el resultado de una migración, sino de un procesamiento con funciones de SQL espacial, entonces deberemos agregar nosotros mismos el nuevo registro en la tabla.

La agregación del nuevo registro se logra abriendo la tabla con el botón *Ver los datos del objeto seleccionado* , y luego editando el último registro de la tabla que se encuentra en blanco e indicado con un asterisco *.

| oid | f_table_catalog | f_table_schema | f_table_name | f_geometry_name | coord_dimension | srid | type | |
|-----|-----------------|----------------|--------------|-----------------|-----------------|------|-------|--------------|
| 4 | 17466 | " | public | departamentos | the_geom | 2 | 22183 | MULTIPOLYGON |
| 5 | 17484 | " | public | localidades | the_geom | 2 | 22183 | POINT |
| * | | | | | | | | |

Cabe aclarar que una misma tabla puede poseer más de un campo con geometrías. Por ejemplo, puede ser que por economía de los recursos de procesamiento, decidamos almacenar la capa de usos del suelo de la Provincia de Santa Fe con el SRID 4326 (en geográficas) y también en 22185 (en faja 5). Para tener estos dos datos, no hace falta crear dos tablas diferentes, sino una misma tabla con dos campos geométrico. Veremos más adelante que se le puede agregar un *disparador* para que la actualización de un dato en cualquiera de los dos campos impacte en el otro.

En ese caso el registro en la tabla geometry_columns quedaría de esta manera:

| | oid | f_table_catalog [PK] character | f_table_schema [PK] character | f_table_name [PK] character | f_geometry_name [PK] character | coord_dimension integer | srid integer | type character varying |
|---|-------|-----------------------------------|----------------------------------|--------------------------------|-----------------------------------|----------------------------|-----------------|---------------------------|
| 1 | 17409 | " | public | autopistas | the_geom | 2 | 4326 | MULTILINESTRING |
| 2 | 17450 | " | public | caminos | the_geom | 2 | 22183 | MULTILINESTRING |
| 3 | 17425 | " | public | ciudades | the_geom | 2 | 4326 | POINT |
| 4 | 17466 | " | public | departamentos | the_geom | 2 | 22183 | MULTIPOLYGON |
| 5 | 17484 | " | public | localidades | the_geom | 2 | 22183 | POINT |
| 6 | 17492 | | public | uso_suelo | the_geom | 2 | 4326 | MULTIPOLYGON |
| 7 | 17493 | | public | uso_suelo | geom2 | 2 | 22183 | MULTIPOLYGON |
| * | | | | | | | | |

3 Tipos de geometría

La clase anterior incluyó un ejercicio para visualizar en un lenguaje comprensible el contenido de los campos geométricos. Vimos que en realidad, el dato geométrico de la entidad representada equivale al conjunto de coordenadas que componen la figura geométrica. Los tipos de datos geométricos están relacionados con esta descripción de las coordenadas.

El tipo POINT supone un solo punto, representado por una sola coordenada:

$(-66.5 \ -32,6)$



Esto podría utilizarse para representar a una ciudad a una escala nacional, por ejemplo.

El tipo LINE implica dos coordenadas que representan a una recta, por ejemplo un tramo de ruta recta entre dos ciudades:

$(-66.5 \ -32.6, \ -64.6 \ -45.3)$

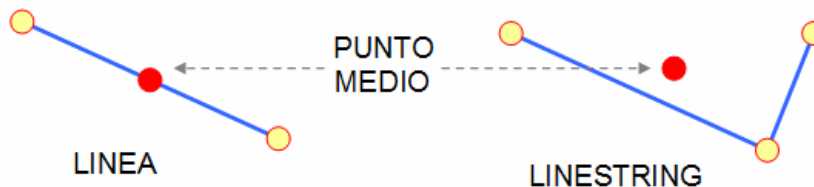


Luego, el tipo LINESTRING también es una línea, pero que puede tener más de dos coordenadas. Es decir, que además del punto de inicio y del final de la línea, necesitamos más datos para poder representarla.

$(-66.5 \ -32.6, \ -65.2 \ -40, \ -64.6 \ -45.3)$



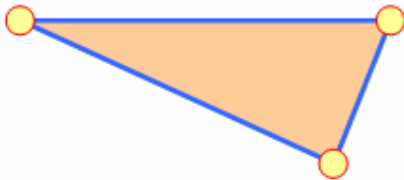
Esta distinción entre líneas y cadena de líneas (linestring) se debe a que existen funciones que se pueden aplicar a una, pero no a la otra, por las diferentes propiedades que tiene cada una en cuanto a figuras geométricas. Por ejemplo, calcular el punto medio de una recta es sencillo, pero pretender hacer lo mismo con una LINESTRING nos puede dar como resultado un punto fuera de la figura.



El tipo POLYGON requiere la presencia de al menos tres coordenadas, que generan un triángulo, que es el más sencillo de los polígonos. Un polígono implica una porción del espacio que queda comprendida dentro de la figura, y esto es muy importante para tener en cuenta a la hora de decidir representar las entidades del terreno.

Por ejemplo, difícilmente se pueda representar una serie de ciudad a escala nacional por medio de polígonos, pero si empleamos una escala con mayor detalle quizá debamos decidir qué tipo de geometría emplearemos. En este caso, los objetivos del proyecto nos pueden ayudar: si necesitamos conocer la superficie del aglomerado, o calcular la cantidad de metros de ruta que atraviesan la ciudad, necesitaremos un polígono que nos brinde sus propiedades para realizar estos cálculos.

(-66.5 -32.6, -65.2 -40, -64.6 -45.3)



Finalmente, veremos las geometrías compuestas: MULTIPOINT, MULTILINESTRING, MULTIPOLYGON.

Muchas veces una entidad del terreno se puede representar con un conjunto de figuras geométricas del mismo tipo, pero como entidad de nuestras bases deseamos que se constituyan en un solo registro dentro de la tabla correspondiente. Por ejemplo, la provincia de Tierra del Fuego, Antártida e Islas del Atlántico Sur en realidad es un solo registro en la tabla "provincias", pero dicho registro está representado por varios polígonos (Isla Grande de Tierra del Fuego, isla de los Estados, Isla Soledad, Isla Gran Malвина, la Antártida, las Orcadas, etc.). En realidad se trata de un multipolígono, y se almacena como tal en la tabla.



Tabla: Tabla de atributos: divisio...

| NOM_PROV | COD_PROV |
|------------------------|----------|
| MENDOZA | 50 |
| LA RIOJA | 46 |
| SAN JUAN | 70 |
| TUCUMAN | 90 |
| SANTIAGO DEL ESTERO | 86 |
| CATAMARCA | 10 |
| BUENOS AIRES | 06 |
| CIUDAD DE BUENOS AIRES | 02 |
| SANTA CRUZ | 78 |
| CHUBUT | 26 |
| TIERRA DEL FUEGO | 94 |

1 / 24 Total registros seleccionados.

Otro caso puede ser la Ruta Nacional 3, que comprende el tramo continental y el tramo de Tierra del Fuego. Si el proyecto necesita que cada ruta sea un registro solo (por ejemplo, porque tiene un único presupuesto y autoridad a cargo), no tiene sentido almacenar la geometría con una lógica distinta al resto del proyecto.

La clave para decidir entre una figura múltiple o figuras sencillas por separado, es que se adapten a la lógica del resto de la información que tenemos en la base. Si cada tramo de la ruta implica datos diferentes, entonces lo correcto es crear más de un

registro. Si no es así, al dividir los registros solo lograremos dificultar la relación entre el aspecto espacial y el alfanumérico de los mismos datos.

**ACTIVIDAD 1**

Crear un documento de texto para luego subir al campus.

Restaurar los archivos de resguardo (.backup) presentes en el material de práctica de esta clase.

Responder. ¿En qué casos usaría puntos, líneas y polígonos para trabajar con la información que gestiona en su propia institución?

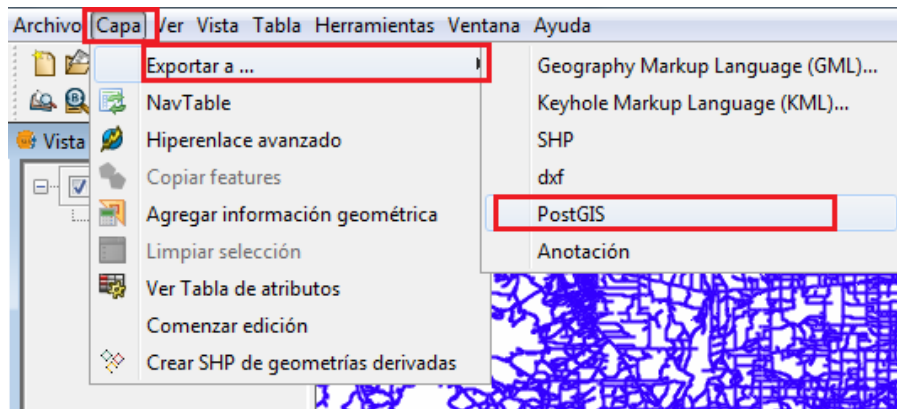
¿Usaría también figuras múltiples? ¿En qué casos?

