



Universidad de El Salvador.
Facultad Multidisciplinaria de Occidente.
Departamento de Ingeniería y Arquitectura.

Cátedra: Programación III.
Semana 2

GUIA #2 – PostgreSQL,pgAdmin, pgModeler.

Objetivo: Que el estudiante se familiarice con postgresQL y algunas herramientas gráficas para modelar y administrar sus bases de datos.

Indicaciones: La presente guía se debe cubrir en su totalidad en los 150 minutos de práctica. Los ejercicios o asignaciones que se propongan ya sea al final de la guía o la hora de su respectiva práctica, se tendrán que resolver de la forma en que se especifique y tomando en consideración todas las instrucciones que se describan. El sistema operativo en que se trabaja toda la guía es Debian Jessie.

Instalación de PostgreSQL 9.4

1. Via terminal instalar los paquetes necesarios

```
sudo apt-get install postgresql-9.4 postgresql-client-9.4
```

2. Verificar la correcta instalación de postgresql

```
ps -ef | grep postgre
```

Si se ha realizado sin ningún problema la instalación, el comando anterior deberá mostrar algo igual o similar:

```
gual@sallyDebian:~$ ps -ef | grep postgre
postgres 3086 1 0 22:13 ? 00:00:00 /usr/lib/postgresql/9.4/bin/postgres -D /var/lib/postgresql/9.4/main -c config_file=/etc/postgresql/9.4/main/postgresql.conf
postgres 3088 3086 0 22:13 ? 00:00:00 postgres: checkpoint process
postgres 3089 3086 0 22:13 ? 00:00:00 postgres: writer process
postgres 3090 3086 0 22:13 ? 00:00:00 postgres: wal writer process
postgres 3091 3086 0 22:13 ? 00:00:00 postgres: autovacuum launcher process
postgres 3092 3086 0 22:13 ? 00:00:00 postgres: stats collector process
gual 4305 4206 0 22:33 pts/2 00:00:00 grep postgre
gual@sallyDebian:~$
```

Configuración de PostgreSQL 9.4

Debian divide la configuración de los archivos de las bases de datos, opuesto a la instalación genérica de PostgreSQL, que pone todo en un sólo directorio. Importante de mencionar, es que Debian permite diferentes clusters (*colección de bases de datos que están administradas por una sola instancia del servidor*) y diferentes versiones de PostgreSQL co-existan en el mismo host.

Archivos de configuración: /etc/postgresql/[version]/[cluster]/

Binarios: /usr/lib/postgresql/[version]

Bases de datos: /var/lib/postgresql/[version]/[cluster]/base

La ubicación de las bases de datos es por defecto el path anterior, acá se encuentra una carpeta por cada base con el número de su respectivo OID (identificador de objetos).

1. Crear usuarios “progra32016” y “mortal32016” que serán utilizados a lo largo del curso, cada uno tendrá una serie de permisos.

Algunos de los parámetros que se pueden especificar a la hora de crear un usuario:

- d: Permite al user crear bases de datos
- l: Para permitirle al user ingresar.
- P: Establece una contraseña para el user, que será necesaria en el momento de la autenticación.
- r: Permite al user crear roles.
- s: El user será un superusuario.

Roles en PostgreSQL

El manejo de roles en PostgreSQL permite diferentes configuraciones, entre ellas estan:

- SUPERUSER/NOSUPERUSER. Super usuario, privilegios para crear bases de datos y usuarios.
- CREATEDB/NOCREATEDB. Permite crear bases de datos.
- CREATEROLE/NOCREATEROLE. Permite crear roles.
- CREATEUSER/NOCREATEUSER. Permite crear usuarios.
- LOGIN/NOLOGIN. Este atributo hace la diferencia entre un rol y usuario. Ya que el usuario tiene permisos para acceder a la base de datos a traves de un cliente.
- PASSWORD. Permite alterar la contraseña.
- VALID UNTIL. Expiración de usuarios.

Antes de crear los usuarios es necesario ingresar en terminal con el usuario *postgres*. La contraseña para los usuarios será **diauesfmo**.

Se crea el usuario **progra32016** con los parámetros de la imagen:

```
gual@sallyDebian:~$ su
Password:
root@sallyDebian:/home/gual# su postgres
postgres@sallyDebian:/home/gual$ createuser -d -l -P -r -s progra32016
Enter password for new role:
Enter it again:
Password:
```

Se crea el usuario **mortal32016** con los parámetros de la imagen:

```
postgres@sallyDebian:/home/gual$ createuser -l -P mortal32016
Enter password for new role:
Enter it again:
Password:
```

2. Moverse al directorio de configuración de postgresql /etc/postgresql/9.4/main

```
cd /etc/postgresql/9.4/main/
```

Dentro de este path se encuentran distintos archivos:

```
gual@sallyDebian:/etc/postgresql/9.4/main$ ls
environment  pg_ctl.conf  pg_hba.conf  pg_ident.conf  postgresql.conf  start.conf
```

- **pg_hba.conf:** Define los diferentes tipos de accesos que un usuario tiene en el cluster.
- **pg_ident.conf:** Define la información necesaria en el caso que utilicemos un acceso del tipo ident en pg_hba.conf .
- **postgresql.conf:** Acá se pueden cambiar todos los parametros de configuracion que afectan al funcionamiento y al comportamiento de PostgreSQL en nuestra maquina.
- **pg_ctl.conf:** Inicializa, comienza, detiene o reinicia el server de postgresQL.
- **enviroment:** Puede ser usado para seleccionar por defecto parámetros de conexión.
- **start.conf:** Contiene configuración automática de inicio.

3. Acceder al archivo de configuración pg_hba.conf y editar

```
postgres@sallyDebian:/home/gual$ exit
exit
root@sallyDebian:/home/gual# exit
exit
gual@sallyDebian:~$ sudo nano /etc/postgresql/9.4/main/pg_hba.conf
[sudo] password for gual:
```

Agregamos los usuarios que acabamos de crear a este archivo, de esta manera:

```
# Database administrative login by Unix domain socket
local    all             postgres                                md5

# TYPE  DATABASE        USER            ADDRESS              METHOD
host    inventario   mortal32016     0.0.0.0/0            md5
host    all           progra32016     0.0.0.0/0            md5
# "local" is for Unix domain socket connections only
local    all             all              md5
# IPv4 local connections:
host     all             all             127.0.0.1/32         md5
# IPv6 local connections:
host     all             all             ::1/128              md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication    postgres        peer
#host    replication    postgres        127.0.0.1/32         md5
#host    replication    postgres        ::1/128              md5
```

La primera columna muestra el tipo de conexión, en el caso de los usuarios que creamos será *host*, que permite especificar los *hosts* que pueden conectarse al server de postgresQL, La segunda columna muestra la base datos, para *mortal32016*, la configuración es para la base *inventario*, mientras que para *progra32016*, la configuración es para todas las bases de datos. La tercera columna, muestra el user que tendrá el acceso. La cuarta columna especifica la dirección del cliente, acá se establece *0.0.0.0/0* para que acepte cualquier conexión remota de cualquier host. La última columna determina el método de autenticación, acá se establece *md5*, que requiere de una contraseña encriptada en md5.

