

# ***Práctica 12***

## ***Seguridad, Respaldo y Restauración***

### **1. Objetivo General**

Conocer métodos para incrementar el nivel de protección y recuperación de una base de datos.

### **2. Objetivos Secundarios**

- Definir usuarios y/o grupos
- Otorgar o negar permisos específicos a usuarios y/o grupos
- Respaldar y restaurar una base de datos

### **3. Introducción**

Debido al amplio alcance de las bases de datos, las tablas que las componen pueden almacenar datos de naturaleza muy distinta y variada. Entre ellos podemos incluir datos confidenciales, como por ejemplo: los datos de saldos bancarios, contraseñas, totales de venta, direcciones, correos electrónicos, teléfonos, etc. Por lo tanto, se puede intuir que los datos que contiene una base de datos pueden generar información que tenga un alto valor.

Por ésta razón es importante proteger los datos de una base de datos contra diversos factores que puedan afectar la integridad y confidencialidad de estos.

#### **3.1. Principales causas de daño a una base de datos**

Entre las principales causas de ataques o daños provocados a una base de datos podemos considerar los siguientes:

- Error humano
- Mal diseño de los niveles de seguridad
- Ataques

En las siguientes subsecciones se detallaran éstas causas.

##### **3.1.1. Error humano**

Un factor que afecta de manera constante a las bases de datos es el error humano. Este defecto es inherente a cualquier usuario que interactúe con la base de datos mediante un SMBD. Consiste en que cualquier usuario, por equivocación, puede construir y ejecutar una consulta que dañe de manera permanente los datos almacenados en la base de datos, comprometiendo la información que de ellos se pudiera derivar.

### ***3.1.2. Mal diseño de los niveles de seguridad***

Las bases de datos permiten el ingreso de múltiples usuarios simultáneamente para aumentar la eficiencia de la explotación de los datos que se encuentran almacenados en ella, de hecho, ésta es una de las grandes ventajas de los SMBD y de las bases de datos relacionales.

El problema reside en que no todos los usuarios realizan las mismas actividades ni tienen las mismas responsabilidades, por lo tanto sus accesos permitidos y privilegios dentro de la base de datos deberán de ser congruentes con sus actividades. Por ejemplo: si una pizzería tiene una base de datos donde almacenan los datos relacionados con clientes, empleados, órdenes y sus detalles, es normal pensar que el repartidor tenga acceso a las direcciones de los clientes más no de sus teléfonos, así como un cocinero deberá tener acceso a los detalles de la orden y no a los datos del cliente.

Si los niveles de seguridad no se diseñan de acuerdo al contexto del problema, puede generar filtraciones de información confidencial, resultando en un grave problema.

### ***3.1.3. Ataques***

Conforme la tecnología y el conocimiento para utilizarla se vuelve más accesible, existe mayor probabilidad de que alguna persona o grupo de personas utilice estas herramientas para causar daño intencionado o hurto de los datos que se almacenan en una base de datos.

A pesar de que los SMBD poseen cualidades para restringir accesos y permisos, no tienen elementos de control propios que potencialicen la seguridad de las bases de datos que almacenan contra éste tipo de ataques. Es por esto que si no se cuenta con otro tipo de control, resulta necesario tener precaución en cuanto a la localización de las computadoras y las redes que las comunican para evitar el acceso de personas no autorizadas que puedan violar la integridad de los datos almacenados.

## **3.2. Repuesta de los SMBD**

Si bien no se está exento de que ocurra un accidente como los que se mencionaron, los SMBD ofrecen herramientas que aumentan la seguridad de las base de datos y con esto reducen la probabilidad de sufrir pérdidas de información o consistencia de ésta.

Los SMBD ofrecen dos posibilidades para incrementar la seguridad de nuestra base de datos, mismas que se presentan a continuación.

### ***3.2.1. Creación de usuarios***

Los SMBD permiten definir usuarios a los cuales se le otorgará el acceso a la base de datos, adicionalmente se les puede asociar una contraseña, vigencia y permisos singulares que modelen los perfiles que se hayan definido en las primeras etapas de la construcción de la base de datos. En otras palabras, podemos hacer explícito quién accede a la base de datos, las tablas que puede

consultar y las acciones que puede realizar sobre los datos. Es decir, otorgando o revocando privilegios para realizar consultas, modificaciones, inserciones o eliminaciones de registros.

