CLASE 1

Bases de datos espaciales

República Argentina
Ministerio de Educación
Programa Nacional Mapa Educativo

Curso de capacitación
Bases de datos espaciales

CLASE 1
Material de lectura. Versión 1

Temas de esta clase:

Introducción a las bases de datos.
Bases de datos relacionales.
Motores de bases de datos: PostgreSQL
Elementos de una base de datos: Esquemas, tablas, campos, registros.
Migración de datos alfanuméricos desde Access.
SQL: comandos de selección.
Visualización de información geográfica de la base desde gvSIG.

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, PGAdmin y las otras herramientas 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 Bases de datos

1.1 Base de datos. Definición.

Una **base de datos** es un conjunto de datos almacenados con una estructura lógica; es decir, tan importante como los datos, es la estructura conceptual con la que se relacionan los datos entre sí mismos. En la práctica, se puede pensar esto como el conjunto de datos más los programas (software) que hacen de ellos un conjunto existente.

Al igual que con el concepto de SIG, es interesante conocer otras definiciones que existen para “Base de datos”:

- "Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es servir a una o más aplicaciones de la mejor forma posible; los datos se almacenan de modo que resulten independientes de

los programas que los usan. Se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados" (Martin, 1975).

- "Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales, de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender todas las necesidades de los diferentes usuarios" (Deen, 1985).

- "Colección de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de los datos" (A. de Miguel, 1993).

- "Una base de datos consiste en alguna colección de datos persistentes e independientes usados por una organización determinada" (Date, 1995).

Las definiciones de base de datos son numerosas. Todas coinciden en que son un conjunto de datos almacenados en un soporte de acceso directo. Los datos están interrelacionados y estructurados de acuerdo a un modelo que sea capaz de recoger el máximo contenido semántico.

1.2 Características de las bases de datos

El concepto de Base de Datos determina algunas características que le son propias, por ejemplo:

- El mundo real considera interrelaciones entre datos y restricciones semánticas que deben estar presentes en una base de datos. Una base de datos no solo debe almacenar entidades y atributos (recordar los sistemas tradicionales de archivos), sino que también debe almacenar interrelaciones entre datos. Por otro lado, actualmente se le está dando mucha importancia a las restricciones semánticas, de manera que éstas se almacenan junto con los datos.
- La redundancia de datos debe ser controlada, de forma que no existan duplicidades perjudiciales e innecesarias. Las redundancias físicas, convenientes muchas veces a fin de responder a objetivos de eficiencia, son tratadas por el mismo sistema, de modo que no puedan producirse incoherencias. Esto significa que en las bases de datos no está permitida la redundancia lógica, pero si se admite cierta redundancia física por motivos de eficiencia.
- Las bases de datos pretenden servir a toda la organización; es decir, a múltiples usuarios y a diferentes aplicaciones.
- La definición y descripción del conjunto de datos contenido en la base debe ser única e integrada con los mismos datos. En las bases de datos, la descripción, y en algunos casos también, una definición y documentación completas se almacenan junto con los datos, de modo que éstos estén documentados, y cualquier cambio que se produzca en la documentación debe quedar recogido en el sistema.
- La actualización y recuperación de las bases de datos debe realizarse mediante procesos bien determinados, procedimientos que han de estar

diseñados de modo que se mantenga la integridad, seguridad y confidencialidad de la base.

El objetivo de disminuir la redundancia de un conjunto de datos determina dos características fundamentales que poseerá cualquier sistema de Bases de Datos:

a) Integrada: una base de datos puede considerarse como una unificación de varios archivos de datos independientes, donde se elimina parcial o totalmente cualquier **redundancia** entre los mismos.

b) Compartida: Se entiende que partes individuales de la Base de Datos pueden compartirse entre varios usuarios distintos, en el sentido que cada uno de ellos puede tener acceso a la misma parte de la Base de Datos y utilizarla con propósitos diferentes. Tal comportamiento es, en verdad, consecuencia del hecho de que la Base de Datos es integrada.

Consecuencia del hecho de que la Base de Datos sea integrada, se advierte que cualquier usuario tendrá acceso para modificar sólo algún subconjunto de la base completa; además, los subconjuntos de diferentes usuarios se procesarán de muy diversas maneras. En otras palabras, diferentes usuarios percibirán de modos muy distintos una base de datos.

1.3 Ventajas de la utilización de bases de datos

Veremos a continuación las principales ventajas de las bases de datos con respecto a gestión no organizada de los datos. Entiéndase *gestión no organizada de los datos* a la forma de trabajo que generalmente se utiliza, en donde los datos se almacenan en archivos sueltos, generados para ser utilizado en cada proyecto y que no pueden ser modificados por más de un usuario a la vez :

- **Independencia de los datos respecto a los tratamientos y viceversa:** esto supone que un cambio en los tratamientos no imponga un nuevo diseño lógico y/o físico de la base de datos. Por otro lado, cambios en la incorporación, desaparición de datos, cambios en la estructura física o caminos de acceso no deben obligar a alterar los programas. Así se evita la reprogramación de las aplicaciones. Es el punto de partida para la adaptación de los sistemas de información a la evolución de las organizaciones.
- **Coherencia de los resultados:** debido a que la información de la Base de Datos se recoge y se almacena una sola vez, en todos los tratamientos se utilizan los mismos datos, por lo que los resultados de estos son coherentes y comparables. Así, se eliminan las divergencias en los resultados.
- **Mejor disponibilidad de los datos para el conjunto de los usuarios:** en una Base de Datos ningún usuario es propietario de los datos, pues éstos se comparten entre las aplicaciones, existiendo una mayor disponibilidad y transparencia.
- **Mayor valor informativo:** esto se refiere al concepto de *sinergia*, en donde el valor informativo del conjunto de datos es superior a la suma del valor informativo de los elementos individuales.
- **Mejor y más normalizada documentación:** la mayoría de los SGBD proporcionan herramientas para reflejar el contenido semántico de los datos; es decir, incluyen una descripción de los datos dentro del sistema.
- **Mayor eficiencia en la captura, validación e ingreso de datos al sistema:** al no existir redundancias, los datos se capturan y validan una sola vez aumentando el rendimiento del proceso previo al almacenamiento.

- **Reducción del espacio de almacenamiento:** por un lado, la disminución de redundancias y las técnicas de compactación hacen que disminuya el espacio en disco. Sin embargo, los diccionarios, referencias, punteros, listas invertidas también ocupan espacio.

1.4 Modelos de bases de datos

Una base de datos puede ser clasificada en función del modelo que se utiliza para construirla. Los modelos implican una forma particular de almacenar los datos, la forma en que se estructuran los datos y las relaciones que se establecen entre estos.

Cada modelo tiene sus ventajas e inconvenientes, y para cada uso es necesario decidir el modelo más conveniente.

De los distintos modelos existentes, los más habituales son:

- Bases de datos jerárquicas
- Bases de datos en red
- Bases de datos relacionales
- Bases de datos orientadas a objetos

De todos estos modelos, las bases de datos relacionales son en este momento el modelo más utilizado en todo tipo de ámbitos, y especialmente en los SIG.

2 Bases de datos relacionales

Este es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente, y es el modelo de base de datos que utilizaremos en el curso. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de *relaciones*. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados *tuplas*. Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían a las tuplas, y los campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

ACTIVIDAD 1. Crear un nuevo documento de texto en Word o LibreOffice para responder a las actividades y luego subirlo al Campus.

Describir el aspecto que considera más importante como ventaja del uso de bases de datos frente al esquema de trabajo con archivos shapefiles. Identifique una situación real en su ámbito de trabajo con respecto a esta ventaja.

3 PostgreSQL

PostgreSQL es un motor de bases de datos, es servidor de bases de datos relacional libre, liberado bajo la licencia BSD. Es el motor de base de datos que utilizaremos en este curso.

Principales características:

Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión) *PostgreSQL* permite que, mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *'commit'*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases; eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos: *PostgreSQL* provee nativamente soporte para:

Números de precisión arbitraria
Texto de largo ilimitado
Figuras geométricas (con una variedad de funciones asociadas)
Direcciones IP (IPv4 e IPv6)
Bloques de direcciones estilo CIDR
Direcciones MAC
Arrays
Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de *PostgreSQL.* Algunos ejemplos son los tipos de datos SIG creados por el proyecto *PostGIS*.

Otras características: Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas *(foreign keys)*
Disparadores *(triggers)*
Vistas
Integridad transaccional
Herencia de tablas
Tipos de datos y operaciones geométricas

Funciones: Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como *bifurcaciones* y *bucles*, hasta las complejidades de la programación de orientación a objetos o la programación funcional.
Los disparadores (**triggers** en inglés) son funciones enlazadas a operaciones sobre los datos.

3.1 Breve historia de PostgreSQL

PostgreSQL es el último resultado de una larga evolución comenzada con el proyecto *Ingres* en la Universidad de Berkeley. El líder del proyecto, Michael Stonebraker abandonó Berkeley para comercializar *Ingres* en 1982, pero finalmente regresó a la academia. Tras su retorno a Berkeley en 1985, Stonebraker comenzó un proyecto *post-Ingres* para resolver los problemas con el modelo de base de datos relacional que había sido aclarado a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos"; es decir, combinaciones de datos simples que conforman una única unidad. Actualmente, estos son llamados objetos.

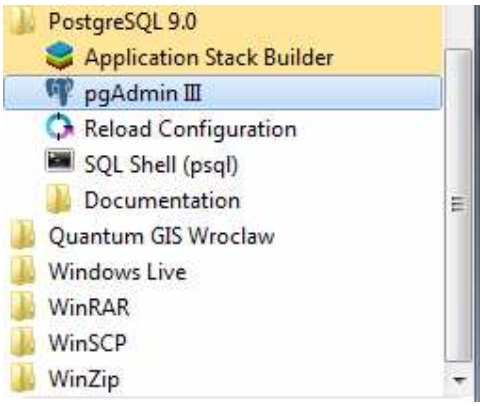
El proyecto resultante, llamado *Postgres*, era orientado a introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones - las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En *Postgres* la base de datos involucraba las relaciones y podía obtener información de tablas relacionadas utilizando reglas.

Habiendo comenzado en 1986, el equipo liberó una serie de documentos que describían la base del sistema y en 1988 ya poseían un prototipo funcional. La versión 1 fue liberada a un pequeño grupo de usuarios en junio de 1989, seguida por la versión 2 con un sistema de reglas reescrito en junio de 1990. Para la versión 3, liberada en 1991, el sistema de reglas fue reescrito nuevamente, pero también agregó soporte para múltiples administradores de almacenamiento y un sistema de consultas mejorado. Hacia 1993, *Postgres* había crecido inmensamente en popularidad y poseía una demanda asfixiante de nuevas funcionalidades. Tras liberar la versión 4, la cual era una simple versión de limpieza, el proyecto fue abandonado.

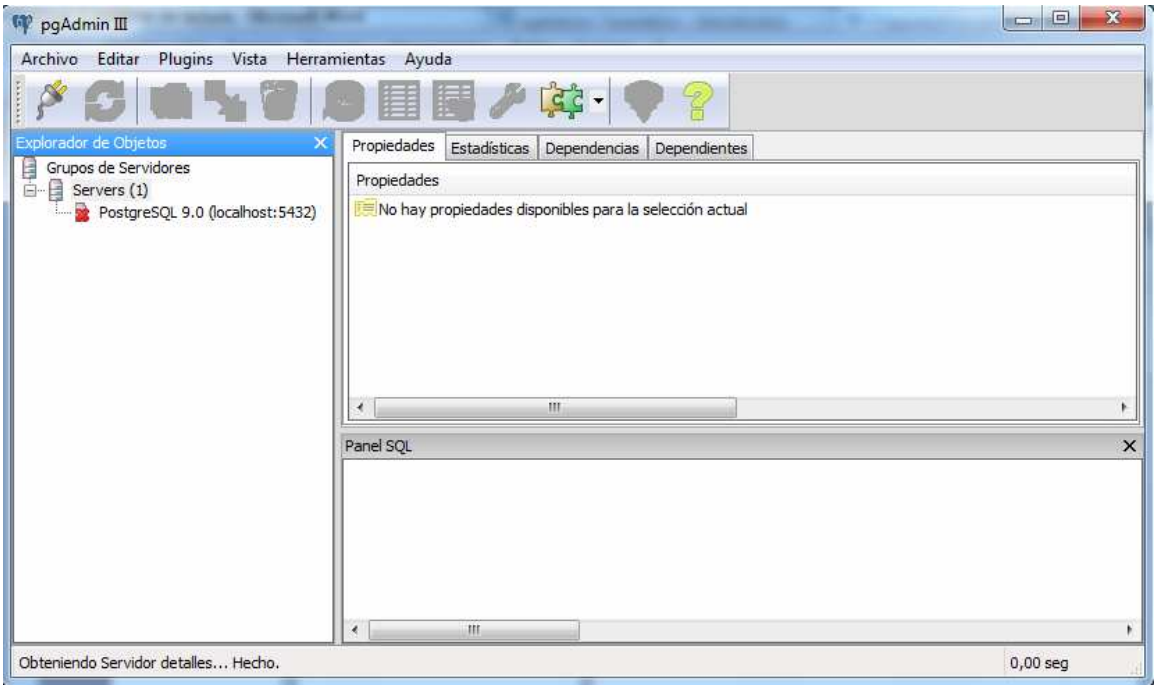
A pesar de que el proyecto *Postgres* hubiese finalizado oficialmente, la licencia BSD bajo la cual *Postgres* había sido liberado permitió a desarrolladores de código abierto obtener una copia del código para continuar con su desarrollo.