4. Crear la base de datos “inventario”

Ingresar de nuevo como usuario *postgres* para poder crear la base de datos y establecer como dueño(owner) a *mortal32016* con el parámetro -O.

```
gual@sallyDebian:~$ su
Password:
root@sallyDebian:/home/gual# su postgres
postgres@sallyDebian:/home/gual$ createdb -O mortal32016 inventario
Password:
```

Luego de haber hecho toda la configuración, se reinicia el servicio de postgresQL

```
gual@sallyDebian:~$ sudo /etc/init.d/postgresql restart
[sudo] password for gual:
[ ok ] Restarting postgresql (via systemctl): postgresql.service.
gual@sallyDebian:~$
```

Ya que hemos creado los usuarios que nos servirán en todo el curso y hemos creado la base de datos, es hora de conocer una herramienta gráfica que nos facilite la el modelado de bases de datos en postgresQL, llamado pgModeler.

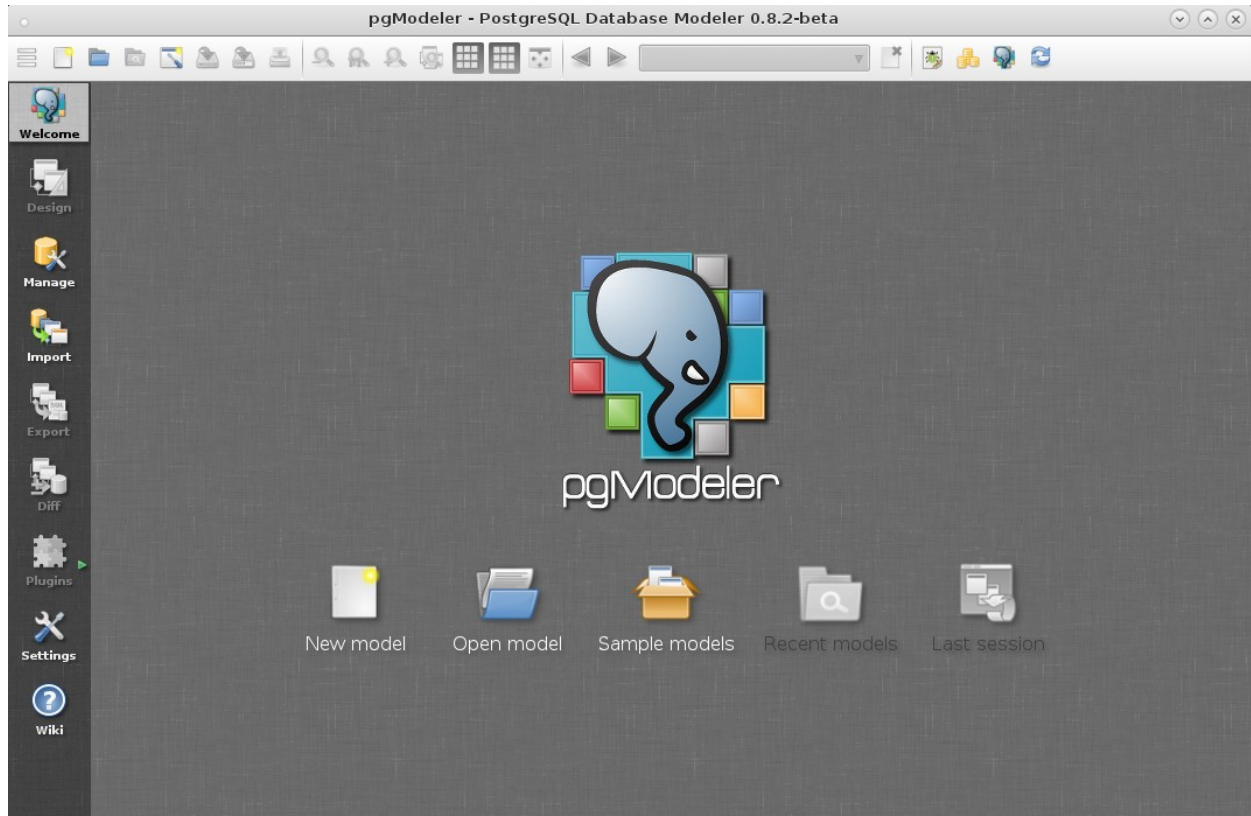
pgModeler

Es una herramienta de código libre para modelar base de datos que incluye el concepto clásico de entidad-relación con específicas características que sólo PostgreSQL implementa.

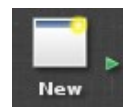
1. Iniciar pgModeler

```
gual@sallyDebian:~$ pgmodeler
```

Pantalla de inicio



Para crear un nuevo modelo hacer click en *New Model* y luego aparece el icono en la parte inferior izquierda, donde muestra los componentes disponibles.



Creación de tablas

tipo_documento y *documento*:

The 'Create / Edit: Table' dialog for the 'documento' table (ID: 4146) in the 'public' schema. The 'Columns' tab is active, showing a table with 6 columns: id_documento (serial, PRIMARY KEY), id_tipo_documento (integer, FOREIGN KEY, NOT NULL), codigo (char(255), NOT NULL), descripcion (char(255)), fecha_realiz... (date, NOT NULL), and ubicacion_fi... (text). The dialog includes fields for Name, Schema, Tablespace, Owner, Comment, Tag, and checkboxes for 'With OID', 'Unlogged', and 'Generate ALTER for columns/constraints'. It also has buttons for 'Custom SQL', 'Edit permissions', and 'Disable SQL code'. A warning message at the bottom states: 'The highlighted fields in the form or one of their values are available only on specific PostgreSQL versions. Generating SQL code for versions other than those specified in the fields' tooltips may create incompatible code.'