4 Migración de datos espaciales: desde GVSIG y PGAdmin.

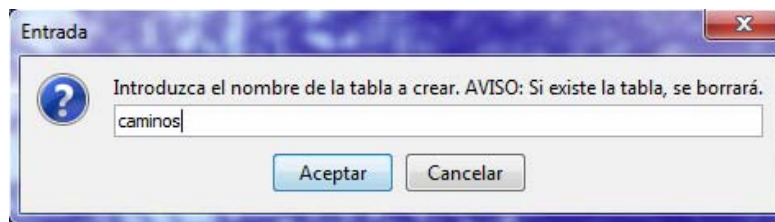
Existen diferentes alternativas para realizar la migración a Postgres de datos espaciales almacenados en formatos shapefile (u otro previamente convertido a shapefile). Por un lado gvSIG y muchos otros SIG de escritorio disponen de herramientas muy sencillas y eficaces. Por otro lado PGAdmin también sumó un plugin para realizar la migración del shapefile o solo de su tabla de atributos.

4.1 gvSIG

Para migrar desde gvSIG es necesario abrir la capa en una vista. Es muy importante que el SRID de la capa esté correctamente configurado². Luego en el menú Capa se selecciona la opción *Exportar a...* y luego *PostGIS*.

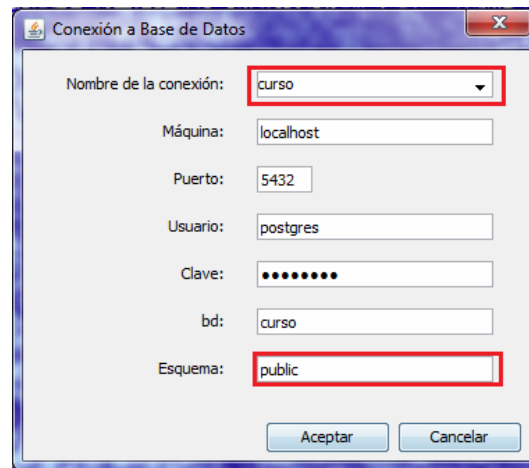


Allí nos aparecerá una ventana en donde colocaremos el nombre que le queremos dar a la nueva tabla en la base de datos. Es importante estar seguros de que no estamos definiendo el nombre de una tabla existente.



² Ante cualquier duda, es conveniente consultar la clase 2 del curso introductorio.

Luego, pasaremos a otra ventana en donde colocaremos los datos ya conocidos para conectarnos a la base de datos: usuario, contraseña, servidor, puerto, etc.

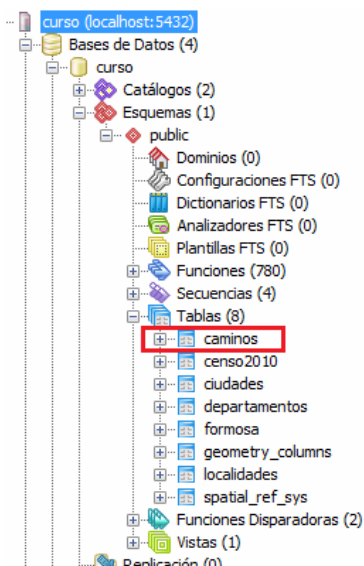


Es muy probable que dentro del menú desplegable nos aparezcan los datos ya utilizados. En caso de que no aparezcan, podemos guardar la configuración a la conexión desde *Ver, Gestión de conexión a bases de datos espaciales*.

A estos datos guardados deberemos agregarle el nombre del esquema de la base de datos a la que deseamos migrar la información. Como por ahora solo tenemos el esquema creado por defecto, allí colocamos “public”.

Dependiendo del tamaño de la capa, el proceso se puede completar en el instante, o bien tomarse un tiempo mayor, lo cual será reflejado a través de una barra de estado.

Una vez finalizada la migración, podemos volver al PGAdmin y comprobar la existencia de la nueva capa.



También podemos ver que en la tabla `geometry_column` se ha sido añadido un nuevo registro correspondiente al campo geométrico que terminamos de crear.

| Archivo Editar Vista Herramientas Ayuda | | | | | | | | |
|---|-------|------------------------|------------------------|------------------------|------------------------|-----------------|---------|-------------------|
| | oid | f_table_catalog | f_table_schema | f_table_name | f_geometry_name | coord_dimension | srid | type |
| | | [PK] character varying | [PK] character varying | [PK] character varying | [PK] character varying | integer | integer | character varying |
| 1 | 17409 | " | public | autopistas | the_geom | 2 | 4326 | MULTILINESTRING |
| 2 | 17450 | " | public | caminos | the_geom | 2 | 22183 | MULTILINESTRING |
| 3 | 17425 | " | public | ciudades | the_geom | 2 | 4326 | POINT |
| 4 | 17466 | " | public | departamentos | the_geom | 2 | 22183 | MULTIPOLYGON |
| 5 | 17484 | " | public | localidades | the_geom | 2 | 22183 | POINT |
| * | | | | | | | | |

4.2 Plugin de PGAdmin

La otra forma de migrar un archivo shapefile es a partir del plugin de PGAdmin Shapefile and DBF loader. Este plugin se instala automáticamente durante la instalación de Postgres y el módulo PostGIS en el servidor. Para las restantes máquinas, es necesario instalarlo manualmente.

4.2.1 Instalar Shapefile and DBF loader en PGAdmin



ATENCIÓN

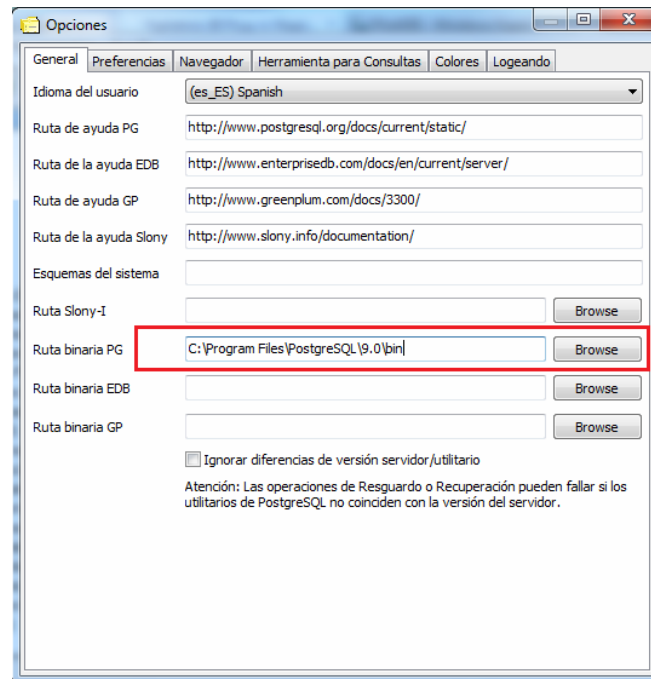
Este procedimiento no es sencillo. Es mejor solicitar ayuda de informáticos para asegurarnos de que están correctamente realizados los procedimientos. Si no estamos seguros de poder hacerlo correctamente, podemos optar por la migración desde gvSIG.

Para instalar el complemento, primero necesitamos descargarlo desde:

http://www.postgis.org/download/windows/extras/postgisgui_pgadmin-2.0.0svn.zip







Luego, abrimos PGAdmin vamos al menú *Archivo*, y seleccionamos *Opciones...*

Allí identificamos la ruta de la carpeta donde se almacenan los binarios de PGAdmin.



Una vez identificada la carpeta de los binarios, nos dirigimos hacia ella y pegamos allí el archivo descomprimido que acabamos de descargar.