3.2 PGAdmin como Sistema gestor de base de datos (SGBD)

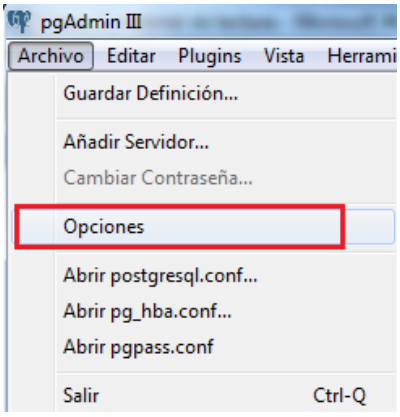
Podemos abrir PGAdmin, el SGBD (sistema gestor de bases de datos) que trae consigo la instalación de Postgres desde *Inicio, Programas, Postgres, PGAdmin*.



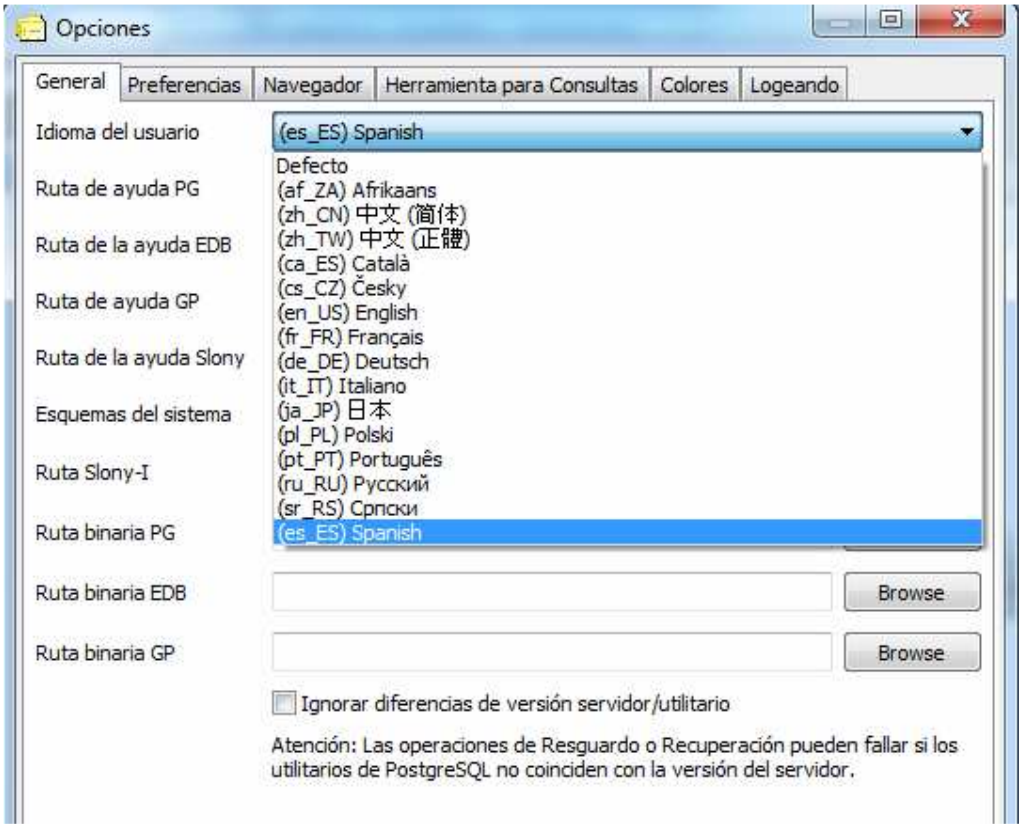
Al abrirse, PGAdmin muestra una ventana con tres secciones y una barra de herramientas en la parte superior.



Antes de continuar, vamos a configurar el idioma de esta aplicación, en caso de que los menús se muestren en inglés. Vamos a *File, Options*...

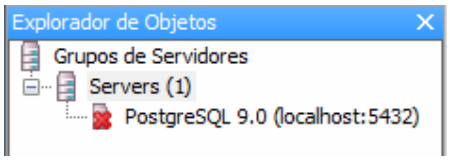


Y allí en la solapa *General*, cambiamos la opción en *User Language*. Seleccionamos la opción *(es_ES) Spanish*.



Luego cerramos la ventana, cerramos PGAdmin, y lo volvemos a iniciar.

Continuemos. Si el motor de Postgres fue instalado en la misma máquina desde donde estamos ejecutando PGAdmin, por defecto aparecerá la conexión en el panel de la izquierda.



Si estamos ejecutando PGAdmin desde una máquina diferente al servidor, debemos hacer la conexión al motor para poder trabajar con Postgres.

Apretamos el botón *Agregar conexión a un servidor*  . Aparecerá una ventana en donde ingresaremos los parámetros de conexión al motor de Postgres. Prestemos atención a estos datos, ya que serán necesarios en todo momento que debamos conectarnos a Postgres, tanto desde PGAdmin, como desde cualquier otro cliente de bases de datos (gvSIG).

Se nos abre la ventana Nuevo registro de servidor en donde se visualiza una gran cantidad de opciones a definir. Vamos a concentrarnos en cuatro datos que nunca deben faltar para lograr una conexión al servidor:

Nombre del servidor: es el nombre o número IP de la máquina donde hemos instalado el motor de Postgres. Cuando se trabaja en la misma máquina en la que está instalado Postgres, el servidor se denomina “localhost”

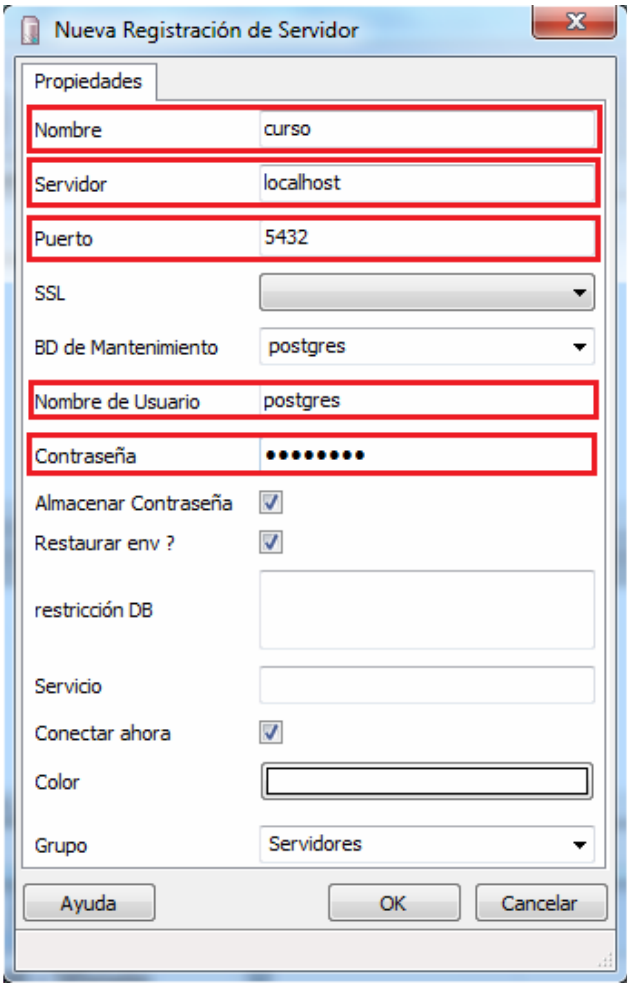
Puerto: Por defecto se le asigna el puerto 5432 a Postgres. En caso de que no se conecte, es necesario preguntar al área de soporte informático.

Nombre de usuario: es el nombre del usuario que nos asignaron para acceder a Postgres. En este primer momento usaremos el usuario administrador llamado “postgres”

Contraseña: es la clave con la que accede el usuario mencionado arriba. En este primer momento la clave es “postgres”. Si no funciona, hay que consultar con quienes hicieron la instalación de Postgres.

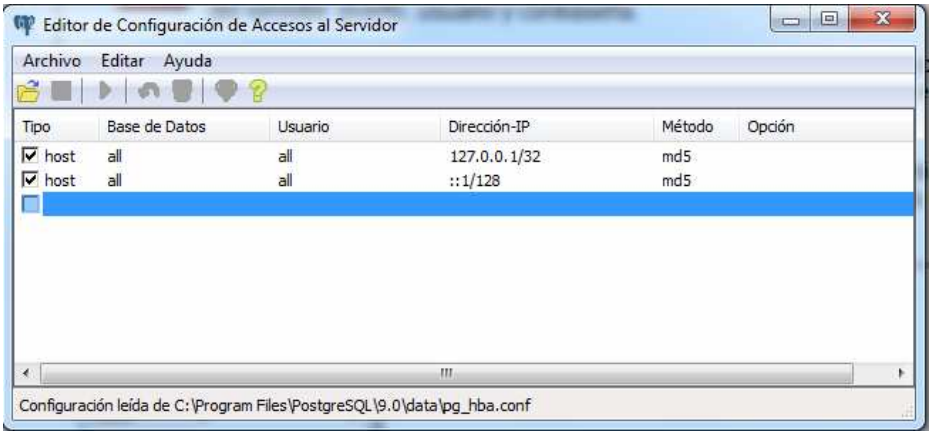
Además de estos datos, PGAdmin nos pide que le asignemos un nombre a la conexión, para diferenciarla de otras posibles conexiones. En este caso, sugerimos colocarle el nombre “curso”.

¡IMPORTANTE!: Si la conexión no se produce, en primer lugar hay que chequear que los parámetros colocados sean los correctos: nombre o IP del servidor, puerto, usuario y contraseña.
También es necesario corroborar que estén correctamente otorgados los permisos en el servidor.

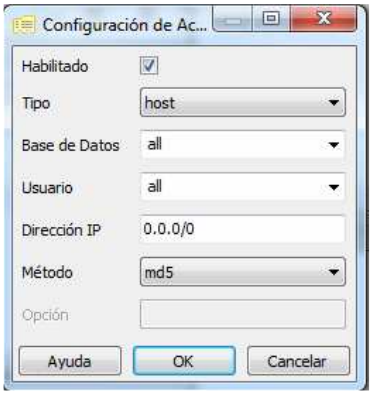


Cuando accedemos a Postgres desde máquinas diferentes al servidor, previamente debemos configurar los permisos. Para esto hay que ejecutar la aplicación PGAdmin en el servidor, y en el menú Archivo, seleccionar la opción Abrir pg_hba.conf.

Esto abre una ventana que muestra los permisos de acceso a Postgres. Para habilitar nuevos permisos, se hace doble clic sobre el último registro vacío.

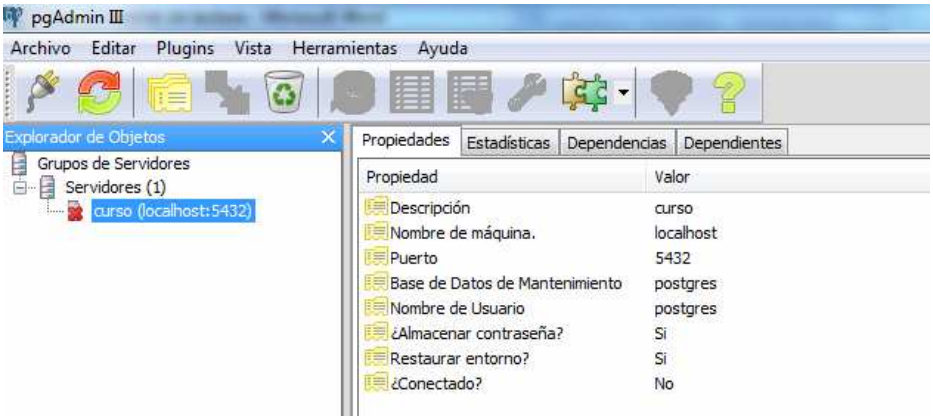


Se nos abrirá una ventana más pequeña en donde colocaremos los siguientes parámetros:

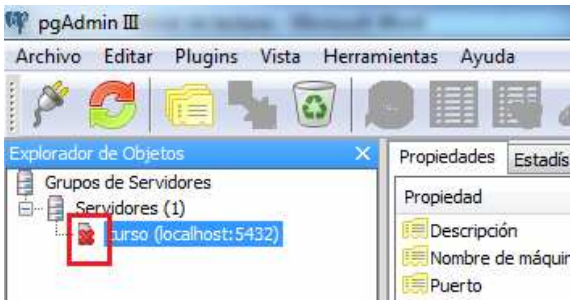


Más adelante veremos a qué se refiere cada uno de ellos. En caso de que sea necesario controlar el acceso a Postgres, se puede poner exactamente el número de IP de la máquina desde la cual se trabajará. Lo mismo se puede hacer con el nombre de usuario.

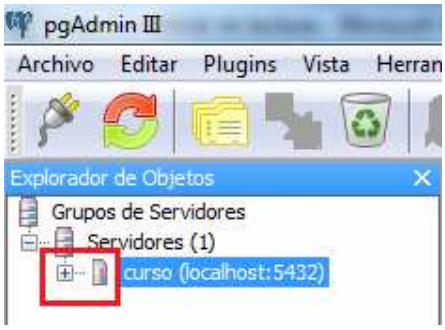
A partir de la creación de la nueva conexión, vemos que aparece un nuevo elemento en el panel de la izquierda. Este panel muestra el explorador de objetos, y veremos que se irá poblando de elemento a medida que avancemos con el curso. El panel superior de la derecha muestra las propiedades del objeto que está seleccionado en el explorador de objetos de la izquierda. Por ejemplo, podemos hacer clic sobre la nueva conexión que hemos creado para ver sus propiedades desplegadas a la derecha.



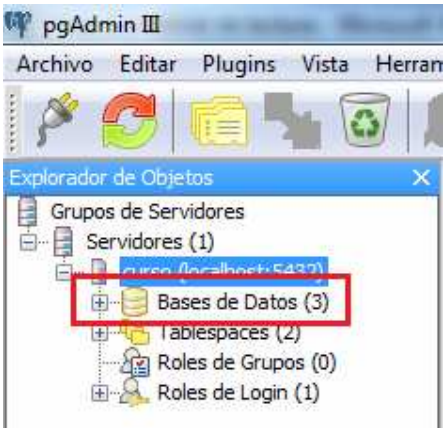
Una vez creada la conexión, debemos activarla para empezar a trabajar con Postgres. Hacemos doble clic en el ícono al lado del nombre de la conexión.