Name	Type	Default Value	Attribute
id_documento	serial	-	PRIMARY KEY
id_tipo_documento	integer	-	FOREIGN KEY, NOT NULL
codigo	char(255)	-	NOT NULL
descripcion	char(255)	-	-
fecha_realiz...	date	-	NOT NULL
ubicacion_fi...	text	-	-

En la pestaña de *Constraints* se agregan las relaciones de llaves primarias y foraneas.

The 'Create / Edit: Constraint' dialog for a PRIMARY KEY constraint. The 'Columns' tab is active, showing a table with 2 columns: id_documento (serial) and id_tipo_documento (integer). The 'Constraint Type' is set to 'PRIMARY KEY'. The 'Fill Factor' is 100, 'Deferrable' is unchecked, and 'Deferral' is 'INITIALLY IMMEDIATE'. The 'Column' field is set to 'id_tipo_documento (integer)'. The dialog includes fields for Name, Tablespace, Comment, and checkboxes for 'Fill Factor', 'Deferrable', and 'Disable SQL code'. It also has buttons for 'Custom SQL', 'Edit permissions', and 'Disable SQL code'. A warning message at the bottom states: 'Columns which were included by relationship can not be added / removed manually from the primary key. If done such changes they can raise errors. To create primary key using columns included by relationship use the following options: identifier field, attributes & constraints tab or primary key tab on the relationship form.'

Column	Type
id_documento	serial
id_tipo_documento	integer

Create / Edit: Constraint

* Name:

Comment:

☐ Disable SQL code

Constraint Type: FOREIGN KEY Match: MATCH FULL

Deferrable: ☐ Deferral: INITIALLY IMMEDIATE

ON DELETE: CASCADE ON UPDATE: CASCADE

Columns Referenced Columns

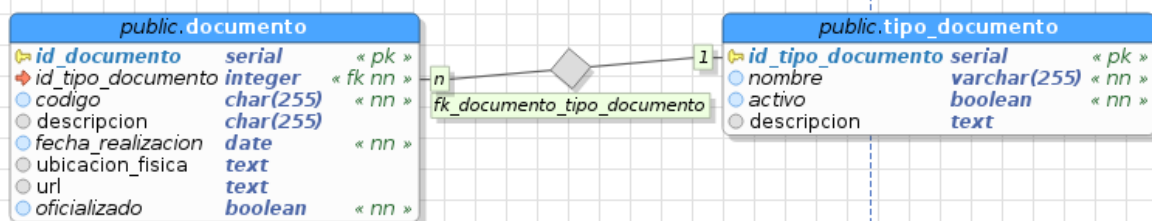
Table:

Column:

Column	Type
id_tipo_documento	serial

pgModeler

Resultado:

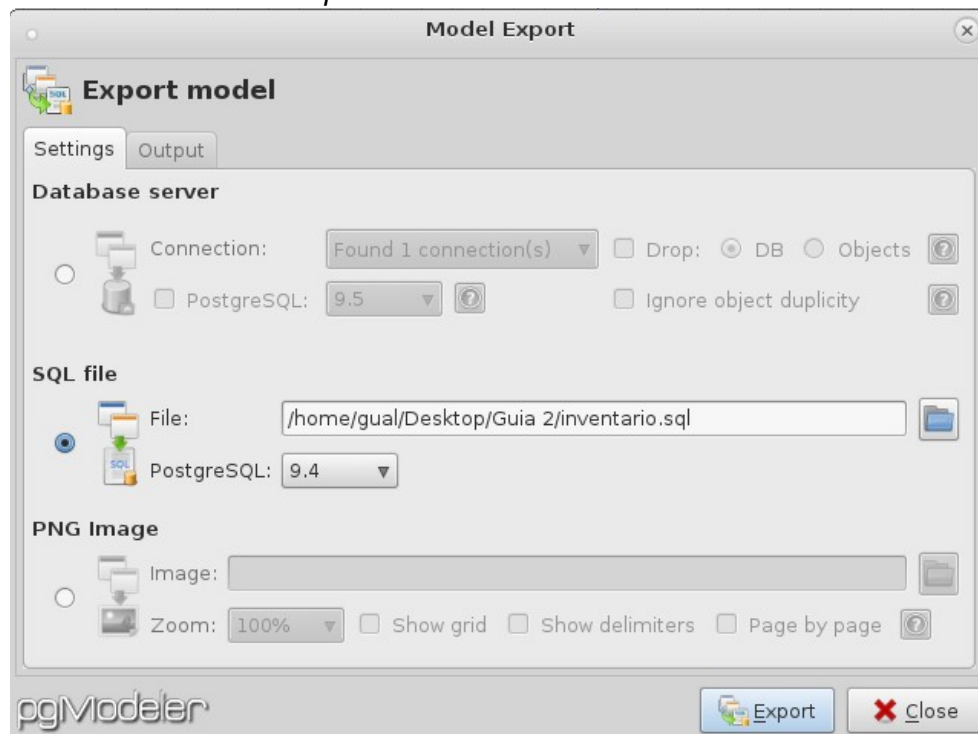


Una de las ventajas que nos ofrece pgModeler es la generación de scripts del modelado de las bases de datos; nos puede ser muy útil cuando trabajamos con bases de datos muy extensas, lo que nos ahorra mucho tiempo, ya que sólo es necesario ejecutar el script en vez de crear cada tabla y sus relaciones una por una.



Para crear el script se debe de dar click en el icono

Lo que nos muestra una ventana emergente, luego de establecido el path y el nombre sólo es necesario dar click en *Export*:



Asignación 1:

Crear las tablas *tipo_requisito*, *requisito*, *tipo_documento_requisito*, *tipo_documento* y *documento* con sus atributos y generar el script.

Teniendo ya elaborado nuestro script es hora de conocer una herramienta gráfica llamada pgAdmin, que nos facilite la administración de PostgreSQL y será donde vamos a ejecutar nuestro script.

pgAdmin III

Es una aplicación gráfica para administrar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está diseñada para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas.