Luego, abrimos como archivo de texto (Botón derecho, *Editar*) el archivo `plugins.ini` que se encuentra en la misma carpeta

| | | | |
|---|---------------------|-----------------------|---------------------------|
|  | docs | 10/09/2011 10:32 ... | Carpeta de archivos |
|  | i18n | 10/09/2011 10:33 ... | Carpeta de archivos |
|  | plugins.ini | 23/12/2009 11:13 a... | Opciones de configuración |
|  | plugins.ini.bak | 03/10/2010 04:03 ... | Archivo BAK |
|  | plugins.postgis.ini | 23/12/2009 11:13 a... | Opciones de configuración |
|  | settings.ini | 10/09/2011 10:35 ... | Opciones de configuración |

En la parte final del archivo agregamos las siguientes líneas de código:

```

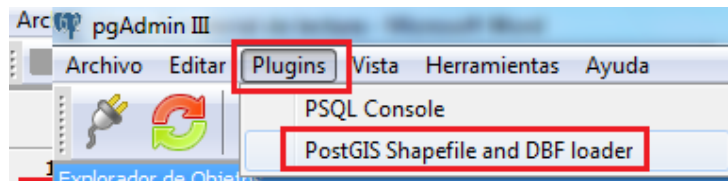
;
;PostGIS shp2pgsql-gui (Windows):
;
Title=PostGIS Shapefile and DBF loader
Command="$$PGBINDIR\postgisgui\shp2pgsql-gui.exe"           -h
"$$HOSTNAME" -p $$PORT -U "$$USERNAME" -d "$$DATABASE" -W
"$$PASSWORD"
Description=Open a PostGIS ESRI Shapefile or Plain dbf loader
console to the current database.
KeyFile=$$PGBINDIR\postgisgui\shp2pgsql-gui.exe
Platform=windows
ServerType=postgresql
Database=Yes
SetPassword=Yes

```

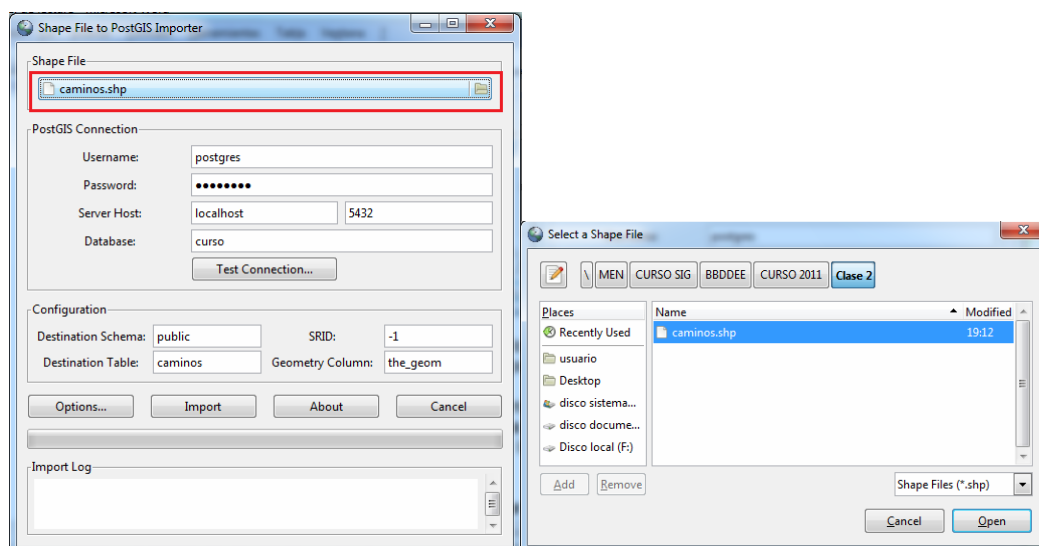
Una vez agregado el código, guardamos y cerramos el archivo. Para comprobar los cambios, cerramos y volvemos a abrir PGAdmin. Deberá aparecer la opción *PostGIS Shapefile and DBF loader* dentro del menú *Plugins*.

4.2.2 Utilizar Shapefile and DBF loader

Se activa desde PGAdmin una vez conectados con la base de datos correspondiente. Hacemos clic en el menú *Plugins* o *Complementos*, y allí seleccionamos la opción *PostGIS Shapefile and DBF loader*.

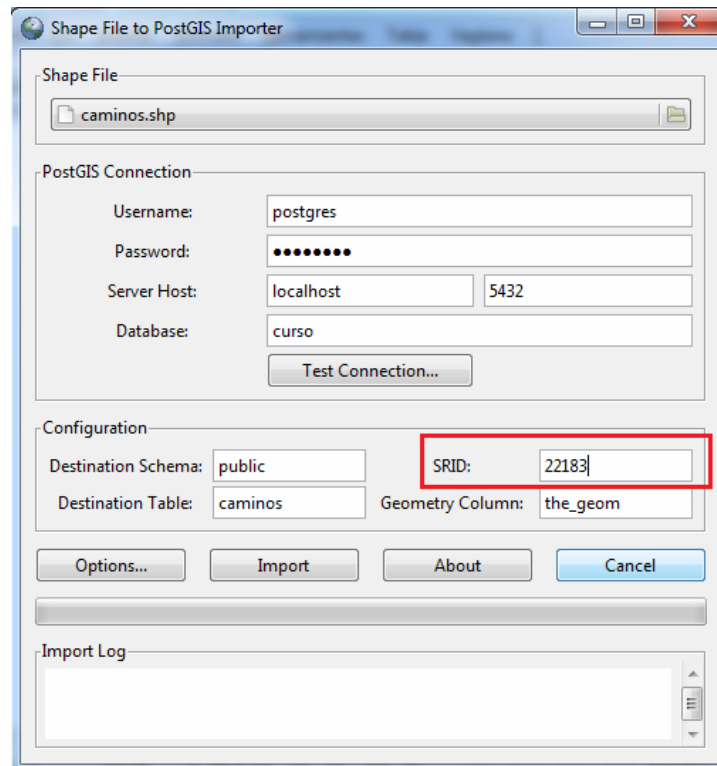


Esto nos abre una ventana que, en primer lugar, nos permite explorar los directorios y seleccionar el shapefile que deseamos migrar.



Una vez seleccionado el archivo, vemos que los datos de conexión a la base están completos y solo nos queda por definir algunos parámetros de la migración.

Colocamos el nombre de la nueva tabla, definimos el esquema en el que se creará dicha tabla y especificamos el SRID de la capa. Es muy importante corroborar que es el SRID correcto, de lo contrario, mucha de las funciones aplicadas a esos datos darán resultados erróneos.



Observemos que también tenemos la opción de migrar sólo la tabla de atributos de la capa, sin los datos geométricos. Esto puede ser un beneficio cuando los datos que nos interesan migrar se nos presentan en este formato y no deseamos utilizar ACCESS y el ODBC de Postgres.

Una vez configurados todos los datos, oprimimos el botón Importar. Esperamos la migración y finalmente nos dirigimos a PGAdmin a comprobar que la tabla ha sido creada.



ACTIVIDAD 2

Seleccionar la opción más conveniente y migrar la capa "caminos.shp" para crear la tabla "caminos" en el esquema "public".

Visualizar la capa y realizar una impresión de pantalla para agregar al documento de texto.

5 SQL: Comandos de selección. Comando WHERE

En la clase anterior vimos lo necesario para realizar una consulta de selección. Recordemos que la estructura de esta consulta es:

```
SELECT campos FROM tablas
```

En este caso, agregaremos un nuevo comando que permite realizar filtros en las consultas de selección. Dichos filtros están basados en los propios datos almacenados en la tabla.

Por ejemplo, si queremos obtener las provincias de la región Centro, colocaremos:

```
SELECT nom_prov FROM censo2010 WHERE region = 'CENTRO'
```

Los valores de tipo texto o string siempre se escriben entre comillas simples: 'CENTRO'

Obsérvese que generalmente la cláusula WHERE se compone de un campo, un operador lógico y un valor.

El tipo de operador lógico empleado depende del tipo de dato de los campos filtrados. Por ejemplo, podemos seleccionar de la tabla de departamentos los campos que contienen el código y el nombre de los departamentos, pero solo para aquellos que en el campo "poblacion" tienen un valor mayor a 500.000 habitantes

```
SELECT cod_depto, nom_depto FROM datos_depto
WHERE poblacion > 500000;
```

En este caso, el operador lógico pertenece a una operación matemática que solo se puede utilizar cuando se cuenta con un campo numérico.