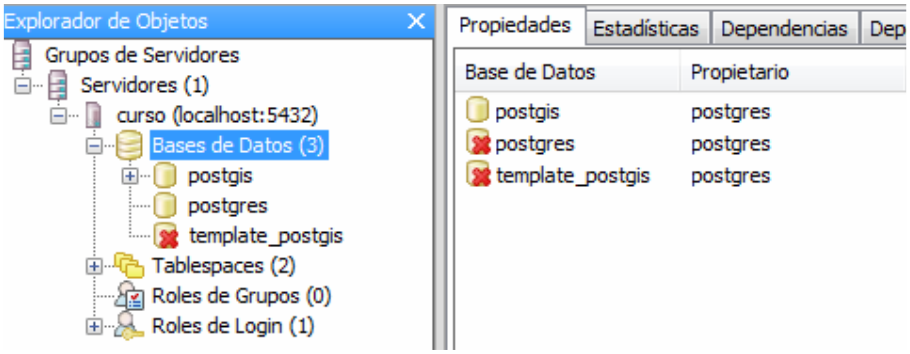
Si la conexión se produce, vemos que el ícono cambia de aspecto, y además aparece un signo “+” que indica que hay más objetos dependientes de la conexión que pueden ser desplegados.



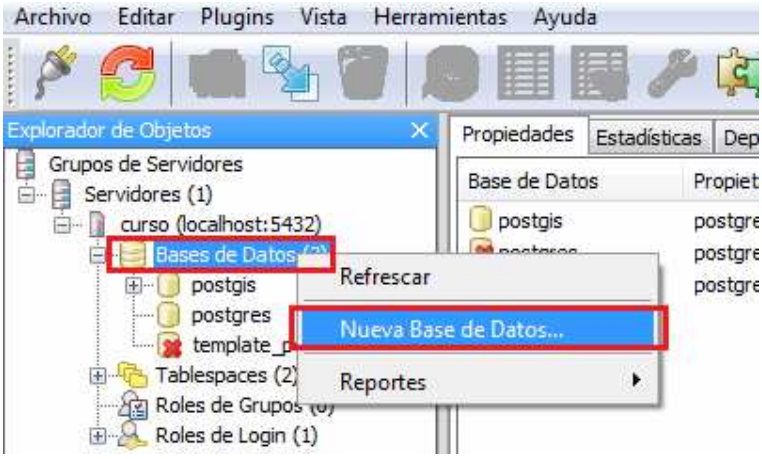
Si hacemos un clic sobre el símbolo “+” vemos como aparecen los nuevos objetos. De entre ellos, nos enfocaremos por ahora en las Bases de Datos.



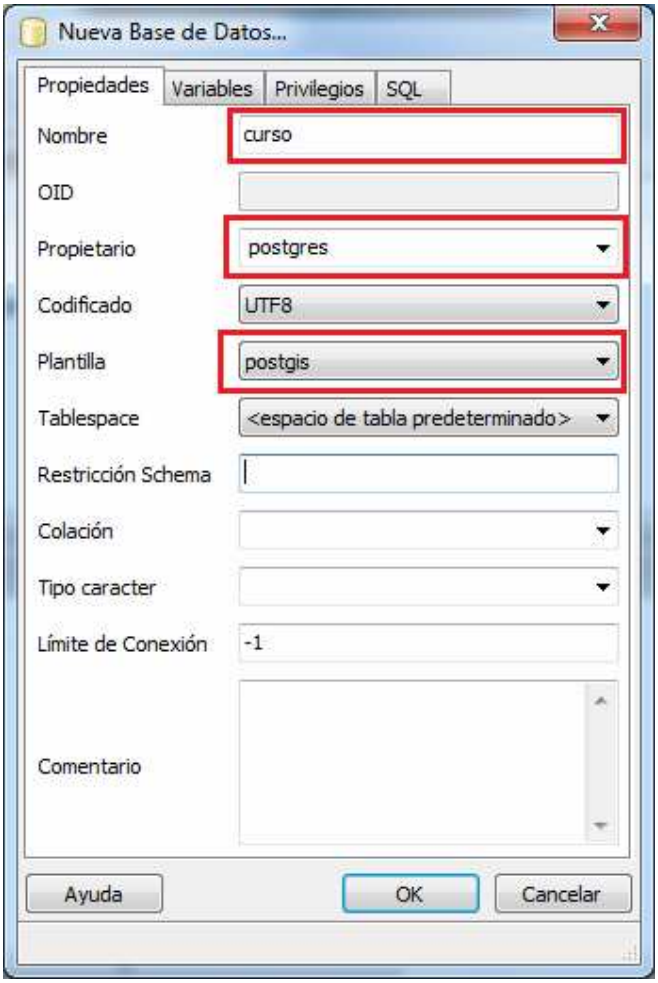
En este momento podemos tener dos o tres bases de datos (dependiendo del tipo de instalación de PostGIS que hayamos hecho).



Vamos a crear una nueva base de datos que se llamará “curso”. Hacemos clic con el botón derecho sobre Bases de Datos



Se nos abrirá una ventana para determinar las propiedades de la nueva base.



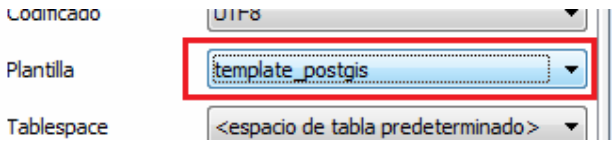
Además del nombre de la nueva base, otro dato importante es el propietario. El propietario es el usuario de Postgres que por defecto tiene permisos especiales sobre la base por el hecho de pertenecerle. En este caso, el propietario es “postgres”.

La plantilla es un elemento muy importante, ya que determina el tipo de base de datos que estamos creando.

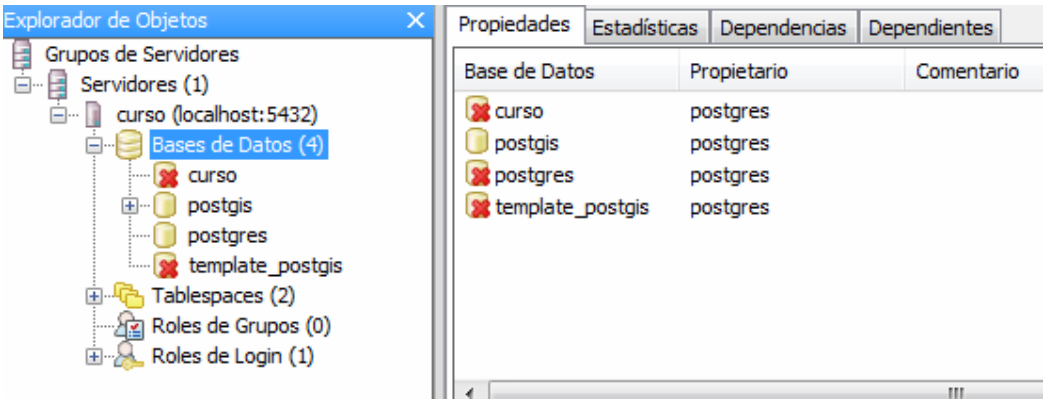
La particularidad que tiene Postgres, es que al añadirle el módulo PostGIS, no solo soporta los datos que se encuentran frecuentemente en cualquier base de datos, como texto, números, fechas, etc., sino que también soporta datos geométricos. Dichos datos geométricos, al contener coordenadas reales del terreno se convierten en datos geográficos, como los que manejan los SIG. Además, PostGIS también agrega funcionalidades específicas de datos espaciales, como cálculo de superficies, distancias, transformaciones geométricas, reproyecciones, etc.

Durante la instalación del módulo PostGIS, se nos preguntó si queríamos crear una base de datos espaciales llamada “postgis”, o bien si deseábamos añadir la plantilla “template_postgis” (según el tipo de instalación). De una u otra manera, hemos creado una plantilla que puede usarse de *molde* para crear nuevas bases de datos espaciales.

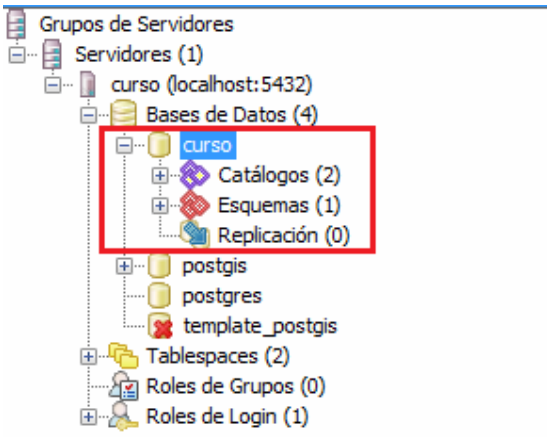
Entonces, en la opción Plantilla, seleccionamos “postgis” o “template_postgis”, según dispongamos de una o de otra.



Luego de presionar *OK*, vemos que en el explorador de objetos de la izquierda, aparece un nuevo elemento: la base curso. En caso de que no aparezca, debemos refrescar el grupo de objetos *Bases de Datos* y luego activar el botón *Refrescar* .



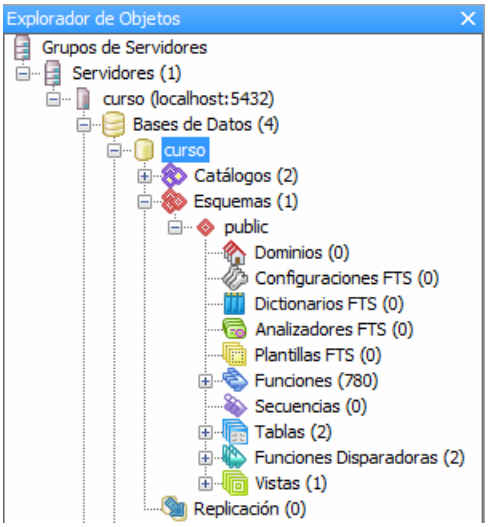
Vamos a explorar los elementos de la base de datos nueva. Para esto hacemos clic sobre el signo “+” que aparece a la izquierda de la base “curso”.



Vemos que se despliegan varios objetos. De ellos, vamos a continuar indagando en los Esquemas.

Los esquemas son compartimentos de la base de datos que nos permiten organizar las tablas según el criterio que deseamos emplear. Por ejemplo, se pueden poner tablas con datos espaciales en un esquema, tablas con solo datos alfanuméricos en otro esquema, o se pueden organizar esquemas en función del origen de los datos (INDEC, INTA, Instituto Geográfico Nacional, relevamientos propios, etc.)

Por defecto, una base siempre debe tener, al menos, un esquema. Es por ello que la base nueva que acabamos de crear ya tiene incorporado el esquema “public”. Para ver el esquema “public” hacemos clic sobre el botón de + de *Esquemas*.

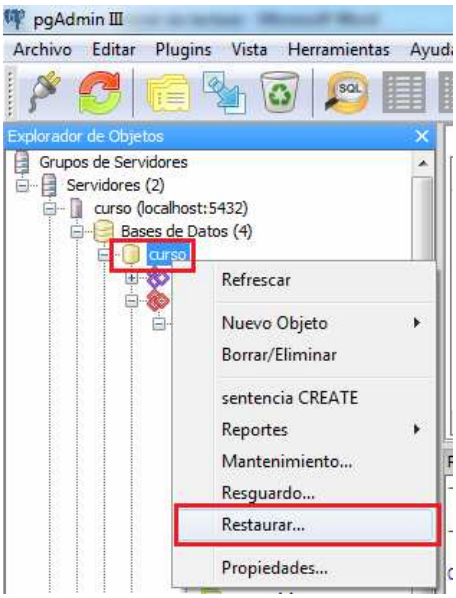


De la larga lista de elementos que aparecen allí, nuevamente nos concentraremos en uno: las tablas. Aquí es donde empezamos a tomar contacto con la información almacenada.

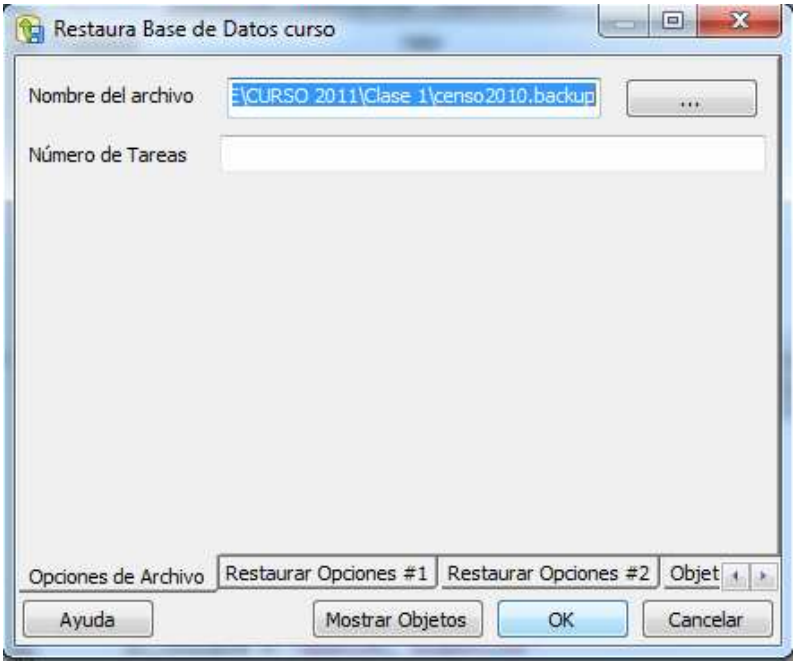
Las tablas que aquí se presentan (clic en "+" de Tablas) son las que utiliza PostGIS para manejar los datos espaciales. En la clase siguiente veremos de qué se tratan los datos contenidos en estas tablas.