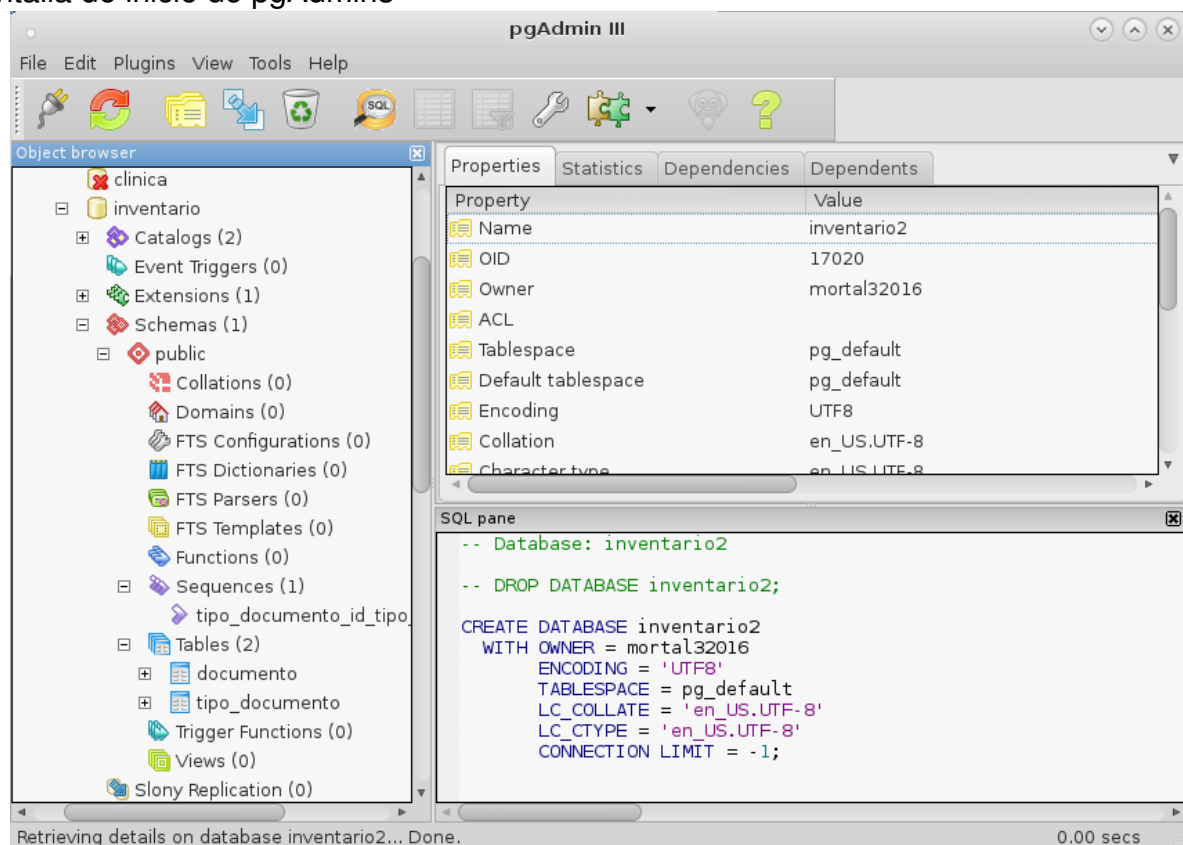
Instalación de pgAdmin III


1. Via terminal instalar los paquetes necesarios

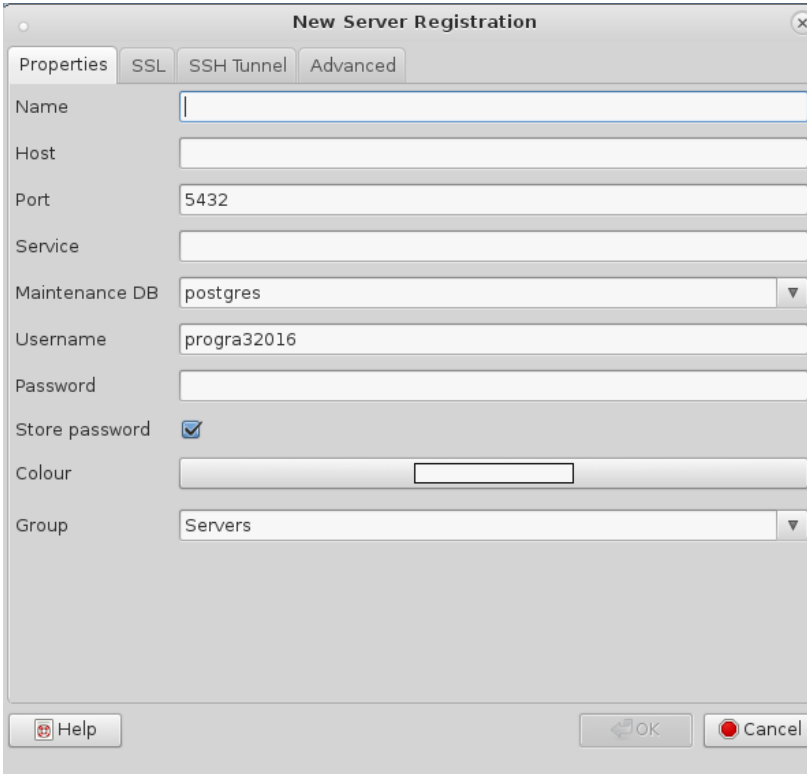
```
gual@sallyDebian:~$ sudo apt-get install pgadmin3
```

2. Abrir pgAdmin ya sea desde terminal escribiendo el comando *pgadmin3* o buscarlo en el menú de Aplicaciones de la distro.

Pantalla de inicio de pgAdmin3




Para agregar un servidor, se selecciona el icono de enchufe  de donde aparece la siguiente ventana emergente:

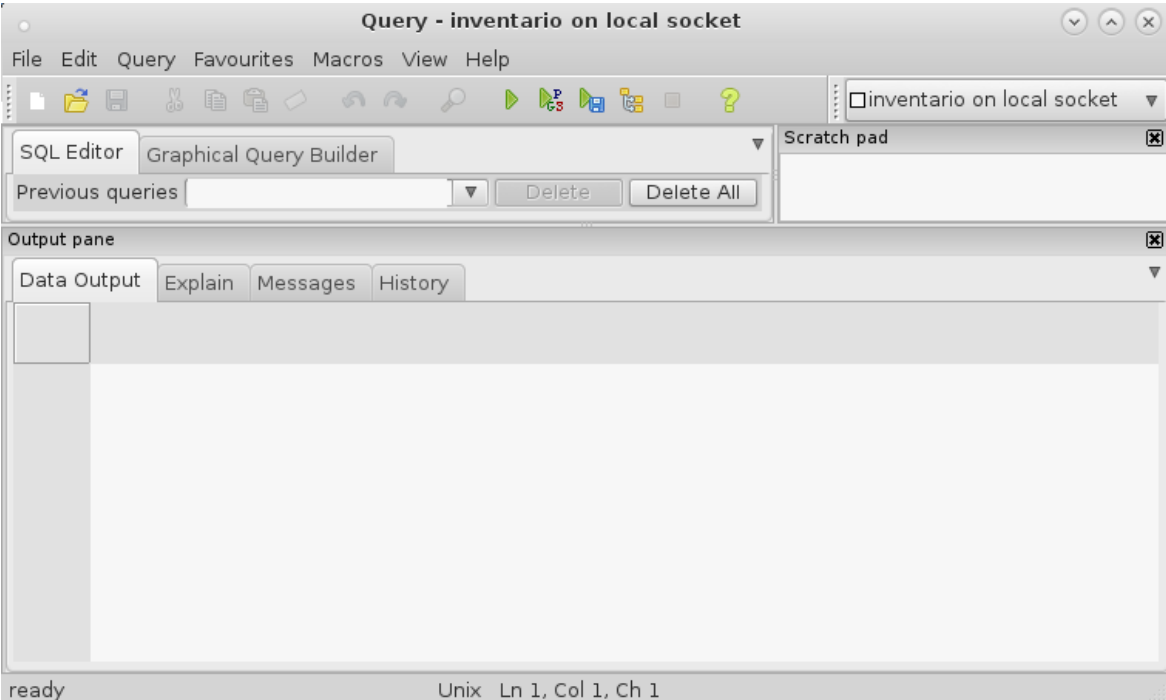


The 'New Server Registration' dialog box is shown with the 'Properties' tab selected. It contains the following fields and options:

- Name: Text input field.
- Host: Text input field.
- Port: Text input field with '5432' entered.
- Service: Text input field.
- Maintenance DB: Dropdown menu with 'postgres' selected.
- Username: Text input field with 'progra32016' entered.
- Password: Text input field.
- Store password: Checked checkbox.
- Colour: Two small adjacent text input fields.
- Group: Dropdown menu with 'Servers' selected.

At the bottom, there are buttons for 'Help', 'OK', and 'Cancel'.

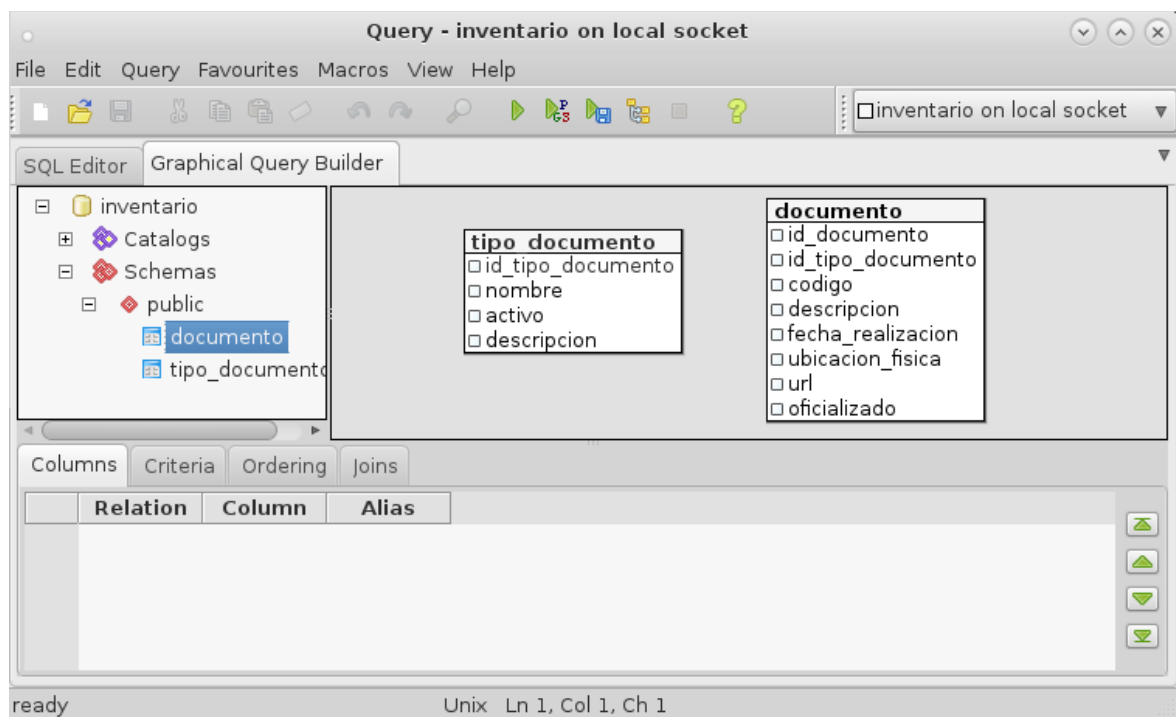
Existe una herramienta en pgAdmin llamada **QueryTool**  la cual permite ejecutar comandos SQL y además nos da la opción de analizar nuestra base de datos de forma gráfica.



The 'Query - inventario on local socket' window is shown. It features a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar with various icons. The main area is divided into several panes:

- SQL Editor**: Contains a 'Graphical Query Builder' tab and a 'Previous queries' dropdown with 'Delete' and 'Delete All' buttons.
- Scratch pad**: A text area for notes.
- Output pane**: Contains tabs for 'Data Output', 'Explain', 'Messages', and 'History'.

The status bar at the bottom indicates 'ready' and 'Unix Ln 1, Col 1, Ch 1'.