Editor SQL Constructor Gráfico de Consultas

```
SELECT cod_depto, nom_depto FROM datos_depto WHERE poblacion > 500000
```

Panel de Salida

Salida de datos Comentar Mensajes Historial

| | cod_depto character varying(5) | nom_depto text |
|---|-----------------------------------|--------------------|
| 1 | 06028 | ALMIRANTE BROWN |
| 2 | 06441 | LA PLATA |
| 3 | 06427 | LA MATANZA |
| 4 | 06490 | LOMAS DE ZAMORA |
| 5 | 06357 | GENERAL PUEYRREDON |
| 6 | 06658 | QUILMES |
| 7 | 90084 | CAPITAL |
| 8 | 82084 | ROSARIO |
| 9 | 14014 | CAPITAL |

Las condiciones de filtrado se pueden concatenar mediante los nexos "OR" y "AND". Por ejemplo, si queremos obtener los departamentos con menos de 10.000 habitantes y que pertenezcan a la región NOA, la consulta será:

```
SELECT cod_depto, nom_depto FROM datos_depto
WHERE poblacion < 10000 AND region = 'NOA'
```

El nexo "AND" indica que los elementos necesitan cumplir las dos condiciones para ser seleccionados (ser del NOA y tener menos de 10.000 habitantes)

Editor SQL

Constructor Gráfico de Consultas

```
SELECT cod_depto, nom_depto FROM datos_depto
WHERE poblacion < 10000 AND region = 'NOA'
```

Panel de Salida

Salida de datos

Comentar

Mensajes

Historial

| | cod_depto character varying(5) | nom_depto text |
|----|--|--------------------------|
| 1 | 86182 | SARMIENTO |
| 2 | 86175 | SAN MARTIN |
| 3 | 86189 | SILIPICA |
| 4 | 86021 | ATAMISQUI |
| 5 | 86042 | BELGRANO |
| 6 | 86007 | AGUIRRE |
| 7 | 86154 | RIVADAVIA |
| 8 | 86084 | GUASAYAN |
| 9 | 86112 | MITRE |
| 10 | 66014 | CACHI |
| 11 | 66098 | LA VIÑA |
| 12 | 66084 | LA CANDELARIA |
| 13 | 66063 | GUACHIPAS |
| 14 | 66119 | MOQUINOS |

Si en cambio queremos seleccionar aquellos elementos que cumplan una u otra condición (cualquiera de las dos), entonces usamos el nexos "OR". Por ejemplo, si deseamos obtener las ciudades más importantes del país, pediremos aquellas que tengan más de medio millón de habitantes. Pero puede pasar que haya otras ciudades importantes, como las capitales de provincia, que sin ser tan grande, igualmente son importantes. Entonces la consulta será:

```
SELECT nom_loc FROM localidades
WHERE poblacion > 500000 OR capital = 'SI'
```

Editor SQL

Constructor Gráfico de Consultas

```
SELECT nom_loc FROM localidades
WHERE poblacion > 500000 OR capital = 'SI'
```

Panel de Salida

Salida de datos

Comentar

Mensajes


Historial

| | nom_loc text |
|---|------------------------|
| 1 | SAN MIGUEL DE TUCUMAN |
| 2 | CORDOBA |
| 3 | ROSARIO |
| 4 | LA MATANZA |
| 5 | ADROGUE |
| 6 | LOMAS DE ZAMORA |
| 7 | QUILMES |
| 8 | MAR DEL PLATA |
| 9 | LA PLATA |

6 SQL: comandos de actualización. Insertar, eliminar y modificar datos de una tabla.

6.1 Insert

La acción de agregar un nuevo registro a una tabla se puede lograr de diferentes maneras.

La más sencilla es ingresando a la visualización de la tabla con el botón *Ver los datos del objeto seleccionado*  y luego escribiendo en el último registro. Esto sólo es posible cuando la tabla tiene definida una clave primaria, ya que de esta manera PGAdmin puede identificar unívocamente a cada uno de sus registros. Las tablas migradas desde gvSIG o desde PGAdmin definen su clave primaria automáticamente durante la migración.

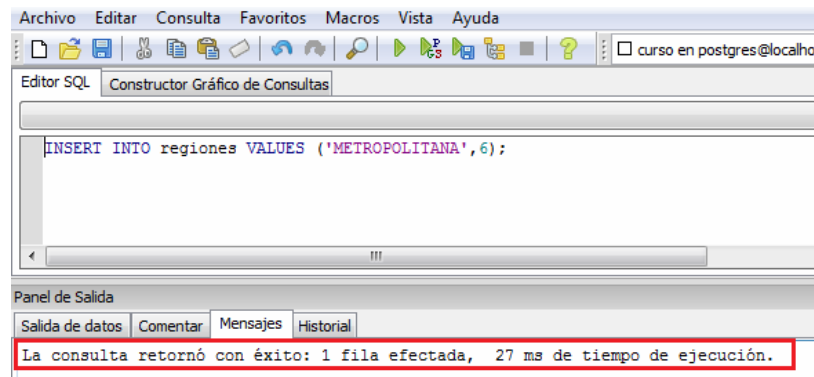
Otra forma es editando la tabla desde gvSIG o cualquier SIG. En este caso, solo se pueden editar tablas que contienen datos geométricos. La forma de editar la tabla es similar a la edición de cualquier otra capa: se pone en estado de edición la tabla y luego se agrega un nuevo elemento a la capa.


Finalmente, se puede agregar nuevo registros también utilizando el lenguaje SQL. El comando utilizado es INSERT INTO.

Por ejemplo, para agregar un nuevo registro a la tabla de regiones, la consulta es la siguiente:

```
INSERT INTO regiones VALUES ('METROPOLITANA',6);
```

Primero se escribe “INSERT INTO”, luego el nombre de la tabla, después “VALUES” y finalmente entre paréntesis se colocan los valores correspondientes a los campos en el mismo orden en que se estructura la tabla. En este caso agregamos el identificador “6” y el nombre de la región “METROPOLITANA”



Como resultado esta vez no obtenemos una tabla, sino un aviso que nos informa que la consulta se ha realizado con éxito, afecta a 1 fila. Podemos ver el resultado abriendo la tabla .

Editar Datos - curso (localhost:5432) - curso - regiones

Archivo Editar Vista Herramientas Ayuda

| | nom_region character varying(255) | cod_region smallint |
|---|--------------------------------------|------------------------|
| 1 | CENTRO | 1 |
| 2 | CUYO | 2 |
| 3 | NEA | 3 |
| 4 | NOA | 4 |
| 5 | SUR | 5 |
| 6 | METROPOLITANA | 6 |

Si solo queremos agregar valores a determinado campo de la tabla y no a todos, entonces agregamos la el nombre de dichos campos antes a continuación del nombre de la tabla:

```
INSERT INTO regiones (nom_region) VALUES ('METROPOLITANA')
```

Editar Datos - curso (localhost:5432) - curso - regiones

Archivo Editar Vista Herramientas Ayuda

| | nom_region character varying(255) | cod_region smallint |
|---|--------------------------------------|------------------------|
| 1 | CENTRO | 1 |
| 2 | CUYO | 2 |
| 3 | NEA | 3 |
| 4 | NOA | 4 |
| 5 | SUR | 5 |
| 6 | METROPOLITANA | |

En este caso el campo “cod_region” quedará vacío. Puede ocurrir que los campos tengan restricciones que no admitan valores nulos. Más adelante veremos de qué se tratan las restricciones.



ACTIVIDAD 3

Insertar un nuevo registro en la tabla “censo2010” con los siguientes datos:

Nom_prov = “PROVINCIA DEL PLATA”

Nom_region = “METROPOLITANA”

Pob2010 = 15.000.000

Cod_prov = “05”

Pegar el texto de la consulta en el documento de texto.

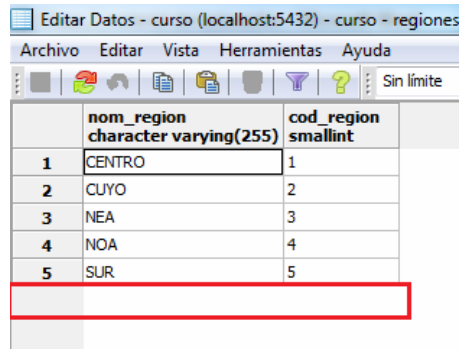
Agregar en el documento una impresión de pantalla que muestre el registro añadido en la tabla

6.2 Delete

El comando utilizado para eliminar registros es “DELETE”. Generalmente se utiliza acompañado de “WHERE”, ya que es muy raro que necesitemos borrar todos los datos de una tabla, pero manteniendo la estructura.