Ahora agregaremos una nueva tabla a la base de datos. Lo haremos a través de la restauración de un archivo de resguardo. Este tipo de archivos se crea desde PGAdmin y sirve para guardar una copia segura de los datos, o para transferir los datos a otro usuario. Utilizaremos el archivo llamado censo2010.backup que acompaña el material de práctica de esta clase.

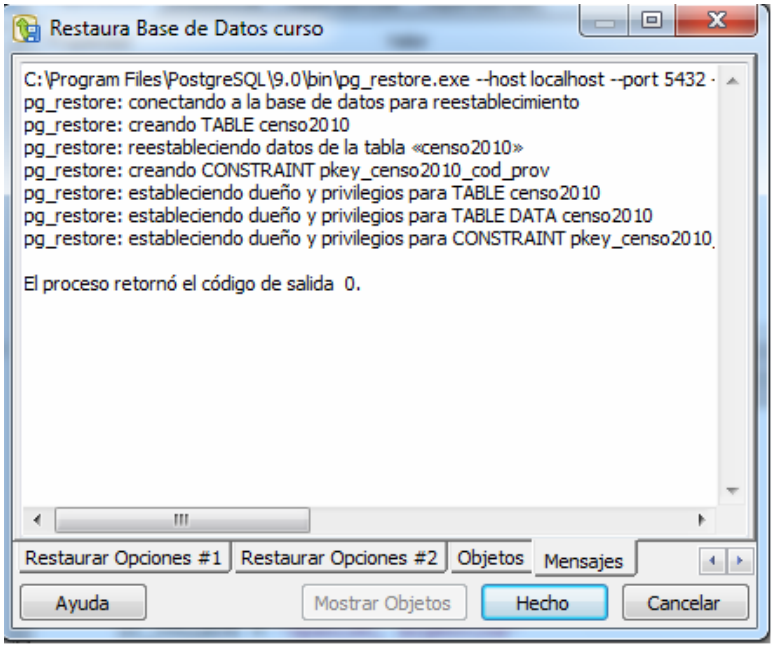
Hacemos clic con el botón derecho sobre el ícono de la base "curso", y de la lista desplegable seleccionamos la opción *Restaurar…*



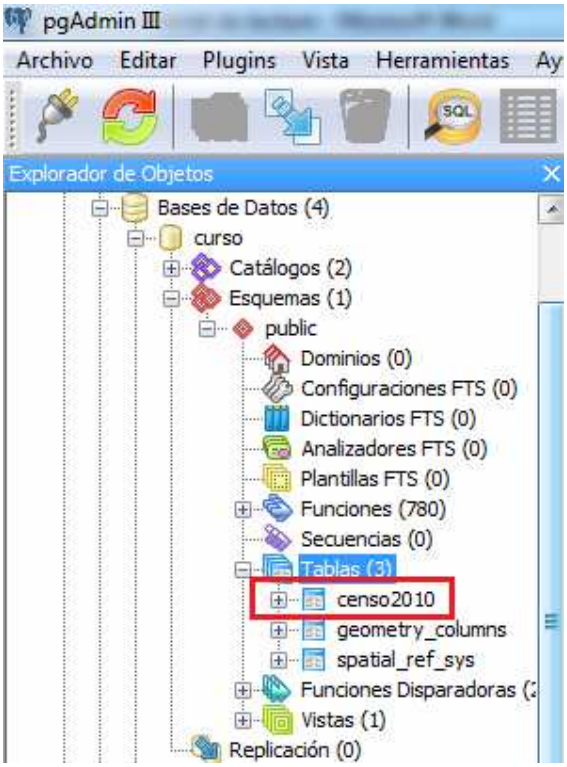
Cuando se abre la ventana, en Nombre del Archivo exploramos hasta encontrar el archivo de resguardo censo2010.backup. Luego presionamos *OK*.



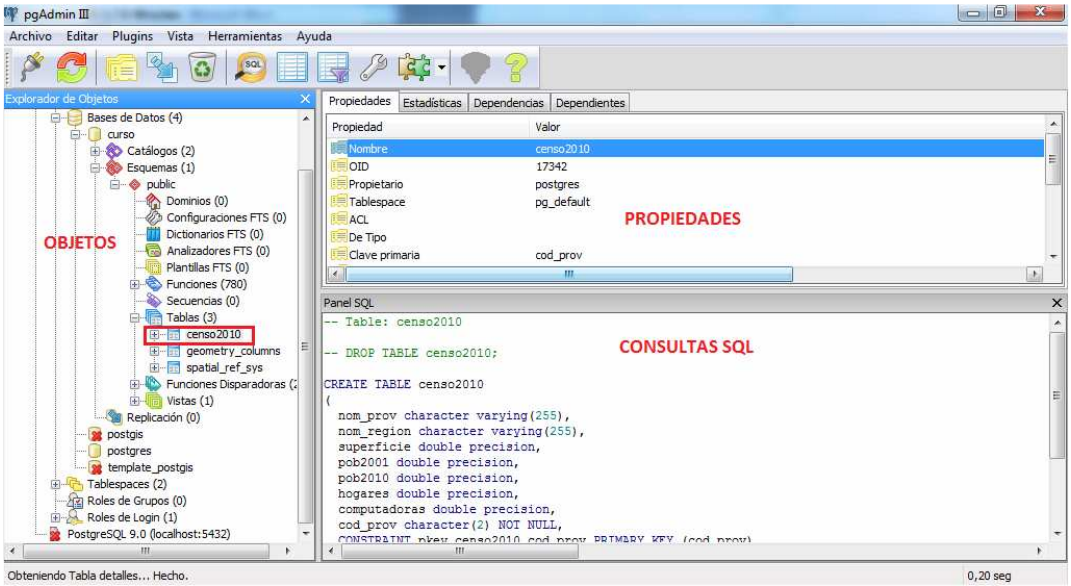
Empieza la restauración de los diferentes elementos implicados y en la ventana aparecen los mensajes correspondientes. Cuando leemos que “El proceso retornó el código de salida 0”, significa que ha terminado exitosamente el proceso.



Ahora podemos ver que se ha agregado un nuevo elemento dentro de *Tablas*. En caso de no aparecer la nueva tabla recordemos que podemos seleccionar el ícono de Tablas y luego apretar el botón de *Refrescar*  .



Podemos conocer la estructura de la tabla, es decir los nombres de las columnas y los tipos de datos que almacena cada una, haciendo un clic sobre la tabla y visualizando los datos que aparecen en los dos paneles de la derecha.



El panel superior, como ya vimos, muestra las propiedades, y el panel inferior nos presenta algo nuevo: la sintaxis SQL que crea el objeto señalado en el panel de objetos.

Si analizamos un poco esta consulta, vemos que hay una orden de crear una tabla, y luego se enumeran los nombres de los campos o columnas con el tipo de dato que cada una de ellas almacenará.

```
CREATE TABLE censo2010
(
  nom_prov character varying(255),
  nom_region character varying(255),
  superficie double precision,
  pob2001 double precision,
  pob2010 double precision,
  hogares double precision,
  computadoras double precision,
  cod_prov character(2) NOT NULL,
  CONSTRAINT pkey_censo2010_cod_prov PRIMARY KEY (cod_prov)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE censo2010 OWNER TO postgres;
```

Por ahora no nos detendremos demasiado en analizar esta consulta, pero sí es importante saber que cada objeto de la base de datos se crea y modifica utilizando el lenguaje de consulta SQL. A pesar de ello, veremos que también se pueden realizar las mismas acciones oprimiendo botones, arrastrando objetos, haciendo clics, etc..

Para visualizar los datos de la tabla, primero la seleccionamos en el árbol de objetos de la izquierda, y luego apretamos el botón *Ver los datos del objeto seleccionado*  .

Editar Datos - curso (localhost:5432) - curso - censo2010

Archivo Editar Vista Herramientas Ayuda

Sin límite

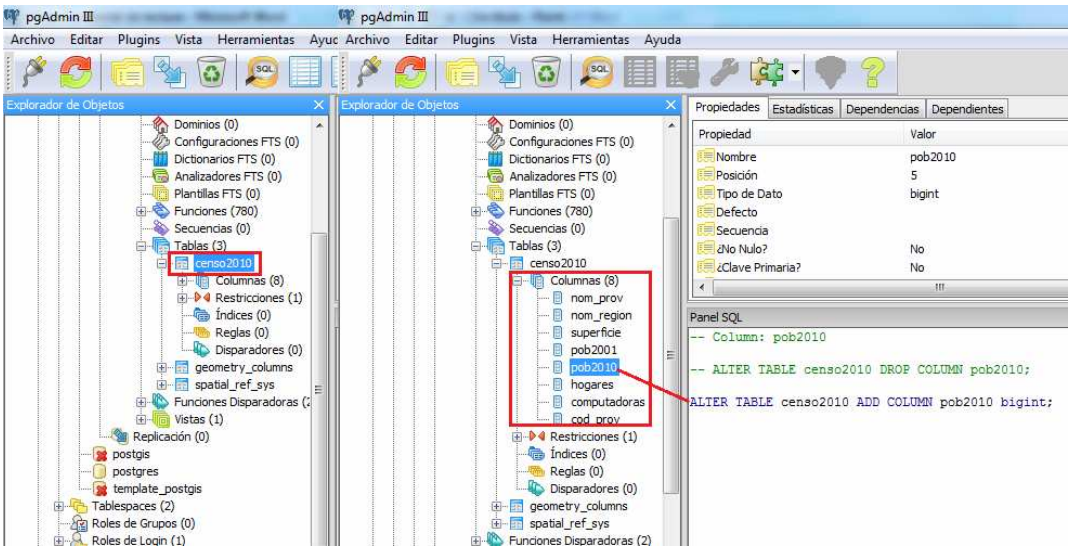
	nom_prov character var	nom_region character var	superficie double precis	pob2001 bigint	pob2010 bigint	hogares bigint	computadora bigint	cod_prov [PK] characte
1	CIUDAD AUTON	CENTRO	200	2776138	2890151	1150134	789145	02
2	BUENOS AIRES	CENTRO	307571	13827203	15625084	4789484	2308740	06
3	CATAMARCA	NOA	102602	334568	367828	96001	34518	10
4	CORDOBA	CENTRO	165321	3066801	3308876	1031843	510197	14
5	CORRIENTES	NEA	88199	930991	992595	267797	86184	18
6	CHACO	NEA	99633	984446	1055259	288422	85393	22
7	CHUBUT	SUR	224686	413237	509108	157166	89422	26
8	ENTRE RIOS	CENTRO	78781	1158147	1235994	375121	164184	30
9	FORMOSA	NEA	72066	486559	530162	140303	36426	34
10	JUJUY	NOA	53219	611888	673307	174630	59114	38
11	LA PAMPA	SUR	143440	299294	318951	107674	51344	42
12	LA RIOJA	CUYO	89680	289983	333642	91097	38013	46
13	MENDOZA	CUYO	148827	1579651	1738929	494841	214868	50
14	MISIONES	NEA	29801	965522	1101593	302953	86162	54
15	NEUQUEN	SUR	94078	474155	551266	170057	90178	58
16	RÍO NEGRO	SUR	203013	552822	638645	199189	97439	62
17	SALTA	NOA	155488	1079051	1214441	299794	97607	66

ACTIVIDAD 2. Desplegar la tabla censo2010. Hacer una impresión de pantalla y agregarla al documento.

Vemos que se trata de una tabla con ocho campos (columnas) y 24 registros (filas). Y también podemos comprobar lo que vimos anteriormente en el script SQL con respecto al tipo de datos de cada campo.

Vemos que los campos nom_prov, nom_region y cod_prov son de tipo “character varing”, lo que significa que pueden almacenar todo tipo de caracteres alfanuméricos. El campo “superficie” es de tipo Double Presicion, lo que le permite almacenar números decimales. Y los demás campos son de tipo bigint, lo cual permite almacenar números enteros grandes.

De la misma manera que la tablas son objetos para Postgres, los campos o columnas también lo son. Es por eso que podemos encontrarlos desplegando aún más el árbol de objetos.



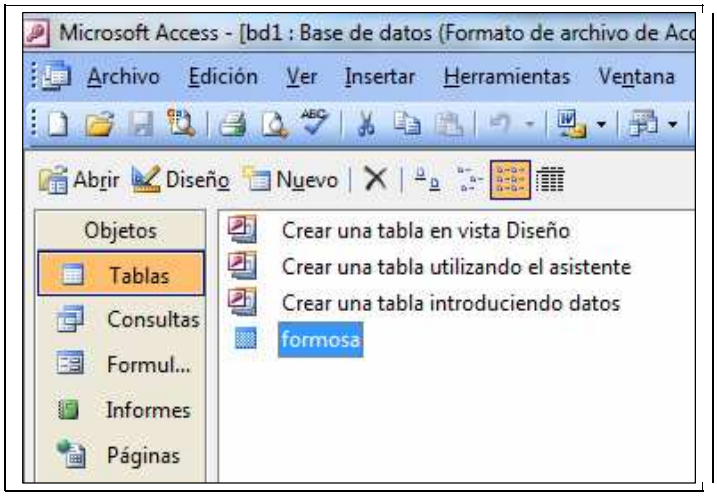
También al hacer un clic sobre un campo, podemos ver sus propiedades y conocer la consulta SQL que lo origina.

```
ALTER TABLE censo2010 ADD COLUMN pob2010 bigint;
```

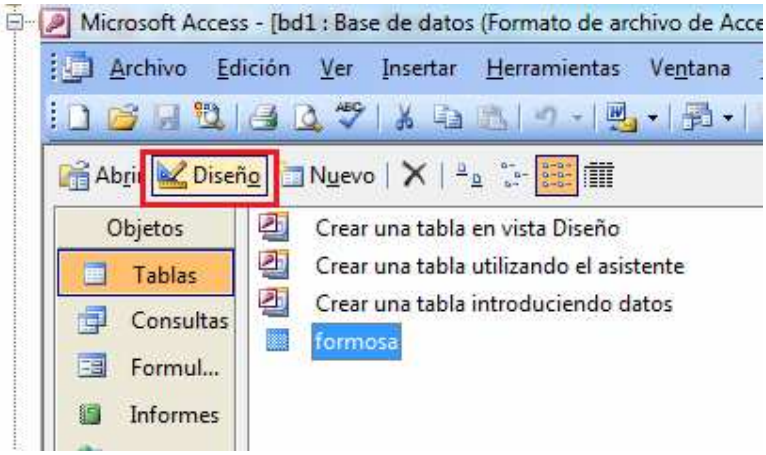
4 Migración de datos alfanuméricos desde Access

Se crea una nueva base de datos desde Archivo, Nuevo, Bases de datos en blanco.