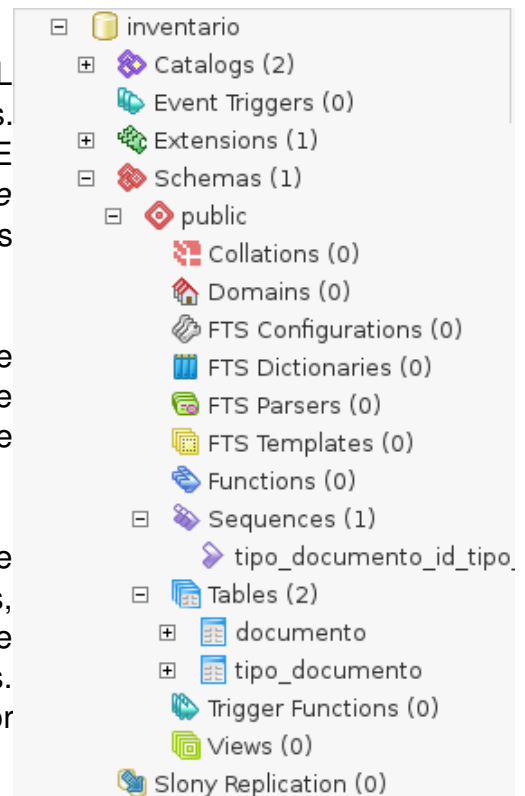
Estructura de una base de datos en postgresSQL.

Algunos de los componentes son:

Catalogs: Almacenan metadata sobre objetos SQL como tablas, columnas, funciones, esquemas y más. Son tablas regulares. Por ejemplo CREATE DATABASE ingresa una fila al catálogo *pg_database* con información sobre la base y crea la base de datos en el disco.

Triggers: Son funciones llamadas por la base de datos, que se ejecutan/invocan automáticamente cuando un evento específico ocurre en la base de datos.

Schema: Es una forma lógica de separar las bases de datos. Un esquema puede contener tablas, vistas, procesos almacenados (funciones), triggers, entre otros componentes usuales de una base de datos. Cada base de datos contiene un esquema *public* por defecto.



Functions: Conocidas también como *Stored Procedures* (procesos almacenados), permiten llevar a cabo operaciones que tomarían normalmente muchas queries y demás procesos en una sola función, dentro la base de datos.

Views: Son pseudo-tablas. Pueden representar un subconjunto de una tabla real, seleccionando ciertas columnas o ciertas filas de una tabla ordinaria. También pueden representar tablas combinadas. Las vistas tienen permisos separados, por lo que se pueden utilizar para restringir el acceso a usuarios y así sólo vean filas o columnas específicas de una tabla.

Extensions: Incluye múltiples objetos SQL. Por ejemplo un nuevo tipo de dato requerirá funciones, operadores, etc. Es útil coleccionar todos estos objetos en un sólo paquete para simplificar el manejo de la base de datos. PostgreSQL llama a ese paquete una *extensión*.

Sequences: Se emplea para generar valores enteros secuenciales únicos y asignárselos a campos numéricos; se utilizan generalmente para las claves primarias de las tablas garantizando que sus valores no se repitan.

Collations: Especifica el orden para la columna de una tabla o para una operación.

Domains: Es esencialmente un tipo de dato con restricciones opcionales (restricciones sobre el conjunto de valores permitidos).

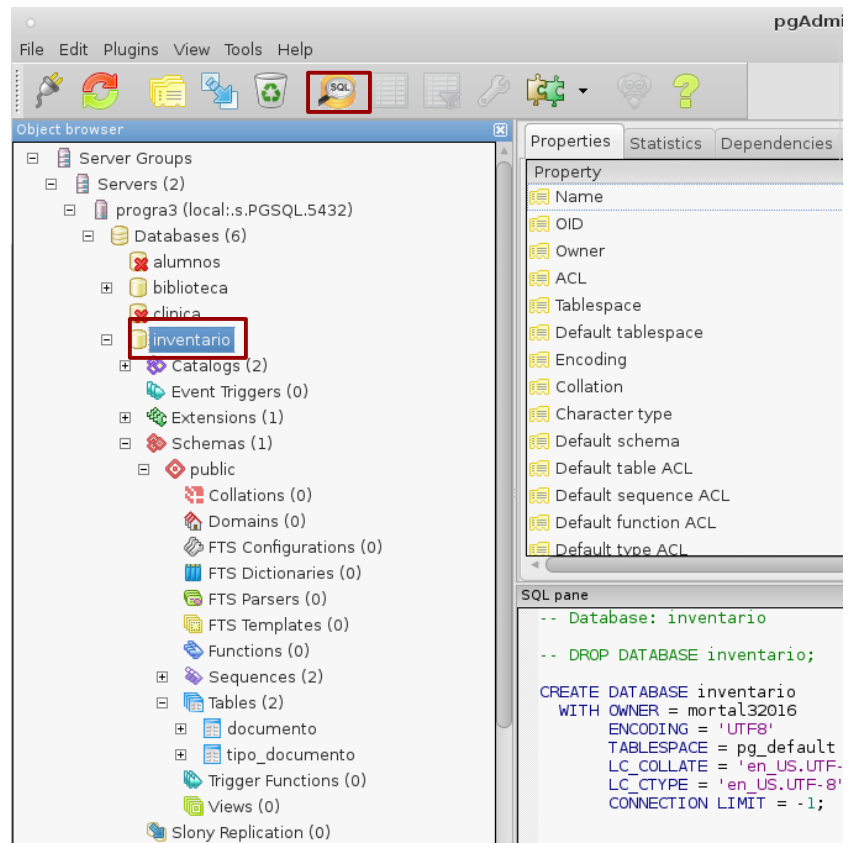
Indexes: Se utilizan para el que motor de búsqueda de la base de datos pueda agilizar el proceso de localizar registros. Desempeña la misma función que el índice de un libro.

Tipos de datos

Algunos tipos de datos de PostgreSQL son:

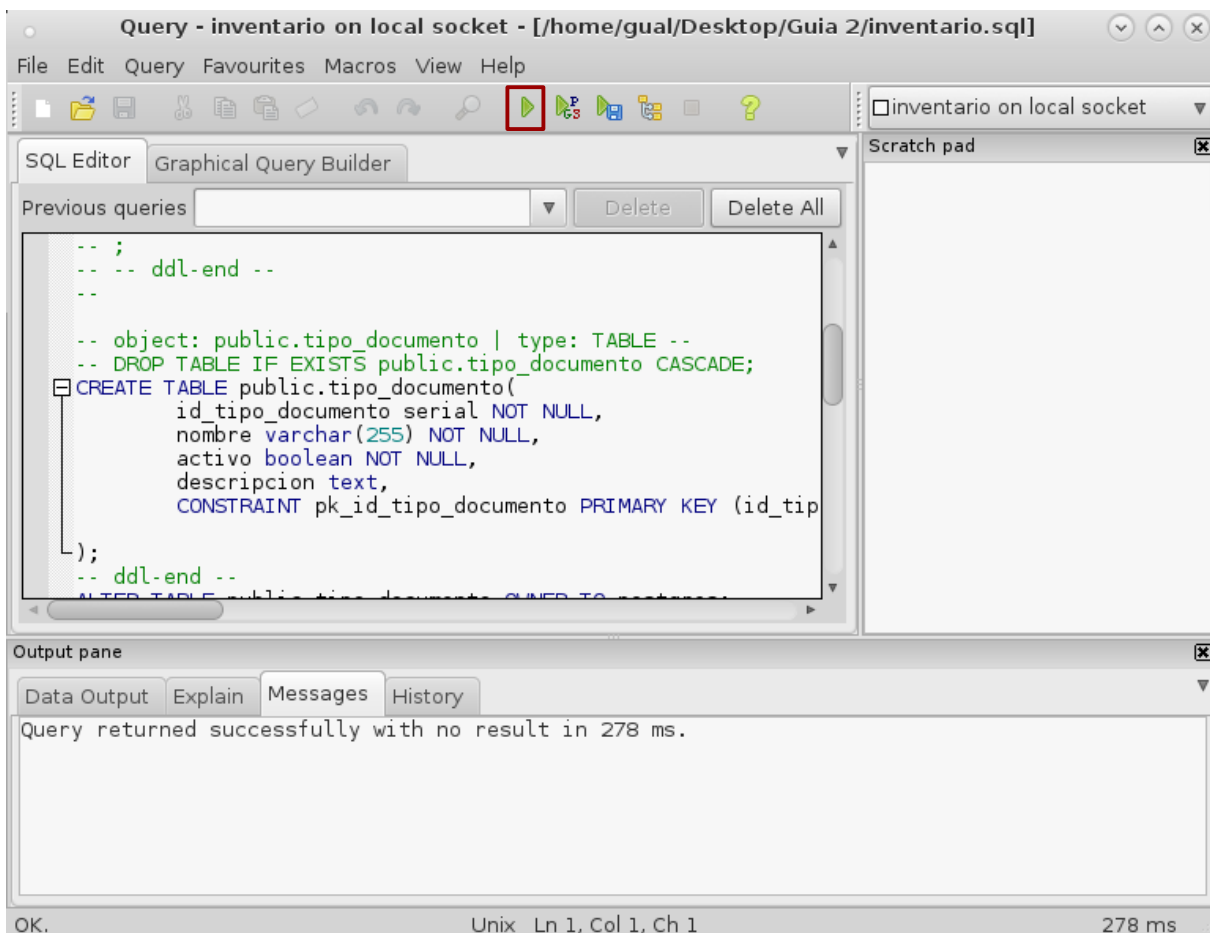
- **Character:** Puede ser abreviado *char*, es una cadena de caracteres de longitud fija, es decir, tiene un tamaño definido.
- **Character varying:** Es una cadena de caracteres de longitud variable que tiene un límite.
- **Text:** Cadena de caracteres de longitud variable sin un límite.
- **Integer:** Entero con signo de cuatro bytes.
- **Serial:** Entero autoincremental de cuatro bytes.
- **Bigserial:** Entero autoincremental de ocho bytes.
- **Bigint:** Entero con signo de ocho bytes.
- **Numeric:** Numérico exacto de precisión seleccionable.
- **Boolean:** Booleano lógico (verdadero/falso).
- **Date:** Fecha de calendario (año, mes, día).
- **Timestamp:** Fecha y hora, puede o no incluir zona horaria.

Para ejecutar el script en pgAdmin realizado anteriormente, necesitamos hacer uso de la herramienta QueryTool, pero antes se selecciona la base de datos *Inventario*, que creamos al inicio de la guía, la cuál por el momento no tiene la estructura de ninguna tabla y luego damos click en *QueryTool*.



Click en *Open File* y buscamos la ubicación donde está nuestro archivo sql; por último se ejecuta el script con *Execute query*:

Resultado:



Asignación 2:

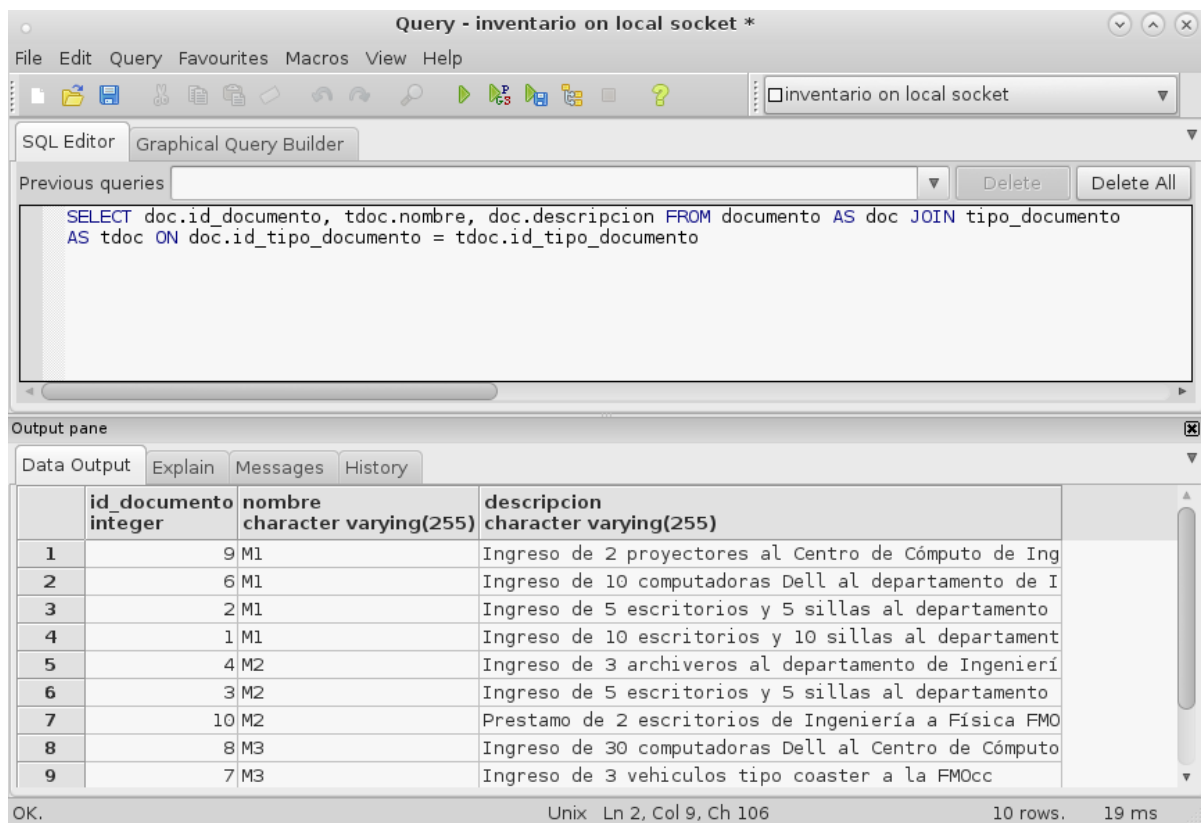
Llenar con algunos datos de prueba cada tabla creada de la asignación 1.