Los usuarios pueden ser agrupados, es decir, un conjunto de usuarios que comparten las mismas características pueden ser considerados bajo un mismo grupo, facilitando el manejo de los permisos, ya que éstos no tienen que ser indicados para cada usuario de manera individual, sino únicamente para el rol al que pertenecen. La Figura 12.1 muestra la jerarquía sobre la que se basa una agrupación de usuarios.

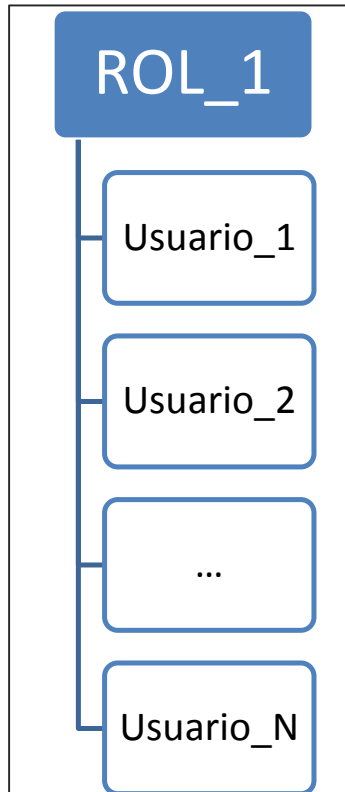


Figura 12.1 - Jerarquía de Rol y Usuario.

La sintaxis para la creación de usuarios en lenguaje SQL se muestra en la Figura 12.2.

```
CREATE USER nombre_usuario;
```

Figura 12.2 - Sintaxis para la creación de usuarios.

En estas sintaxis encontramos los siguientes componentes:

1. **CREATE USER**  
Palabras reservadas para definir un usuario o un rol.
2. **nombre\_usuario**  
Es el nombre del nuevo usuario o rol que queremos definir en la base de datos.

Esta sintaxis permite agregar opciones específicas para un usuario, algunas de estas son:

**1. SUPERUSER, NOSUPERUSER**

Otorga la característica de ser o no súper usuario. Un súper usuario tiene todos los permisos dentro de cualquier tabla de cualquier base del SMBD. Un ejemplo de superuser sería el usuario postgres.

**2. CREATEDB, NOCREATEDB**

Otorga el permiso para crear o no crear bases de datos en el SMBD.

**3. CREATEUSER, NOCREATEUSER**

Otorga el permiso para crear o no crear usuarios.

**4. PASSWORD**

Define una contraseña o clave de acceso para el usuario que se está definiendo.

**5. VALID UNTIL**

Define una fecha de caducidad del usuario, así como de los permisos que le fueron otorgados.

A modo de ejemplo, en la Figura 12.3, se muestra la sintaxis para definir un usuario llamado `nombre_usuario` con privilegios de súper usuario, con contraseña `clave_para_ingresar` y con caducidad en el año 2099, mes de julio (7) y el día 4.



```
CREATE USER nombre_usuario WITH  
SUPERUSER  
PASSWORD 'clave_para_ingresar'  
VALID UNTIL '2099-07-04';
```

Figura 12.3 - Ejemplo de creación de usuarios con opciones.

### **3.2.2. Permisos específicos**

Los SMBD permiten otorgar permisos específicos para las bases y las tablas que contenga, así como sobre las acciones que se pueden realizar en estas. Es decir, los permisos específicos limitarán las posibilidades de un usuario para operar dentro del SMBD.

Algunos de estos permisos son:

- SELECT: Consultas a tablas.
- INSERT: Inserción de nuevos registros
- UPDATE: Actualización o modificación de los registros existentes.
- DELETE: Eliminación de registros existentes.

Para poder definir estos permisos se usa lenguaje SQL. La Figura 12.4 muestra la sintaxis que permite realizar estas definiciones.

```
GRANT permiso_1, permiso_2,..., permiso_n  
ON tabla_1, tabla_2, ..., tabla_m  
TO usuario_1, usuario_2,..., usuario_p  
WITH GRANT OPTION;
```

Figura 12.4 - Sintaxis para otorgar permisos específicos.