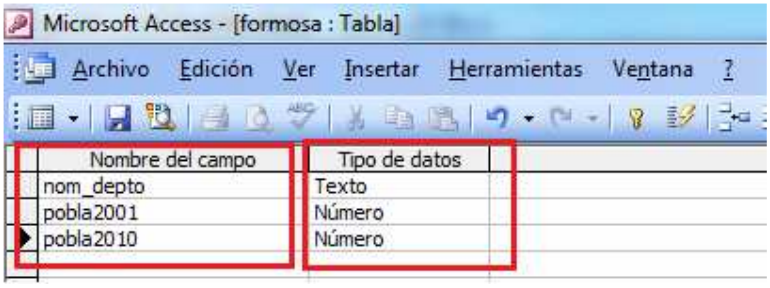
Luego desde Insertar, Tabla, Importar Tabla traemos la tabla en formato dbf, Excel, csv, etc..



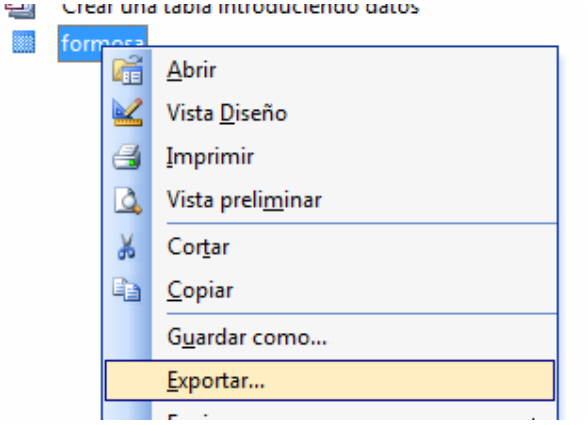
Una vez que tenemos la nueva tabla en Access, conviene preparar la estructura de la tabla para que sea compatible con Postgres. Esto básicamente consiste en renombrar los campos quitando espacios en blanco y letras mayúsculas. Esto se hace desde el botón Diseño.



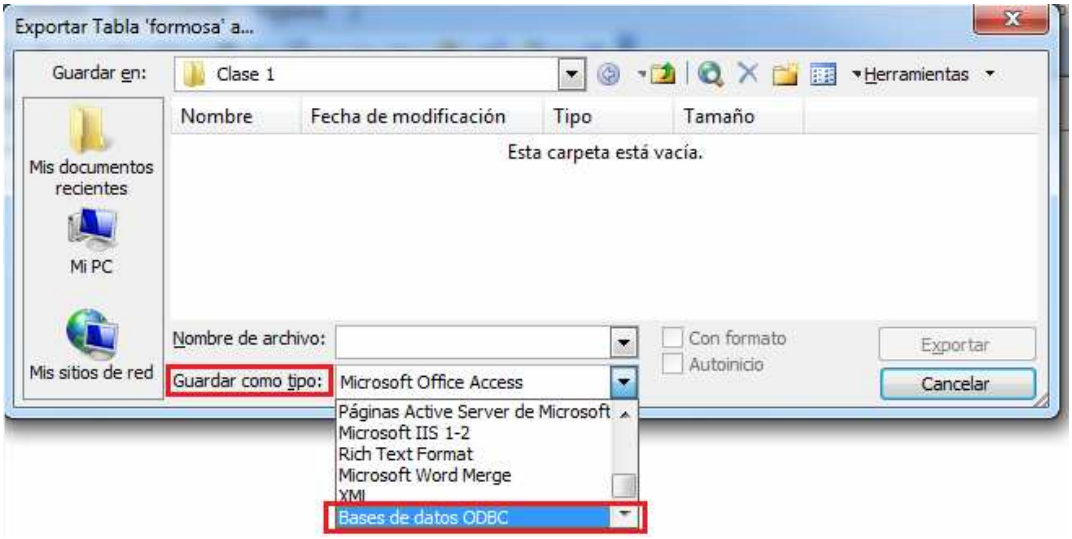
También podemos aprovechar para corroborar los tipos de datos correspondientes a cada campo.



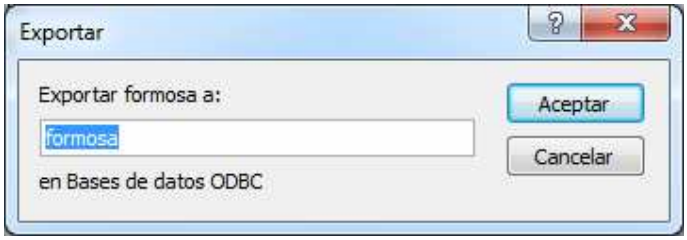
Una vez que adecuamos los campos de la tabla hacemos clic con el botón derecho sobre la tabla y elegimos la opción Exportar.



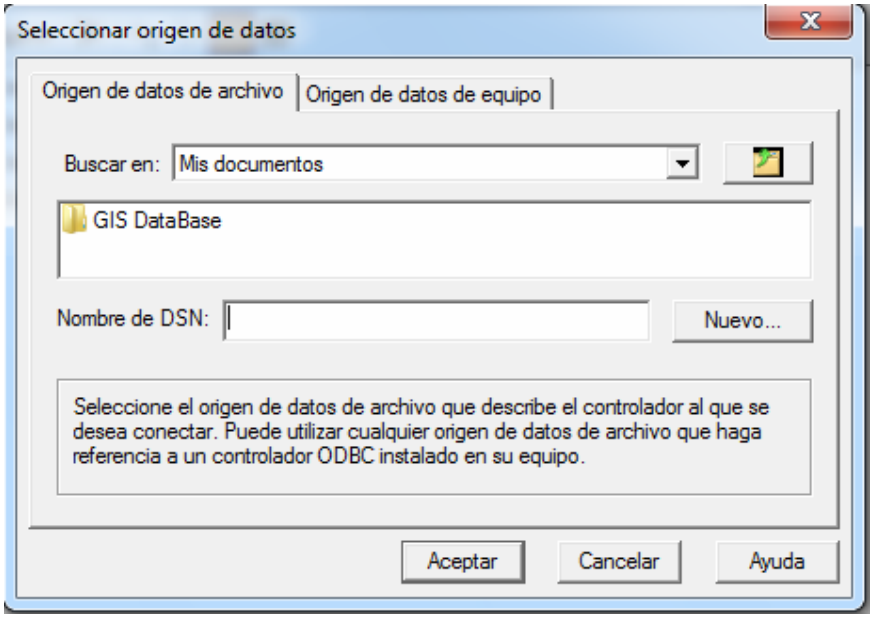
En la opción Guardar como tipo elegimos *Base de datos ODBC*. Esta es la razón por la cual en la clase anterior instalamos el controlador de psqlODBC, para poder establecer comunicaciones entre los distintos motores de bases de datos.



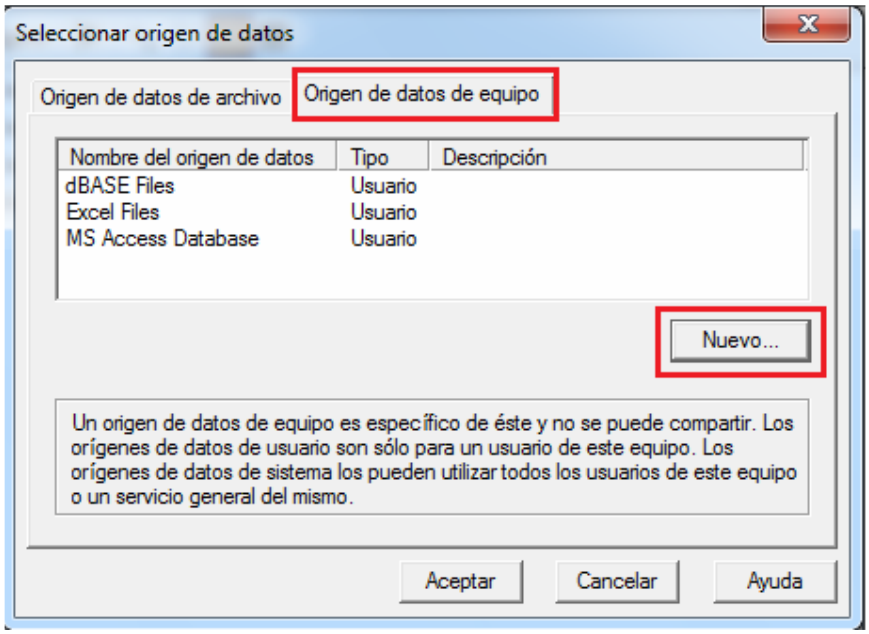
Luego Access nos pide el nombre de la nueva tabla.



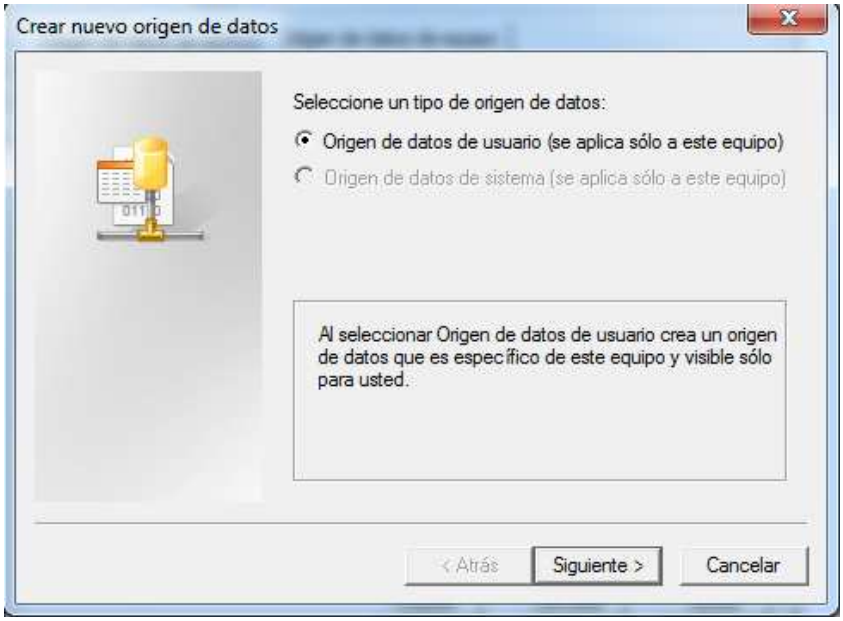
Luego nos pide seleccionar el origen de los datos.



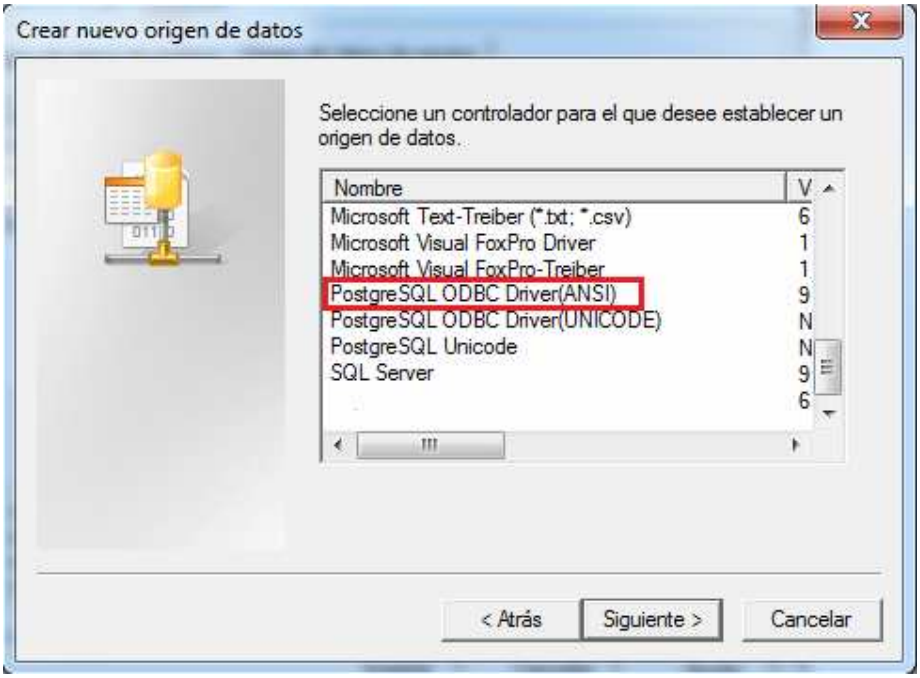
Vamos a la solapa *Origen de datos del equipo* y allí apretamos *Nuevo…*



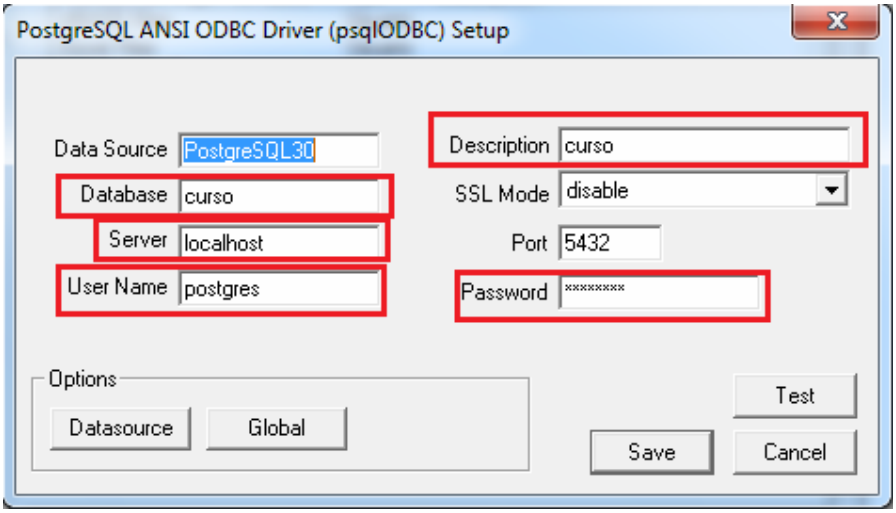
Allí seleccionaremos un nuevo origen de datos para el usuario o para el equipo. Esta segunda opción solo estará disponible si estamos logueados como administradores del equipo. Luego *Siguiente*.