Por ejemplo, si queremos borrar el registro que recién creamos, escribimos:

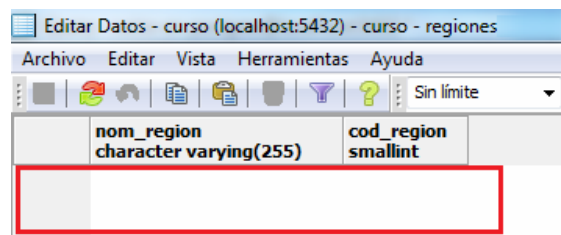
```
DELETE FROM regiones WHERE nom_region = 'METROPOLITANA'
```



| | nom_region character varying(255) | cod_region smallint |
|---|--------------------------------------|------------------------|
| 1 | CENTRO | 1 |
| 2 | CUYO | 2 |
| 3 | NEA | 3 |
| 4 | NOA | 4 |
| 5 | SUR | 5 |

En caso de que realmente necesitemos borrar todo el contenido de una tabla, pero manteniendo su estructura, entonces sí utilizamos el comando “TRUNCATE”.

```
TRUNCATE TABLE regiones; ---(Atención. Esta orden borra el contenido. No correrla.)
```



| | nom_region character varying(255) | cod_region smallint |
|--|--------------------------------------|------------------------|
|--|--------------------------------------|------------------------|

6.3 Modificar

El comando SQL utilizado para modificar los valores en las celdas de una tabla es “UPDATE”. Se diferencia de “INSERT” y de “DELETE” en que “UPDATE” no agrega ni quita registros, sino que solo altera el contenido de las celdas existentes.

Se pueden actualizar todas las celdas de una columna. Por ejemplo, podemos decirle a Postgres que todos los registros de la tabla caminos figuren como “PAVIMENTADO” en el campo “clase”

```
UPDATE caminos SET clase = 'PAVIMENTADO'
```

Como vemos, la estructura de esta consulta es “UPDATE”, el nombre de la tabla, luego el comando “SET”, el nombre del campo y el valor que irá en ese campo.


Editor SQL Constructor Gráfico de Consultas

```
UPDATE caminos SET clase = 'PAVIMENTADO'
```

Panel de Salida

Salida de datos Comentar Mensajes Historial

La consulta retornó con éxito: 8789 filas afectadas, 3810 ms de tiempo de ejecución.

La consulta no devuelve una tabla, ya que no es de selección. Pero sí nos devuelve el mensaje que dice que han sido afectadas 8789 filas. Para ver el resultado podemos abrir la tabla con el botón  o bien, realizar la consulta de selección correspondiente:

```
SELECT * FROM caminos
```

Editor SQL Constructor Gráfico de Consultas

Borrar Eliminar Todos

```
SELECT * FROM caminos
```

Panel de Salida

Salida de datos Comentar Mensajes Historial

| | gid integer | union double precision | tipo text | nombre text | jurisdicci text | clase text | transitabi text | provincia text | pais text | observacio text | hoja text | fuentes text | fec_act text | modif_shp text | codigo text | the_geom geometry |
|----|----------------|---------------------------|--------------|----------------|--------------------|---------------|--------------------|-------------------|--------------|--------------------|--------------|-----------------|-----------------|-------------------|----------------|----------------------|
| 1 | 1 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 2 | 2 | 0 | RUTA | 4 | PROVINCIA | PAVIMENTADO | PERMANENT | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 3 | 3 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 4 | 4 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 5 | 5 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 6 | 6 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 7 | 7 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 8 | 8 | 0 | RUTA | 10 | PROVINCIA | PAVIMENTADO | PERMANENT | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 9 | 9 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 10 | 10 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 11 | 11 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 12 | 12 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 13 | 13 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 14 | 14 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |
| 15 | 15 | 0 | CAMI | SIN NOME | VECINAL | PAVIMENTADO | TEMPORARJ | LA PAMPA | ARGE | | 3563-IGM | DIC05 | | | 42 | 0105000020 |

OK. Unix Lín 1 Col 22 Car 22 8789 filas, 2010 ms

También puede ocurrir que sólo deseemos actualizar una parte de los registros en función de alguna condición que cumplen. Por ejemplo, podemos actualizar de nuevo el campo “clase”, pero esta vez le pedimos a Postgres que le coloque el valor “DE TIERRA” a los caminos de jurisdicción vecinal. Para agregar la condición utilizaremos el comando “WHERE” visto al principio de la clase.

```
UPDATE caminos SET clase = 'DE TIERRA'
WHERE jurisdicci = 'VECINAL'
```

En este caso, podemos ver que los únicos registros que cambiaron sus valores en el campo “clase” son aquellos que cumplen con la condición solicitada.

Editor SQL | Constructor Gráfico de Consultas

```
UPDATE caminos SET clase = 'DE TIERRA'
WHERE jurisdicci = 'VECINAL'
```

Panel de Salida

Salida de datos | Comentar | Mensajes | Historial

La consulta retornó con éxito: 6766 filas afectadas, 1012 ms de tiempo de ejecución.

Y el resultado...

Editor SQL | Constructor Gráfico de Consultas

select * from caminos

Panel de Salida

Salida de datos | Comentar | Mensajes | Historial

| | gid | union | double | precision | tipo | nombre | jurisdicci | clase | transitabi | provincia | pais | observacio | hoja | fuelle | fec_act | modif_shp | codigo |
|------|---------|-------|--------|-----------|---------------|------------|-------------|--------------------|------------|-----------|-------|------------|------|--------|---------|-----------|--------|
| | integer | | | | text | text | text | text | text | text | text | text | text | text | text | text | text |
| 6740 | 8777 | 0 | | | HUELL SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3963-IGM | DIC05 | | 42 | | | | |
| 6741 | 8778 | 0 | | | HUELL SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3963-IGM | DIC05 | | 42 | | | | |
| 6742 | 8779 | 0 | | | HUELL SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3963-IGM | DIC05 | | 42 | | | | |
| 6743 | 8780 | 0 | | | HUELL SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3963-IGM | DIC05 | | 42 | | | | |
| 6744 | 8781 | 0 | | | HUELL SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3963-IGM | DIC05 | | 42 | | | | |
| 6745 | 8782 | 0 | | | HUELL SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3963-IGM | DIC05 | | 42 | | | | |
| 6746 | 8783 | 0 | | | HUELL SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3963-IGM | DIC05 | | 42 | | | | |
| 6747 | 1 | 0 | | | CAME SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |
| 6748 | 3 | 0 | | | CAME SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |
| 6749 | 4 | 0 | | | CAME SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |
| 6750 | 5 | 0 | | | CAME SIN NOM | VECINAL | DE TIERRA | TEMPORARI LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |
| 6751 | 2 | 0 | | | RUUTA 4 | PROVINCIAL | PAVIMENTADO | PERMANENT LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |
| 6752 | 8 | 0 | | | RUUTA 10 | PROVINCIAL | PAVIMENTADO | PERMANENT LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |
| 6753 | 19 | 0 | | | RUUTA 35 | NACIONAL | PAVIMENTADO | PERMANENT LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |
| 6754 | 61 | 0 | | | RUUTA 35 | NACIONAL | PAVIMENTADO | PERMANENT LA PAMPA | ARGE | 3563-IGM | DIC05 | | 42 | | | | |

OK. Unix Lín 1 Col 22 Car 22 8789 filas. 1919 ms

También podemos actualizar los datos de un campo aprovechando los valores existentes en la misma tabla. En este caso, luego de SET colocamos la expresión que genera el nuevo valor.

Por ejemplo, podemos agregar la columna “densidad” a la tabla “censo2010” y calcular dentro de ella el resultado de la división entre los campos “poblacion” y “superficie”

Para agregar el nuevo campo podemos correr la siguiente consulta:

```
ALTER TABLE censo2010 ADD COLUMN densidad double precision;
```

O bien pararnos en el objeto Columnas dentro de la tabla “censo2010”, y luego cliqueamos con el botón derecho y seleccionamos *Nueva Columna*.

Objetos (7)

Tablas (12)

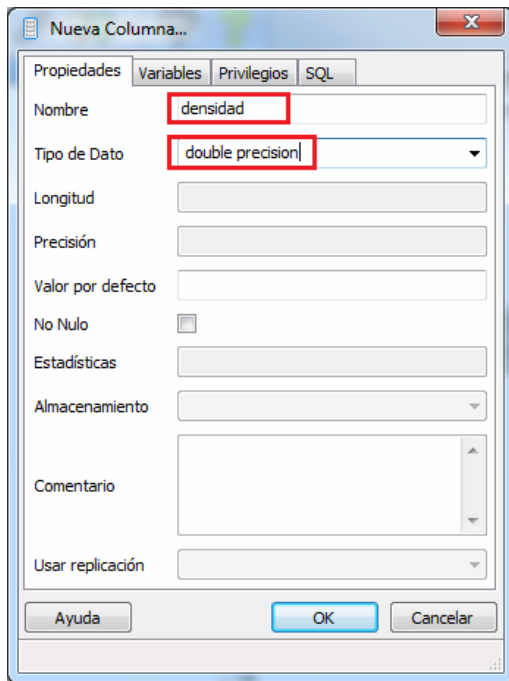
- caminos
- censo2010
- Columnas (8)
 - nom_pro
 - nom_reg
 - superficie
 - pob2001
 - pob2010
 - hogares
 - computadoras
 - ...