En esta sintaxis encontramos:

1. **GRANT**  
Palabra reservada para comenzar a definir los permisos. Se traduce cómo conceder u otorgar.
2. *permiso\_1, permiso\_2, ..., permiso\_n*  
Es la lista de los permisos que se quieren otorgar.
3. **ON**  
Palabra reservada para indicar sobre cuales tablas o bases de datos serán válidos los permisos que se están definiendo. Si se quiere definir una base de datos completa se deberá de agregar la palabra reservada DATABASE.
4. *tabla\_1, tabla\_2, ..., tabla\_m*  
Es la lista de las tablas sobre las que serán válidos los permisos.
5. **TO**  
Palabra reservada para indicar a que usuario, usuarios o grupo de usuarios se les otorgará los permisos que se están definiendo.
6. *usuario\_1, usuario\_2, ..., usuario\_p*  
Es la lista de los usuarios a los que se le otorgarán los permisos.
7. **WITH GRANT OPTION**
8. Palabras reservadas para otorgar a la lista de usuarios definida en el punto 6, el privilegio de otorgar permisos. Es decir, si se escribe esta opción, los usuarios a los que se les otorgó, podrán ser capaces de conceder a otros usuarios, a lo más, todos los permisos con los que éstos cuentan. Esta parte de la sintaxis es opcional ya que si no se requiere que los usuarios otorguen permisos, simplemente se omitirá.

Así como se otorgan permisos, de la misma manera se pueden revocar. La sintaxis es similar a la mostrada anteriormente. En la Figura 12.5 se muestra la sintaxis para revocar permisos específicos.

```
REVOKE INSERT, UPDATE  
ON costo, detalle_de_orden  
FROM nombre_usuario, nombre_usuario1;
```

Figura 12.5 - Sintaxis para revocar permisos específicos.

En esta sintaxis encontramos la diferencia de REVOKE por GRANT además de FROM por TO y no podemos usar WITH GRANT OPTION. Estos cambios se deben a que estamos revocando los permisos, por ejemplo, en el caso de TO se usa para otorgar un permiso "para" algún usuario, y al cambiarlo por FROM para revocar un permiso "de" un usuario.

### 3.2.3. Acceso al SMBD con el perfil de un usuario creado con anterioridad

A continuación se muestra como acceder como un usuario que ha sido creado con anterioridad. Existen dos maneras: por la interfaz pgAdmin III o por consola.

- **pgAdminIII:** Iniciar la interfaz y desconectarse del servidor (si es que se encuentra conectado) oprimiendo el botón derecho sobre el servidor *PostgreSQL 9.2 (localhost:5432)* y seleccionar la opción **Disconnect server**. Ver Figura 12.6.

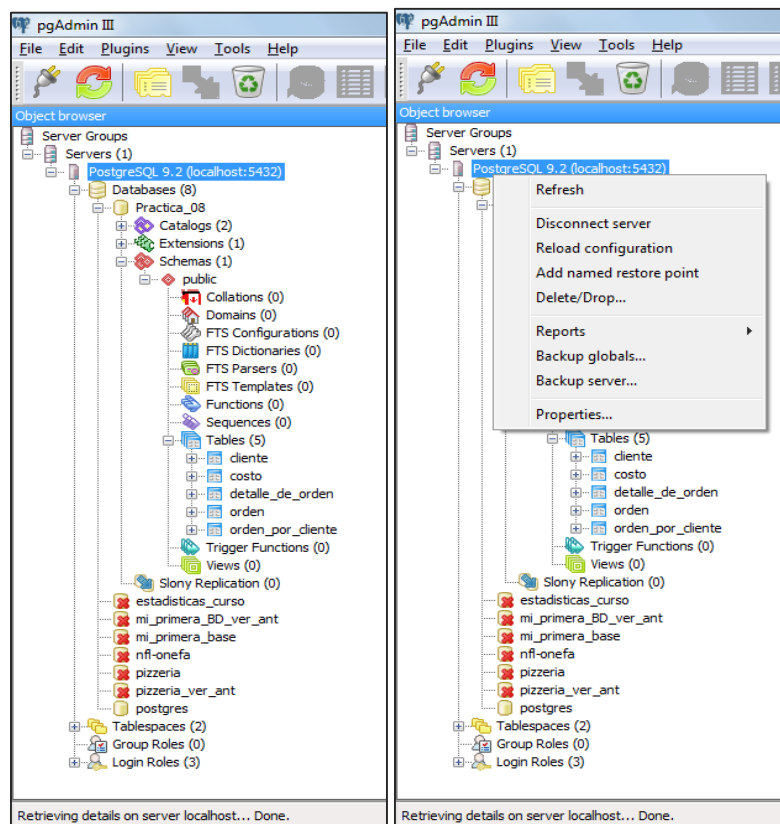


Figura 12.6 - Ingreso al SMBD con un usuario determinado (1).