Consultas SQL

Como ya sabemos pgAdmin nos facilita la herramienta QueryTool para poder hacer consultas de forma ágil a nuestras bases de datos.

Ejemplos:

1. Mostrar los documentos registrados con su id, el nombre del tipo de documento al que pertenecen y su descripción.



The screenshot shows the Query Tool interface in pgAdmin. The title bar reads "Query - inventario on local socket *". The menu bar includes File, Edit, Query, Favourites, Macros, View, and Help. The toolbar contains various icons for file operations and query execution. The SQL Editor tab is active, displaying the following query:

```
SELECT doc.id_documento, tdoc.nombre, doc.descripcion FROM documento AS doc JOIN tipo_documento AS tdoc ON doc.id_tipo_documento = tdoc.id_tipo_documento
```

Below the SQL Editor is the Output pane, which has tabs for Data Output, Explain, Messages, and History. The Data Output tab is selected, showing the results of the query in a table with 9 rows. The table has three columns: id_documento (integer), nombre (character varying(255)), and descripcion (character varying(255)).

	id_documento integer	nombre character varying(255)	descripcion character varying(255)
1	9	M1	Ingreso de 2 proyectores al Centro de Cómputo de Ing
2	6	M1	Ingreso de 10 computadoras Dell al departamento de I
3	2	M1	Ingreso de 5 escritorios y 5 sillas al departamento
4	1	M1	Ingreso de 10 escritorios y 10 sillas al departament
5	4	M2	Ingreso de 3 archiveros al departamento de Ingenierí
6	3	M2	Ingreso de 5 escritorios y 5 sillas al departamento
7	10	M2	Prestamo de 2 escritorios de Ingeniería a Física FMO
8	8	M3	Ingreso de 30 computadoras Dell al Centro de Cómputo
9	7	M3	Ingreso de 3 vehiculos tipo coaster a la FMOcc

At the bottom of the window, the status bar shows "OK.", "Unix Ln 2, Col 9, Ch 106", "10 rows.", and "19 ms".

- Mostrar los documentos cuyo tipo de documento estén activos. Limitar el total de resultados a 3 registros, exceptuando los primeros dos.

The screenshot shows a SQL query editor window titled "Query - inventario on local socket - [/home/gual/Progra3/Guia 2/q1]". The query is as follows:

```
SELECT doc.id_documento, tdoc.id_tipo_documento, doc.codigo, doc.descripcion FROM
documento AS doc JOIN tipo_documento AS tdoc ON tdoc.activo = true AND
tdoc.id_tipo_documento = doc.id_tipo_documento ORDER BY doc LIMIT 3 OFFSET 2
```

The output pane shows the results of the query in a table with 5 columns: id_documento, id_tipo_documento, codigo, descripcion, and descripcion. The results are as follows:

	id_documento integer	id_tipo_documento integer	codigo character varying(255)	descripcion character varying(255)
1	5	3	1-5	Ingreso de 2 impresoras y 2 fotocopi
2	6	1	1-6	Ingreso de 10 computadoras Dell al d
3	7	3	1-7	Ingreso de 3 vehiculos tipo coaster

The status bar at the bottom indicates "OK.", "Unix Ln 3, Col 69, Ch 224", "3 rows.", and "13 ms".

En esta consulta hemos utilizado LIMIT para limitar el número de registros devueltos y OFFSET para especificar cuantas filas saltarse antes de empezar a mostrar los resultados, es importante utilizar la clausula ORDER BY cuándo se quiera utilizar LIMIT y OFFSET, para ordenar por un criterio único los resultados, de otra manera serán resultados inconsistentes.

3. Mostrar los documentos con su id, descripcion y los nombres de todos sus requisitos obligatorios, ordenados por id_documento.

Query - inventario on local socket *

File Edit Query Favourites Macros View Help

SQL Editor Graphical Query Builder

Previous queries Delete Delete All

```
SELECT DISTINCT req.nombre, doc.id_documento, doc.descripcion FROM documento AS doc
JOIN tipo_documento AS tdoc ON doc.id_tipo_documento = tdoc.id_tipo_documento
JOIN tipo_documento_requisito AS tdr ON tdoc.id_tipo_documento = tdr.id_tipo_documento
JOIN requisito AS req ON tdr.id_requisito = req.id_requisito WHERE tdr.obligatorio = true
ORDER BY id_documento
```

Output pane

Data Output Explain Messages History

	nombre character varying(255)	id_documento integer	descripcion character varying(255)
1	cargo_autorizador	1	Ingreso de 10 escritorios y 10 sillas al departamento de
2	cargo_encargado_recibe	1	Ingreso de 10 escritorios y 10 sillas al departamento de
3	encargado_recibe	1	Ingreso de 10 escritorios y 10 sillas al departamento de
4	nombre_autorizador	1	Ingreso de 10 escritorios y 10 sillas al departamento de
5	unidad_recibe	1	Ingreso de 10 escritorios y 10 sillas al departamento de
6	cargo_autorizador	2	Ingreso de 5 escritorios y 5 sillas al departamento de I
7	cargo_encargado_recibe	2	Ingreso de 5 escritorios y 5 sillas al departamento de I
8	encargado_recibe	2	Ingreso de 5 escritorios y 5 sillas al departamento de I
9	nombre_autorizador	2	Ingreso de 5 escritorios y 5 sillas al departamento de I
10	unidad_recibe	2	Ingreso de 5 escritorios y 5 sillas al departamento de I
11	cargo_autorizador	2	Ingreso de 5 escritorios y 5 sillas al departamento de I

OK. Unix Ln 1, Col 64, Ch 64 64 rows. 14 ms