Refrescar

Nueva Columna...

Reporte de lista de objetos

Esto nos abrirá una ventana en donde colocaremos el nombre y el tipo de dato, que en este caso es “double precision”, que admite decimales.



Una vez agregada la columna, podemos correr la consulta que coloca el valor de densidad calculado.

```
UPDATE censo2010 SET densidad = pob2010 / superficie;
```

| | nom_prov character vari | nom_region character vari | superficie double precis | pob2001 bigint | pob2010 bigint | hogares bigint | computadora bigint | cod_prov [PK] charact | densidad double precis |
|----|----------------------------|------------------------------|-----------------------------|-------------------|-------------------|-------------------|-----------------------|--------------------------|---------------------------|
| 1 | CIUDAD AUTON | CENTRO | 200 | 2776138 | 2890151 | 1150134 | 789145 | 02 | 14450.755 |
| 2 | BUENOS AIRES | CENTRO | 307571 | 13827203 | 15625084 | 4789484 | 2308740 | 06 | 50.8015515116 |
| 3 | CATAMARCA | NOA | 102602 | 334568 | 367828 | 96001 | 34518 | 10 | 3.5849983431 |
| 4 | CORDOBA | CENTRO | 165321 | 3066801 | 3308876 | 1031843 | 510197 | 14 | 20.0148559468 |
| 5 | CORRIENTES | NEA | 88199 | 930991 | 992595 | 267797 | 86184 | 18 | 11.2540391614 |
| 6 | CHACO | NEA | 99633 | 984446 | 1055259 | 288422 | 85393 | 22 | 10.5914606606 |
| 7 | CHUBUT | SUR | 224686 | 413237 | 509108 | 157166 | 89422 | 26 | 2.26586436182 |
| 8 | ENTRE RIOS | CENTRO | 78781 | 1158147 | 1235994 | 375121 | 164184 | 30 | 15.6889859230 |
| 9 | FORMOSA | NEA | 72066 | 486559 | 530162 | 140303 | 36426 | 34 | 7.35661754502 |
| 10 | JUJUY | NOA | 53219 | 611888 | 673307 | 174630 | 59114 | 38 | 12.6516281779 |
| 11 | LA PAMPA | SUR | 143440 | 299294 | 318951 | 107674 | 51344 | 42 | 2.22358477412 |
| 12 | LA RIOJA | CUYO | 89680 | 289983 | 333642 | 91097 | 38013 | 46 | 3.72036128456 |
| 13 | MENDOZA | CUYO | 148827 | 1579651 | 1738929 | 494841 | 214868 | 50 | 11.6842306839 |
| 14 | MISIONES | NEA | 29801 | 965522 | 1101593 | 302953 | 86162 | 54 | 36.9649676183 |
| 15 | NEUQUEN | SUR | 94078 | 474155 | 551266 | 170057 | 90178 | 58 | 5.85966963583 |
| 16 | RÍO NEGRO | SUR | 203013 | 552822 | 638645 | 199189 | 97439 | 62 | 3.14583302547 |
| 17 | SALTA | NOA | 155488 | 1030051 | 1214441 | 200304 | 03607 | 66 | 7.81051270823 |

Aquí podríamos agregar también a continuación la cláusula WHERE, pero en este caso queremos que el cambio se realice en todas las filas.

**ACTIVIDAD 4**

Agregar un campo “diferencia” a la tabla “censo2010”. Calcular la diferencia de población entre 2010 y 2001.

Hacer una impresión de pantalla que muestre el resultado del ejercicio y agregarlo en el documento de texto.

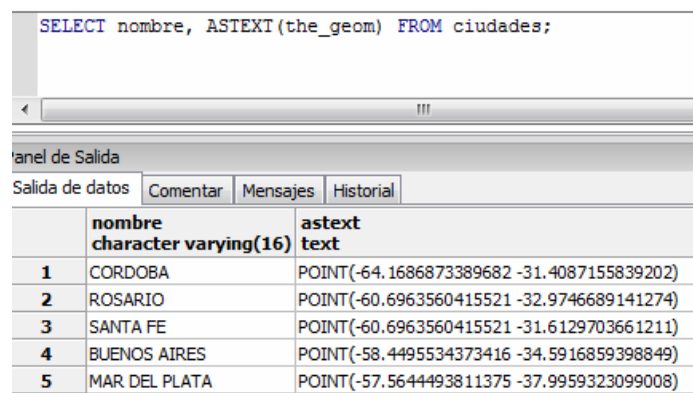
7 SQL espacial: consulta de datos espaciales: distancia, superficie, longitud, SRID, tipo de geometría y cantidad de vértices.

Ahora veremos una serie de comandos que nos devuelven información extraída o procesada a partir de los datos geométricos.

7.1 Geometría como texto

En la primera clase vimos una consulta muy sencilla que nos devuelve el dato geométrico traducido a un lenguaje fácil de interpretar:

```
SELECT nombre, ASTEXT(the_geom) FROM ciudades;
```



| | nombre character varying(16) | astext text |
|---|---------------------------------|--|
| 1 | CORDOBA | POINT(-64.1686873389682 -31.4087155839202) |
| 2 | ROSARIO | POINT(-60.6963560415521 -32.9746689141274) |
| 3 | SANTA FE | POINT(-60.6963560415521 -31.6129703661211) |
| 4 | BUENOS AIRES | POINT(-58.4495534373416 -34.5916859398849) |
| 5 | MAR DEL PLATA | POINT(-57.5644493811375 -37.9959323099008) |

Esto nos devuelve el dato geométrico de manera literal, y se puede comprobar lo expuesto al principio de la clase referido a los tipos de geometrías.

7.2 SRID

También dentro del campo geométrico se almacena el SRID, el número identificador del sistema de referencia en el que está proyectada la información.

```
SELECT nombre, SRID(the_geom) from ciudades;
```

```
SELECT nombre, SRID(the_geom) from ciudades;
```

Panel de Salida

Salida de datos Comentar Mensajes Historial

| | nombre character varying(16) | srid integer |
|---|---|-------------------------------|
| 1 | CORDOBA | 4326 |
| 2 | ROSARIO | 4326 |
| 3 | SANTA FE | 4326 |
| 4 | BUENOS AIRES | 4326 |
| 5 | MAR DEL PLATA | 4326 |

Esta consulta nos sirve para corroborar un sistema de referencias antes de realizar una reproyección, o algún cálculo que necesite un determinado sistema de referencias.

7.3 Tipo de geometría

Para averiguar el tipo de geometría que se ha utilizado para almacenar un dato, escribimos

```
SELECT GEOMETRYTYPE (the_geom) FROM ciudades;
```

```
SELECT GEOMETRYTYPE (the_geom) FROM ciudades;
```

Panel de Salida

Salida de datos Comentar Mensajes Historial

| | geometrytype text |
|---|------------------------------------|
| 1 | POINT |
| 2 | POINT |
| 3 | POINT |
| 4 | POINT |
| 5 | POINT |

Es importante remarcar que siempre es necesario colocar el campo geométrico en las consultas de este tipo. No solo porque la estructura de la función lo requiere, sino también porque podría existir otro campo geométrico dentro de la misma tabla.

Cantidad de vértices

En la primera parte de la clase vimos que para poder utilizar algunos tipos de geometría, era importante la cantidad de vértices de la figura. Para averiguar este dato escribimos:

```
SELECT NPOINTS (the_geom) FROM caminos;
```

```
SELECT NPOINTS (the_geom) FROM caminos;
```

| | npoints integer |
|------|--------------------|
| 3119 | 12 |
| 3120 | 2 |
| 3121 | 7 |
| 3122 | 4 |
| 3123 | 4 |
| 3124 | 2 |
| 3125 | 3 |
| 3126 | 5 |
| 3127 | 6 |
| 3128 | 3 |
| 3129 | 8 |
| 3130 | 7 |

Esta función como las anteriores pueden utilizarse tanto como parte del pedido de campos en la consulta de selección, como también en las condiciones de “WHERE”. Por ejemplo si queremos conocer los tramos de ruta que tienen solo 2 vértices, y por lo tanto podrían ser convertidos de “LINESTRING” a “LINE” para aplicarle cálculos específicos, podemos escribir:

```
SELECT * FROM caminos
WHERE NPOINTS (the_geom) = 2;
```

7.4 Longitud, area y perímetro.

En este caso veremos como calcular algunos datos básicos de nuestras geometrías. Cabe aclarar, previamente, que este tipo de cálculo solo es posible cuando se cuentan con coordenadas planas, es decir, cuando la unidad en la que están expresadas las coordenadas son metros (para nuestro país). Esto es así debido a que estos cálculos son propios de la geometría euclidiana, en donde el espacio es plano; y en el caso de las coordenadas angulares (en grados) el principio es diferente.