A continuación elegimos el controlador del origen de datos: PostgreSQL ODBC Driver (ANSI), y apretamos *Siguiente* y luego Finalizar.

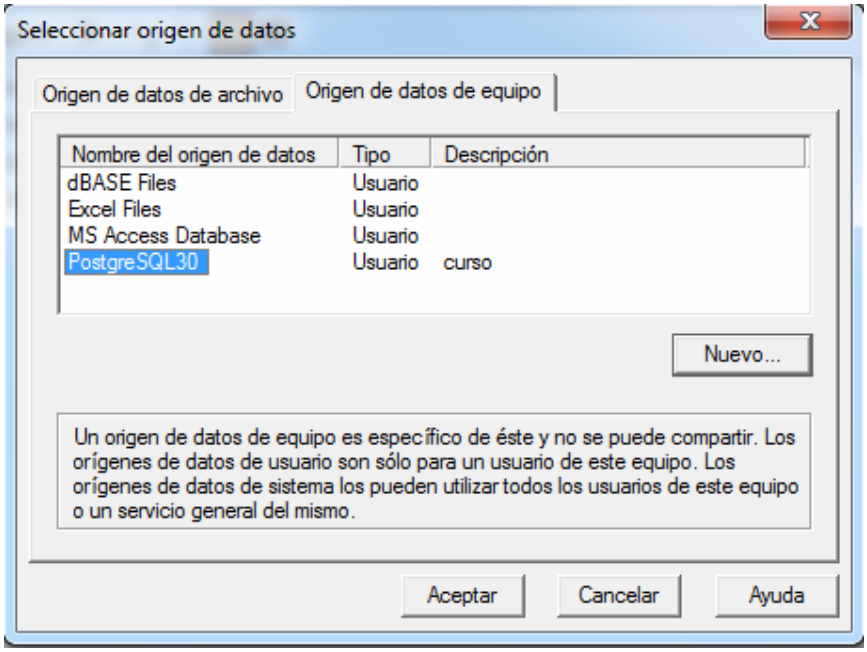


Luego nos pide en una nueva ventana los datos para la conexión. Como veremos, son muy similares a los utilizados para conectarnos desde PGAdmin, con la diferencia de que esta vez especificamos el nombre de la base de datos.



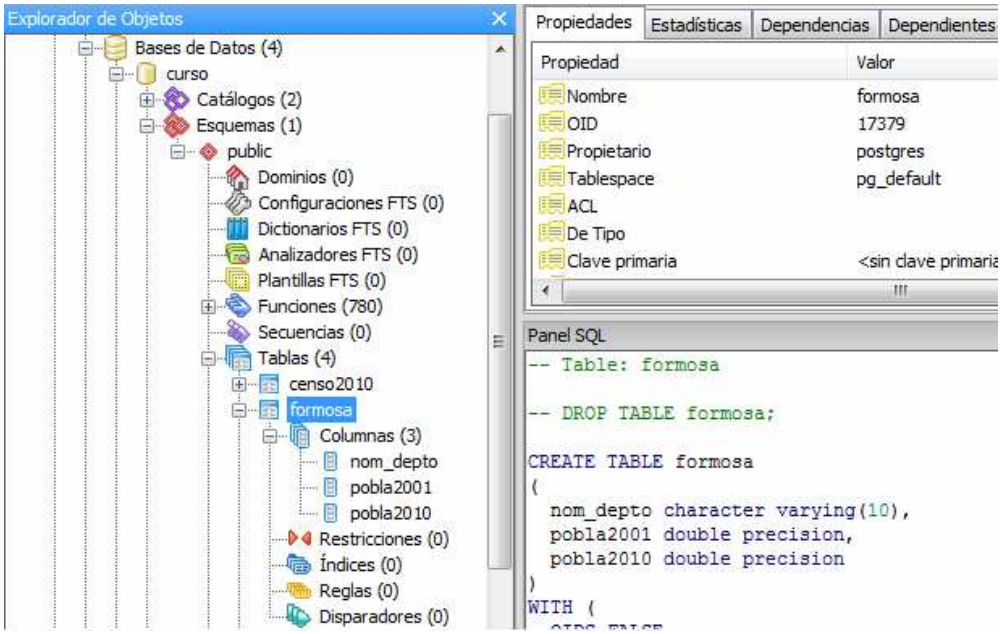
Para probar si los datos de conexión son correctos podemos apretar el botón *Test*.

Una vez creada la conexión nos aparece en el listado de orígenes de datos. La seleccionamos y apretamos *Aceptar*.



Cuando la tabla es pequeña la migración se produce casi instantáneamente. Si no, aparece una barra de estado que muestra la evolución del proceso.

Una vez migrada la tabla, podemos volver a PGAdmin y comprobar si la tabla ha sido migrada correctamente.



Recordemos Refrescar el ícono de *Tablas*.

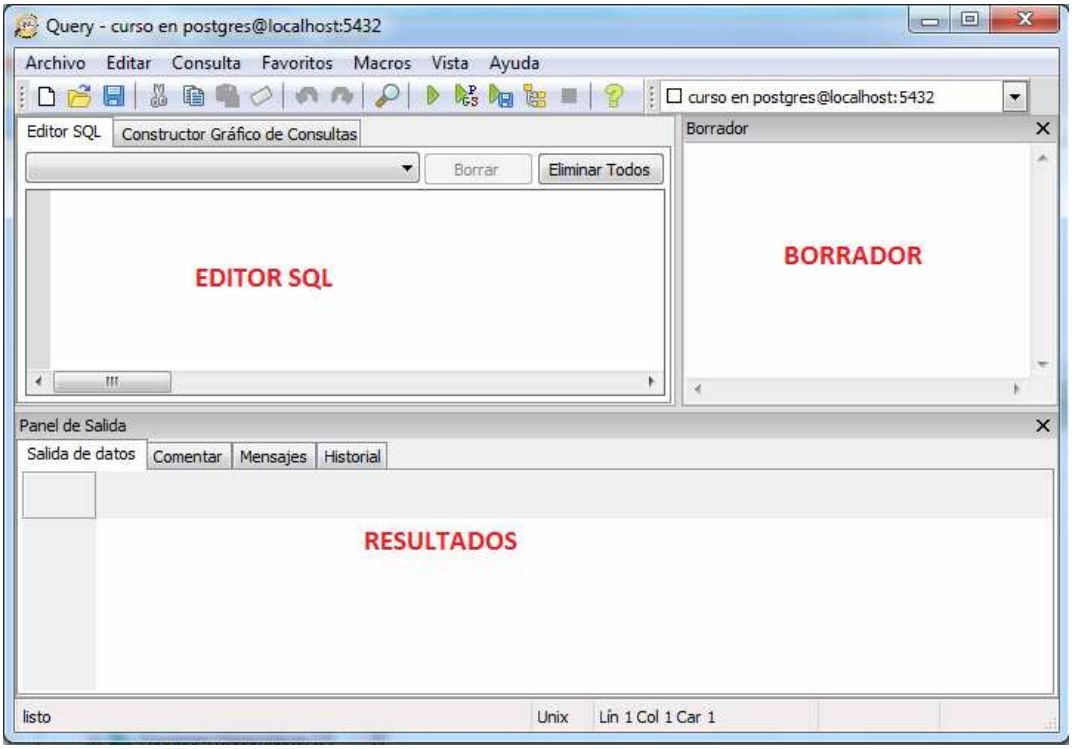
ACTIVIDAD 3. Migrar la tabla formosa.dbf utilizando Access vía OBDC. Visualizar la tabla en PGAdmin. Imprimir pantalla y agregarla al documento.

5 SQL. Comandos de selección

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL (Structured Query Language) o *Lenguaje Estructurado de Consultas*, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Veremos los comandos más sencillos que son aquellos que permiten realizar una selección de campos de las tablas.

Para poder ejecutar los comandos, necesitamos entrar al *Constructor de consultas arbitrarias* con el botón  .



En el sector del Editor de consultas es donde se escriben los comandos. En la parte inferior se muestran los resultados, y el ángulo superior derecho se utiliza para escribir consultas en borrador, o anotaciones que sirven para armarlas consultas.

La consulta de selección tiene la siguiente estructura:

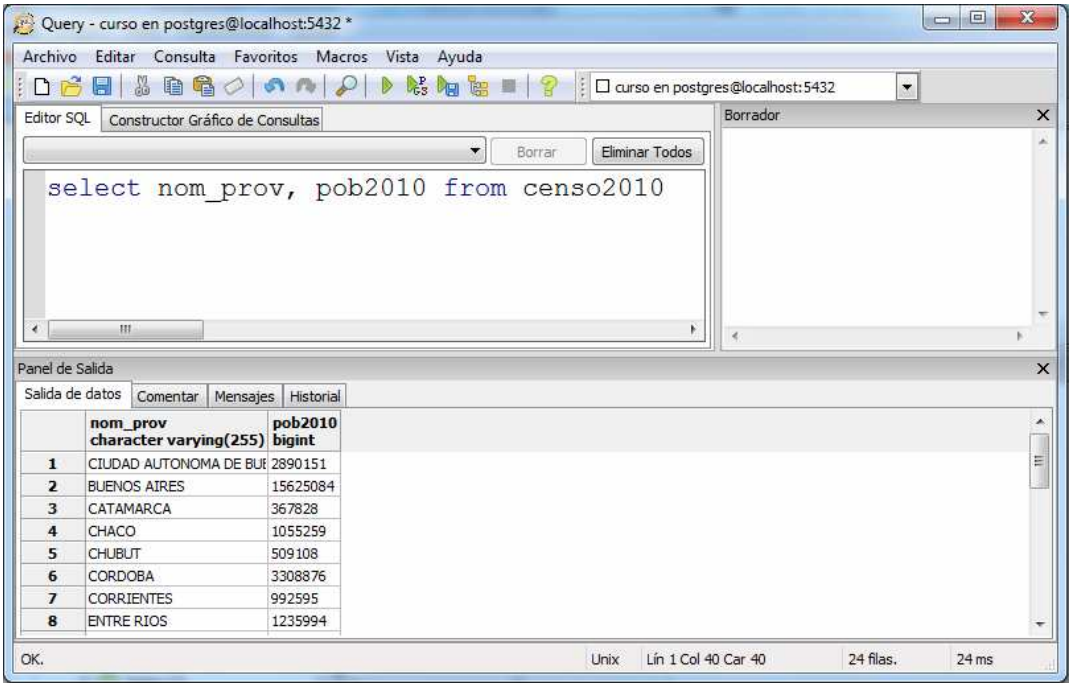
```
Select campos from tablas
```

campos es la enumeración de campos que queremos que nos devuelva la consulta, y **tablas** es el origen de estos campos.

Vamos a escribir la consulta que equivale a la acción de visualizar los campos de nombre de provincia y población del año 2010 de la tabla “censo2010”

```
select nom_prov, pob2010 from censo2010
```

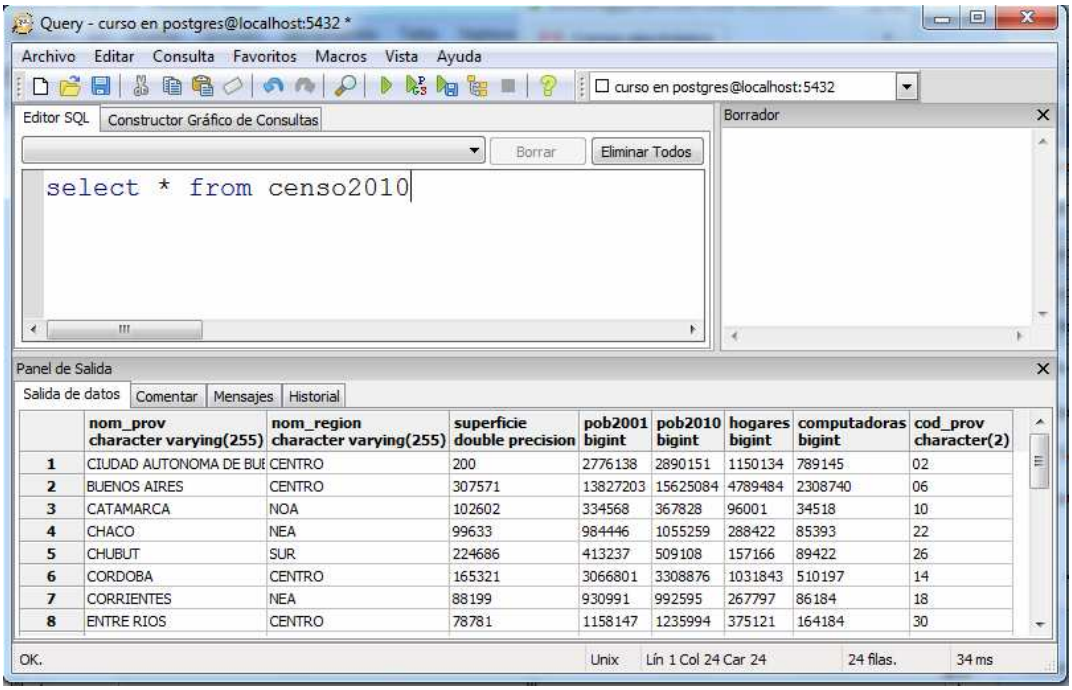
Escribimos la consulta en el sector superior del Constructor de consultas, y luego presionamos el botón Ejecutar consultas  .