Hay cálculos que se pueden realizar solo en ciertos tipos de geometrías, y en otro no, como la superficie, que no puede ser aplicada ni en líneas ni en puntos.

Empecemos por la más simple. La longitud de una línea.

```
SELECT nombre, LENGTH (the_geom) FROM caminos;
```

| Salida de datos | | | |
|-----------------|----------------|----------------------------|-----------|
| | Comentar | Mensajes | Historial |
| | nombre text | length double precision | |
| 1 | SIN NOME | 1.10402396784503 | |
| 2 | SIN NOME | 0.70451929856017 | |
| 3 | SIN NOME | 0.768258322033389 | |
| 4 | SIN NOME | 0.776402796331662 | |
| 5 | SIN NOME | 0.784220175712043 | |
| 6 | SIN NOME | 0.800187096332865 | |
| 7 | SIN NOME | 9202.5381973084 | |
| 8 | SIN NOME | 503.382769047457 | |
| 9 | SIN NOME | 0.408893341020468 | |
| 10 | SIN NOME | 9278.45070534614 | |
| 11 | SIN NOME | 781.527506351805 | |
| 12 | SIN NOME | 574.893760197835 | |
| 13 | SIN NOME | 600.143732777903 | |
| 14 | SIN NOME | 1325.50546132387 | |
| 15 | SIN NOME | 6109.06685287565 | |

Nos devuelve el nombre de la ruta y la longitud de de cada tramo. Las unidades en que están expresadas dichas longitudes las mismas que las coordenadas, es decir: metros. Si deseamos ver estas longitudes, por ejemplo, en kilómetros, solo debemos aplicar la operación matemática correspondiente a la consulta.

```
SELECT nombre, LENGTH (the_geom)/1000 FROM caminos;
```

| Salida de datos | | | |
|-----------------|----------------|------------------------------|-----------|
| | Comentar | Mensajes | Historial |
| | nombre text | ?column? double precision | |
| 3118 | SIN NOMBRE | 5.01333966397383 | |
| 3119 | SIN NOMBRE | 8.13171049204507 | |
| 3120 | SIN NOMBRE | 0.115168295120848 | |
| 3121 | SIN NOMBRE | 5.0348317020348 | |
| 3122 | SIN NOMBRE | 7.37849161051354 | |
| 3123 | SIN NOMBRE | 7.59997944375357 | |
| 3124 | SIN NOMBRE | 5.01053364265565 | |
| 3125 | SIN NOMBRE | 2.46873948821776 | |
| 3126 | SIN NOMBRE | 5.00976670887293 | |

Y podemos agregar un alias al campo de resultado que aparece sin nombre

```
SELECT nombre, LENGTH (the_geom)/1000 AS long_km FROM caminos;
```

| Salida de datos | | | |
|-----------------|----------------|-----------------------------|-----------|
| | Comentar | Mensajes | Historial |
| | nombre text | long_km double precision | |
| 7687 | 14 | 2.76635914025765 | |
| 7688 | 11 | 0.154118187361585 | |
| 7689 | 143 | 2.30248606745332 | |
| 7690 | 14 | 13.7602890102535 | |
| 7691 | 14 | 25.6740032320493 | |
| 7692 | 14 | 15.7386636946097 | |
| 7693 | 14 | 0.889804379734685 | |
| 7694 | 25 | 10.7047009345685 | |
| 7695 | 143 | 2.7043256767337 | |

El cálculo del area o superficie es similar, pero se aplica solamente a los polígonos:

```
SELECT nom_depto, AREA (the_geom) FROM departamentos;
```


| Salida de datos | | | Comentar | Mensajes | Historial |
|-----------------|---------------------------------|--|----------|----------|-----------|
| | nom_depto text | area double precision | | | |
| 1 | REALICO | 2543746803.76172 | | | |
| 2 | TRENEL | 1977869406.39648 | | | |
| 3 | CHALILEO | 8851365279.14258 | | | |
| 4 | LOVENTUE | 9081798801.16797 | | | |
| 5 | TOAY | 5102215923.37305 | | | |
| 6 | CAPITAL | 2520744305.57617 | | | |
| 7 | LIMAY MAHUIDA | 10150513600.666 | | | |
| 8 | UTRACAN | 12802666196.9414 | | | |

En este caso el resultado está expresado en metros cuadrados. Para pasarlo a kilómetros, que es la unidad más adecuada para el cálculo de superficie a esta escala, solo debemos dividir el área por 1000000.

```
SELECT nom_depto, AREA (the_geom) / 1000000 as area_km2
FROM departamentos;
```

| Salida de datos | | | Comentar | Mensajes | Historial |
|-----------------|---------------------------------|--|----------|----------|-----------|
| | nom_depto text | area_km2 double precision | | | |
| 1 | REALICO | 2543.74680376172 | | | |
| 2 | TRENEL | 1977.86940639648 | | | |
| 3 | CHALILEO | 8851.36527914258 | | | |
| 4 | LOVENTUE | 9081.79880116797 | | | |
| 5 | TOAY | 5102.21592337305 | | | |
| 6 | CAPITAL | 2520.74430557617 | | | |
| 7 | LIMAY MAHUIDA | 10150.513600666 | | | |
| 8 | UTRACAN | 12802.6661969414 | | | |
| 9 | CHAPALEUFU | 2567.19821254883 | | | |
| 10 | MARACO | 2509.39755757031 | | | |

Ahora sí el resultado está en kilómetros cuadrados, y además el campo tiene un nombre apropiado.

El cálculo del perímetro es similar al de la longitud, pero también se aplica sólo a los polígonos.

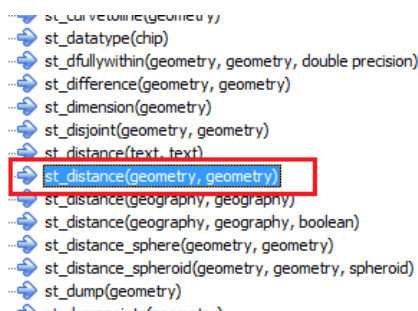
```
SELECT nom_depto, PERIMETER (the_geom) / 1000 as perimetro
FROM departamentos;
```

| Salida de datos | | | |
|-----------------|-------------------|-------------------------------|-----------|
| | Comentar | Mensajes | Historial |
| | nom_depto text | perimetro double precision | |
| 1 | REALICO | 201.753846334993 | |
| 2 | TRENEL | 179.063778663949 | |
| 3 | CHALILEO | 377.056509690004 | |
| 4 | LOVENTUE | 387.672463274246 | |
| 5 | TOAY | 316.182481413879 | |
| 6 | CAPITAL | 215.821869531267 | |
| 7 | LIMAY MAHUIDA | 403.124884632724 | |
| 8 | UTRACAN | 484.894890498836 | |
| 9 | CHAPALEUFU | 202.681928497832 | |
| 10 | MARACO | 200.45144671765 | |
| 11 | QUEMU QUEMU | 199.471123725505 | |
| 12 | CATRILO | 199.095405821453 | |

En este caso, convertido a kilómetros y con el nombre agregado.

7.5 Distancias

Para finalizar veremos como se realiza el cálculo de distancias. Se realiza a partir de la función ST_DISTANCE, la cual podemos conocer a partir de navegar por el árbol de objetos de PGAdmin, dentro del esquema "public".



Allí vemos que la función tiene la estructura:

ST_DISTANCE (geometry, geometry)

Esto significa que se necesitan dos geometrías para poder calcular una distancia. Por ejemplo, si tenemos una tabla de vuelos que realiza un avión sanitario, podemos tener en una columna "geom_inicial" donde se almacena el punto donde inicia cada viaje, y luego otra columna "geom_final" que contiene el punto del lugar donde termina el vuelo. En este caso, para calcular la distancia de cada vuelo haríamos la siguiente consulta:

```
SELECT ST_DISTANCE (geom_inicial, geom_final) from vuelos.
```

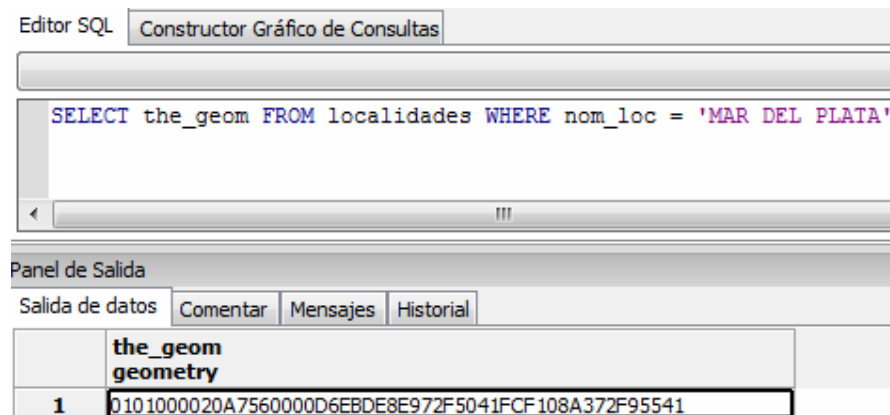
Pero en la realidad, generalmente se calcula la distancia existente entre cada elemento de la tabla y otro punto conocido. En este caso el valor variable es el del campo de la tabla, mientras que el valor permanente es ese punto fijo que deseamos establecer.

Por ejemplo, si queremos conocer la distancia de cada localidad del país hacia la ciudad de Mar del Plata, podemos empezar la consulta de esta manera:

```
SELECT nom_loc, ST_DISTANCE (the_geom, PUNTOFIJO)
FROM localidades;
```

Luego continuamos reemplazando “PUNTOFIJO” por la consulta que nos trae el punto de la ciudad de Mar del Plata:

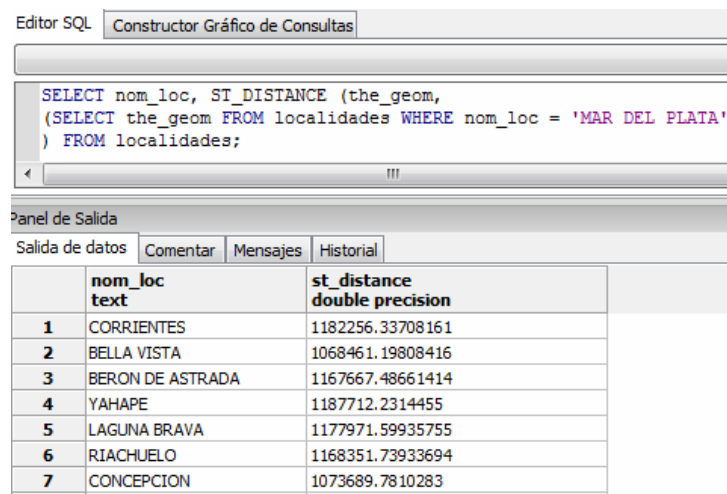
```
SELECT the_geom FROM localidades WHERE nom_loc = 'MAR DEL PLATA'
```



Esto nos devuelve un solo registro del campo “the_geom”. Este será el punto fijo que utilizaremos en la consulta. La forma de incorporarlo es reemplazar “PUNTOFIJO” por esta consulta encerrada entre paréntesis.

```
SELECT nom_loc, ST_DISTANCE (the_geom,
(SELECT the_geom FROM localidades WHERE nom_loc = 'MAR DEL
PLATA')
) FROM localidades;
```

Todo lo resaltado en amarillo es lo que reemplaza a “PUNTOFIJO”. Nótese que la consulta respeta la estructura de la función ST_DISTANCE, ya que proporciona dos geometrías: la primera es un campo geométrico y la segunda es una geometría producida por otra consulta.



Por supuesto, la distancia obtenida está en metros, por lo cual es necesario pasarla a kilómetros, para hacer una lectura más adecuada.

```
SELECT nom_loc, ST_DISTANCE (the_geom,
(SELECT the_geom FROM localidades WHERE nom_loc = 'MAR DEL
PLATA')
) /1000 as distancia FROM localidades;
```

| Editor SQL | | Constructor Gráfico de Consultas |
|--|-----------------------|----------------------------------|
| SELECT nom_loc, ST_DISTANCE (the_geom, (SELECT the_geom FROM localidades WHERE nom_loc = | | |
| SELECT nom_loc, ST_DISTANCE (the_geom, | | |
| (SELECT the_geom FROM localidades WHERE nom_loc = 'MAR DEL PLATA') | | |
|) /1000 as distancia FROM localidades; | | |
| Panel de Salida | | |
| Salida de datos | | |
| Comentar Mensajes Historial | | |
| | nom_loc text | distancia double precision |
| 1 | CORRIENTES | 1182.25633708161 |
| 2 | BELLA VISTA | 1068.46119808416 |
| 3 | BERON DE ASTRADA | 1167.66748661414 |
| 4 | YAHAPÉ | 1187.7122314455 |
| 5 | LAGUNA BRAVA | 1177.97159935755 |
| 6 | RIACHUELO | 1168.35173933694 |
| 7 | CONCEPCION | 1073.6897810283 |
| 8 | SANTA ROSA | 1088.28498747054 |
| 9 | TABAY | 1084.66045443552 |
| 10 | CAZADORES CORRENTINOS | 898.407931617647 |



ACTIVIDAD 5

Calcular la distancia desde todas las localidades a la ciudad de Salta.
Pegar en el documento de texto una impresión de pantalla que de cuenta del resultado.

Bibliografía

OLAYA, Víctor. (2011). *Sistemas de Información Geográfica*. Versión 1.0, Rev. 24 de marzo de 2011. Proyecto “Libro Libre SIG”.

http://forge.osor.eu/docman/view.php/13/577/Libro_SIG.zip

THE POSTGIS TEAM. Manual de PostGIS 1.5.3.

<http://www.postgis.org/download/postgis-1.5.3.pdf>

OBE, Regina y HSU Leo. (2011). *PostGIS in Action*. Editorial Manning. Stamford.

Usted es libre de compartir - copiar, distribuir, ejecutar y comunicar públicamente y de hacer obras derivadas de este documento

Este documento es una obra compartida bajo la licencia Creative Commons.

Atribución-NoComercial-CompartirIgual 2.5 Argentina (CC BY-NC-SA 2.5)



Atribución — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo o que apoyan el uso que hace de su obra).



No Comercial — No puede utilizar esta obra para fines comerciales.



Compartir bajo la Misma Licencia — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

Aviso: Al reutilizar o distribuir la obra, tiene que dejar muy en claro los términos de la licencia de esta obra. La mejor forma de hacerlo es enlazar a la siguiente página:

<http://creativecommons.org/licenses/by-nc-sa/2.5/ar/>

Programa Nacional Mapa Educativo
Ministerio de Educación
República Argentina

Teléfono/Fax: 54 11 4129-1408
Correo electrónico: mapaedu_nac@me.gov.ar
www.mapaeducativo.edu.ar



Índice

| | |
|--|-----------|
| 1 EXTENSIÓN POSTGIS | 1 |
| 2 TABLAS DE POSTGIS: SISTEMAS DE REFERENCIA Y COLUMNAS GEOMÉTRICAS | 2 |
| 3 TIPOS DE GEOMETRÍA | 5 |
| 4 MIGRACIÓN DE DATOS ESPACIALES: DESDE GVSIG Y PGADMIN. | 8 |
| 4.1 GVSIG | 8 |
| 4.2 PLUGIN DE PGADMIN | 10 |
| 4.2.1 <i>Instalar Shapefile and DBF loader en PGAdmin.....</i> | <i>10</i> |
| 4.2.2 <i>Utilizar Shapefile and DBF loader.....</i> | <i>12</i> |
| 5 SQL: COMANDOS DE SELECCIÓN. COMANDO WHERE | 13 |
| 6 SQL: COMANDOS DE ACTUALIZACIÓN. INSERTAR, ELIMINAR Y MODIFICAR DATOS DE UNA TABLA..... | 16 |
| 6.1 INSERT | 16 |
| 6.2 DELETE | 18 |
| 6.3 MODIFICAR | 18 |
| 7 SQL ESPACIAL: CONSULTA DE DATOS ESPACIALES: DISTANCIA, SUPERFICIE, LONGITUD, SRID, TIPO DE GEOMETRÍA Y CANTIDAD DE VÉRTICES. | 22 |
| 7.1 GEOMETRÍA COMO TEXTO | 22 |
| 7.2 SRID | 22 |
| 7.3 TIPO DE GEOMETRÍA | 23 |
| 7.4 LONGITUD, AREA Y PERÍMETRO..... | 24 |
| 7.5 DISTANCIAS..... | 27 |
| BIBLIOGRAFÍA | 30 |