Como vemos, el resultado es el contenido de los campos “nom_prov” y “pob2010” de la tabla “censo2010”.

Si deseamos traer todos los campos, podemos utilizar el asterisco “*” como comodín.

```
Select * from censo2010
```



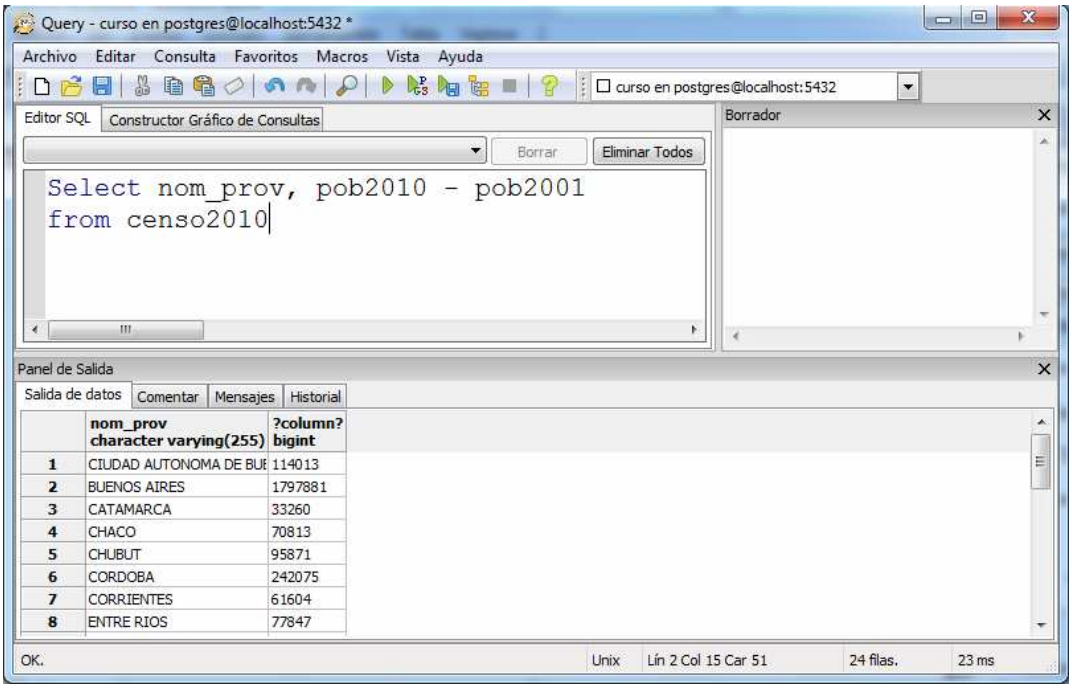
Como vemos arriba, esta consulta nos devuelve todos los campos de la tabla “censo2010”.

Dentro de la misma consulta, también se pueden realizar operaciones matemáticas con el contenido de los campos o entre campos.

Por ejemplo, podemos conocer la diferencia de población entre el año 2010 y el año 2001 para cada provincia:

```
Select nom_prov, pob2010 - pob2001
from censo2010
```

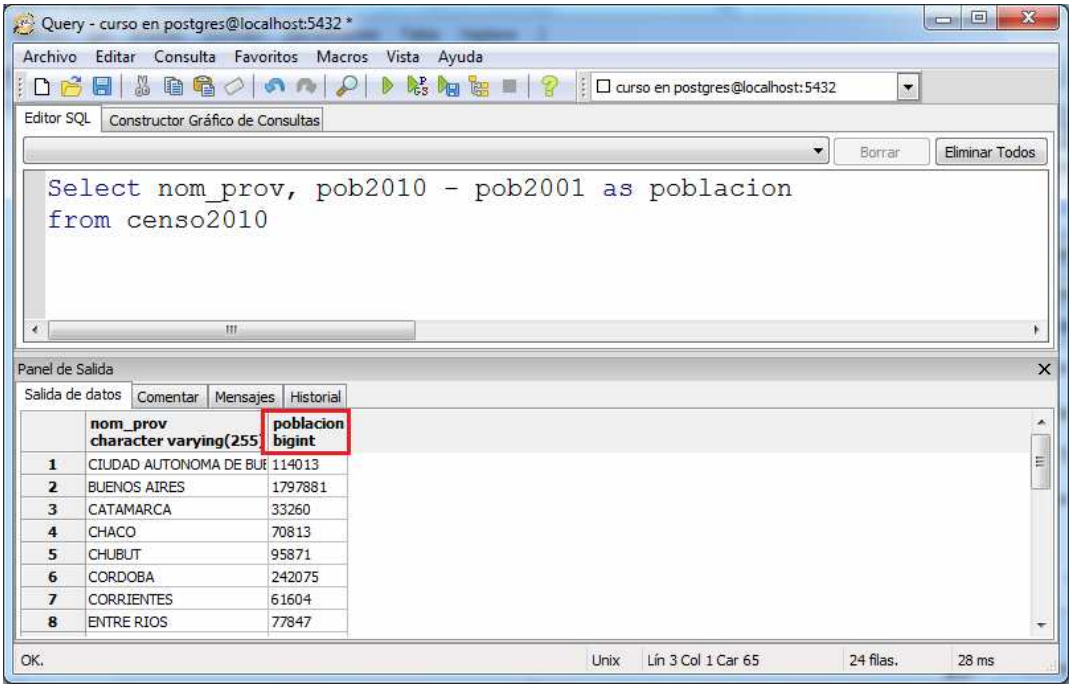
La operación matemática entre los dos campos (**`pob2010 - pob2001`**) se convierte en un solo campo que contiene el resultado de la resta.



Como el campo que resulta de la operación matemática no existía previamente, carece de nombre, por lo tan en el resultado aparece como “?column?”

Si deseamos darle un nombre a este nuevo campo, o a cualquier otro campo, podemos utilizar un Alias. El alias se agrega luego del campo.

```
Select nom_prov, pob2010 - pob2001 as poblacion
from censo2010
```



ACTIVIDAD 4. Escriba la consulta que trae el código de la provincia y la densidad de población por kilómetro cuadrado para el año 2010 (población del 2010 / superficie).
Colocar el alias "densidad" al campo correspondiente.
Imprimir pantalla con el resultado y añadir al documento.

6 Visualización de información geográfica de la base desde gvSIG

Vamos a agregar una nueva tabla, pero esta vez será una tabla con datos geométricos, para empezar a conocer como se componen.

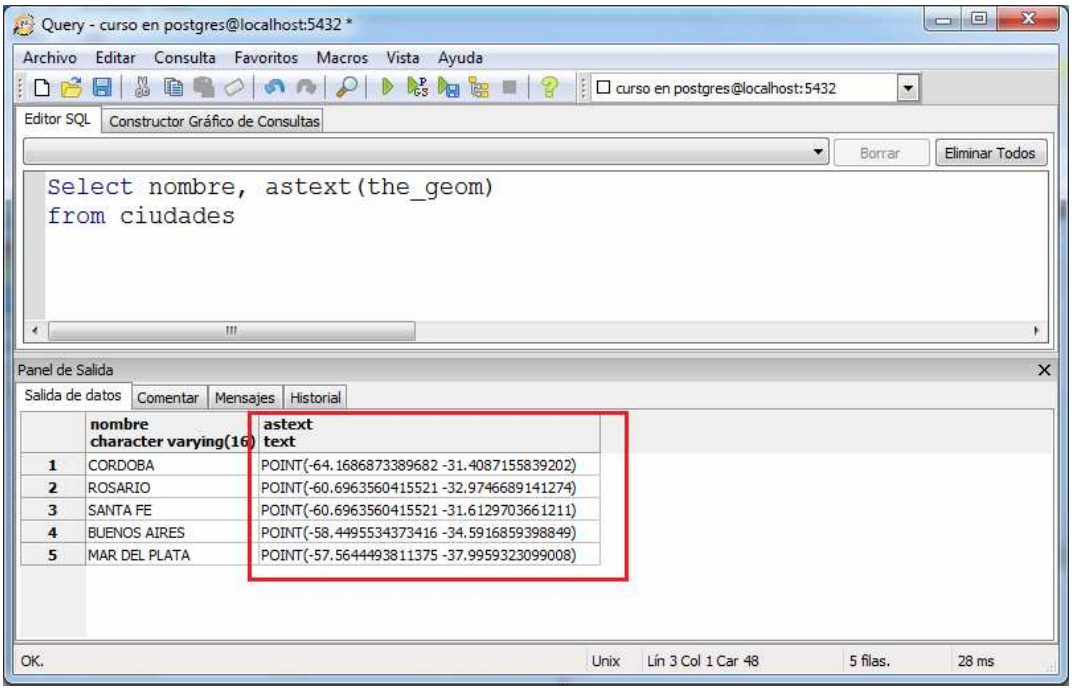
Restauramos el archivo ciudades.backup disponible en el material de práctica, y veremos que se agrega una nueva tabla llamada ciudades.

Si visualizamos la tabla con el botón *Ver los datos del objeto seleccionado*  , aparece el campo "the_geom" con el tipo de dato geometry.

	gid [PK] serial	nombre character var	the_geom geometry
1	1	CORDOBA	0101000020E61
2	2	ROSARIO	0101000020E61
3	3	SANTA FE	0101000020E61
4	4	BUENOS AIRES	0101000020E61
5	5	MAR DEL PLATA	0101000020E61
*			

Este campo es el que contiene los datos geométricos de las entidades representadas en las tablas, es decir, las ciudades. Pero los datos están almacenados en un formato difícil de interpretar. Por esto le aplicaremos una función al campo geométrico para saber qué hay almacenado allí.

```
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)

La consulta trae como resultado el contenido del campo geométrico pero traducido a un lenguaje que podemos comprender.

```
CORDOBA   POINT(-64.1686873389682 -31.4087155839202)
```

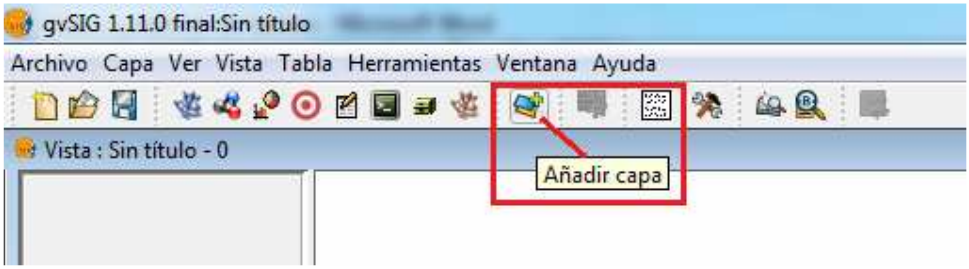
En primer lugar “POINT” indica el tipo de geometría. Si recordamos, los tipos de geometría que ya conocemos son las líneas, los puntos y los polígonos. Luego viene encerrado entre paréntesis la coordenada del punto en donde se encuentra la ciudad, expresado en grados decimales en este caso.

ATENCION: Para visualizar la tabla de ciudades en gvSIG, primero debemos "registrar" la columna geométrica haciendo correr la siguiente consulta SQL en el editor de consultas:

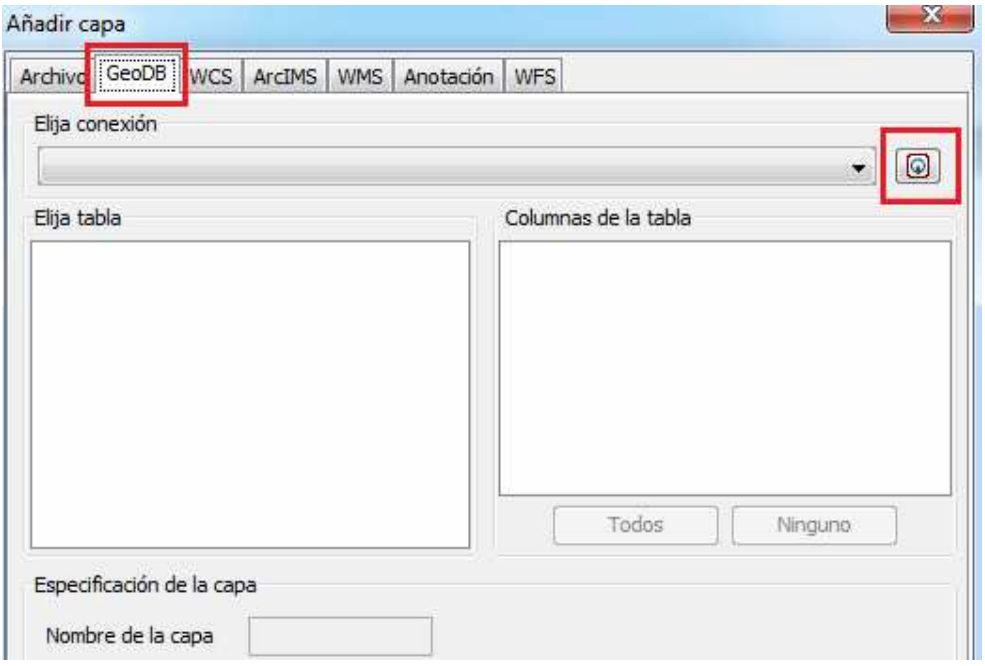
```
INSERT INTO geometry_columns VALUES ('', 'public', 'ciudades', 'the_geom', 2, 4326, 'POINT');
```

En la clase siguiente veremos de qué se trata el registro de columnas geométricas.

Estos mismos datos pueden ser visualizados desde gvSIG. Esto se logra desde el mismo botón que utilizamos para agregar un shapefile, pero seleccionando la solapa correspondiente a las bases de datos espaciales.

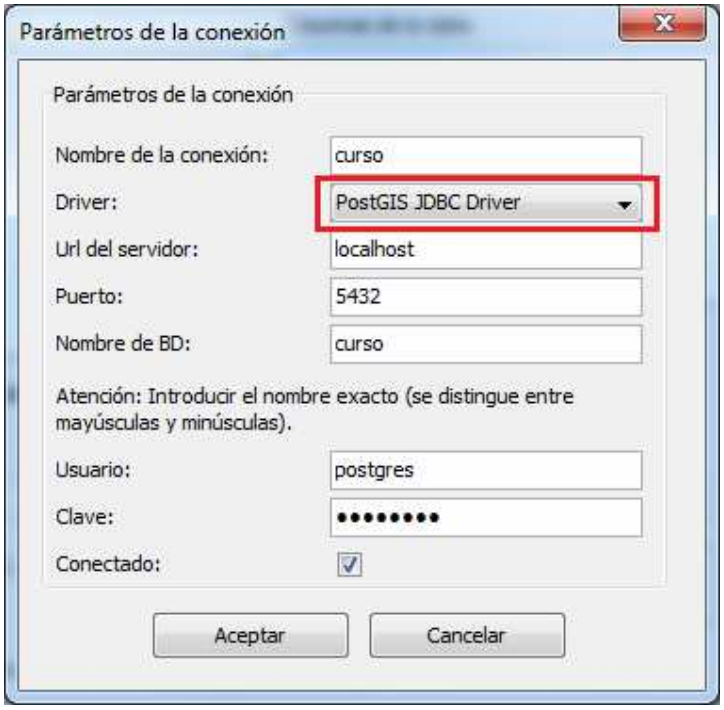


Seleccionamos la solapa GeoDB y luego presionamos el botón Añadir conexión  .

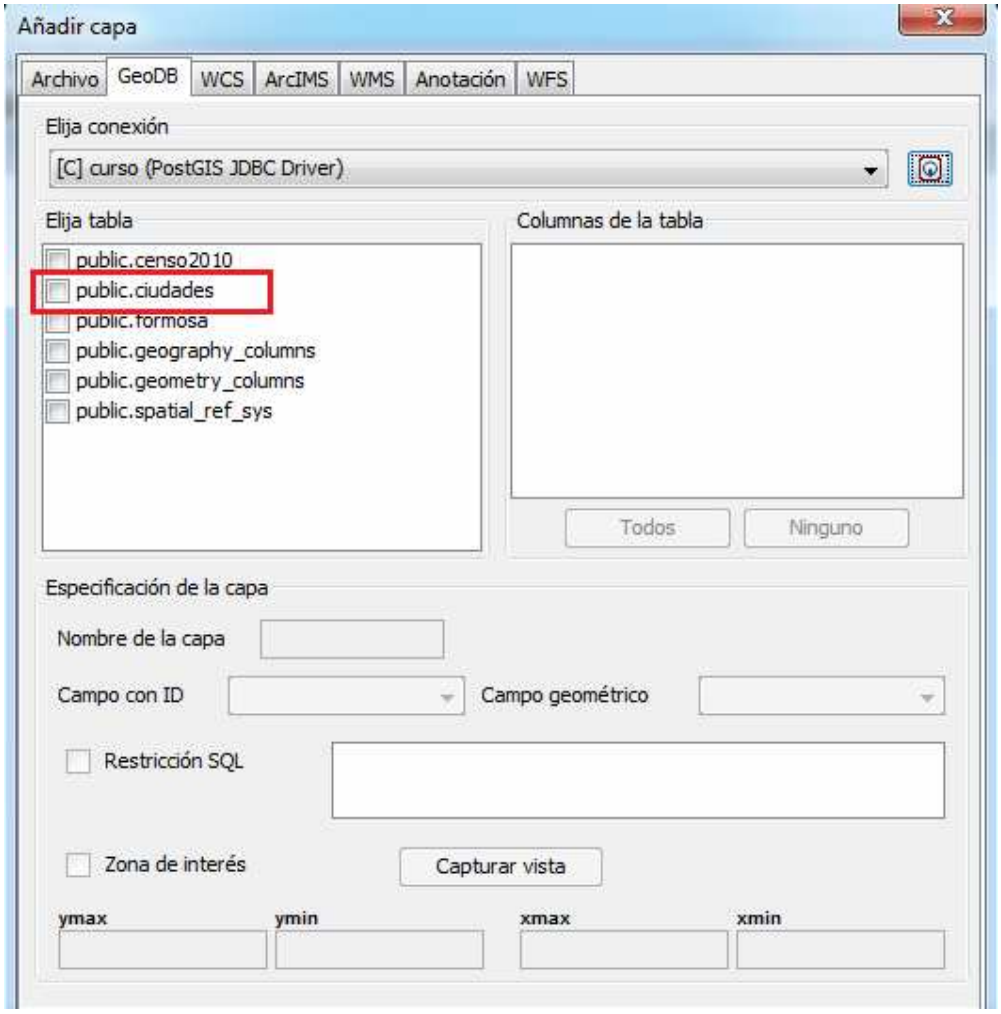


Esto nos abre una ventana que ya nos es familiar. Se trata de los parámetros para conectarnos a la base de datos, al igual que lo hicimos desde PGAdmin y el ODBC desde Access.

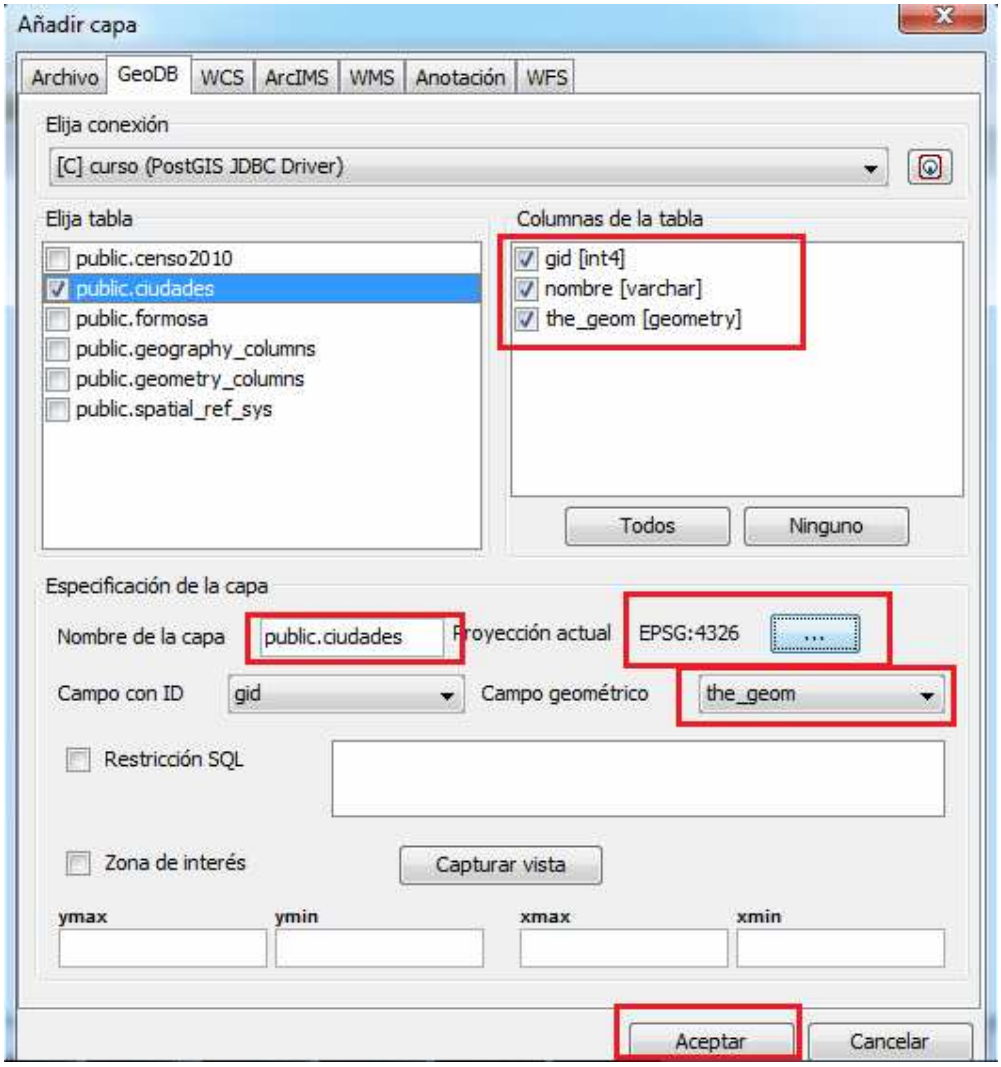
Solo debemos prestar atención al driver, que debe ser el de PostGIS



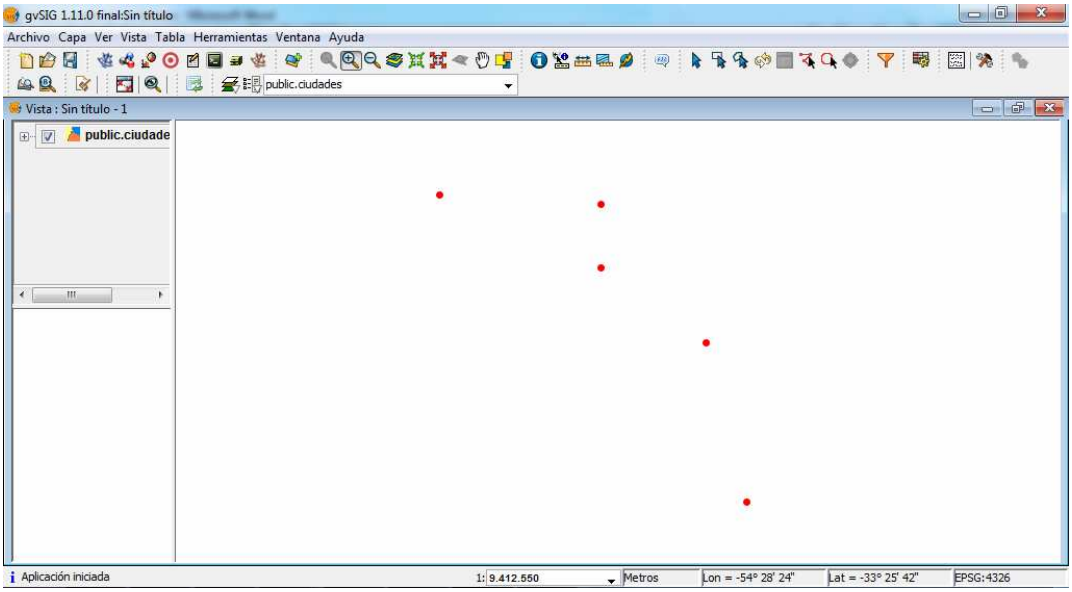
Luego de aceptar, volveremos a la ventana anterior, en donde se puede ver el listado de tablas existentes en la base de datos.



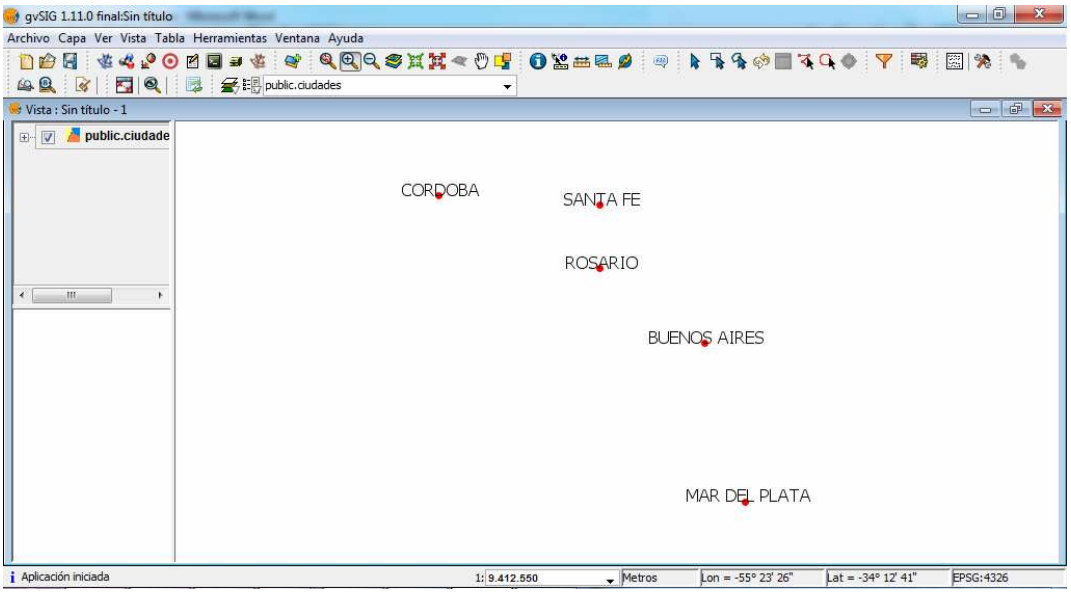
Cuando activamos el checkbox de la tabla de ciudades, vemos que se llenan los espacios referidos a los campos de la tabla, y de la columna geométrica, etc.



Al hacer clic en aceptar vemos que se agrega una capa, al igual que cuando añadimos un archivo shapefile, o de cualquier otro formato.



A partir de allí podemos continuar trabajando como lo haríamos con cualquier capa. Por ejemplo, se puede cambiar la simbología o etiquetar.



ACTIVIDAD 5. Ejecutar la consulta que se encuentra en el archivo "autopistas.sql" desde el constructor de consultas arbitrarias . (abrir archivo autopistas.sql desde allí).
Visualizar la capa de autopistas desde gvSIG.
Colocar una simbología llamativa, hacer una impresión de pantalla y agregarla al documento.

Para tener en cuenta: la capa de autopistas tiene el EPSG 4326

6.1 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.

Índice

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