Una vez desconectado, se deberá conectar cambiando el nombre de usuario. Esto se puede hacer desde la ventana de propiedades del servidor *PostgreSQL 9.2 (localhost:5432)* que se abre con el botón derecho como se realizó en la Figura 12.6.

Dentro de la ventana de propiedades se cambiará el nombre de usuario (*Username*) por el nombre de usuario que se creó anteriormente. La Figura 12.7 muestra el cambio de Username a *nombre\_usuario*, que fue creado en la Figura 12.3 de ésta práctica.

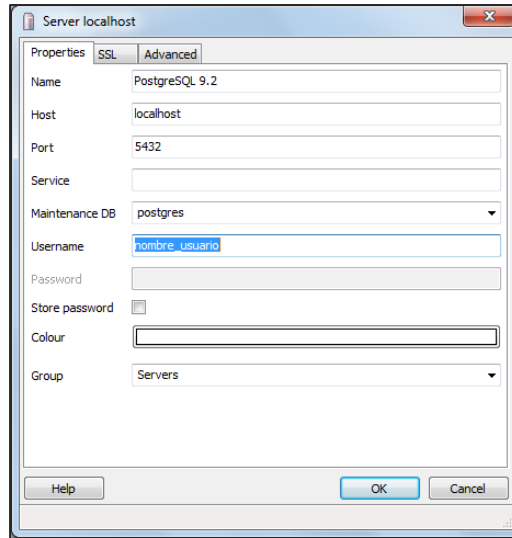


Figura 12.7 - Ingreso al SMBD con un usuario determinado (2).

Cuando se haya terminado de cambiar el nombre de usuario, se debe oprimir *OK* y conectarse al servidor "PostgreSQL 9.2 (localhost:5432)" normalmente. Se abrirá una ventana para solicitar la contraseña del nuevo usuario. Se puede corroborar el nombre del usuario en la información que exhibe ésta pantalla. Ver Figura 12.8.

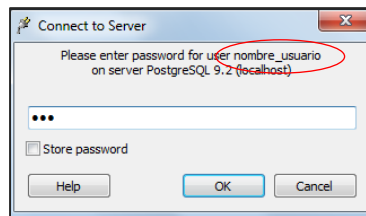


Figura 12.8 - Ingreso al SMBD con un usuario determinado (3).

De ésta manera el nuevo usuario estará conectado a la base de datos y los permisos que le fueron otorgados o denegados se verán reflejados cada vez que trate de realizar alguna acción. La Figura 12.9 muestra un ejemplo con el usuario *nombre\_usuario* al que no se le otorgaron permisos de *SELECT* en la tabla *Costo* y el SMBD le niega la petición.

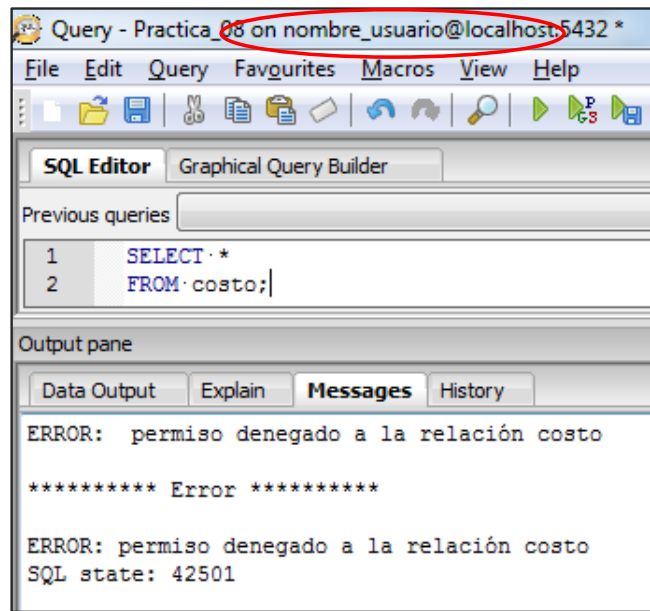


Figura 12.9 - SELECT interrumpido por permisos no otorgados.

- **Consola:** Iniciar la consola **SQL Shell (psql)** desde la carpeta del programa PostgreSQL 9.2, como se muestra en la Figura 12.10.

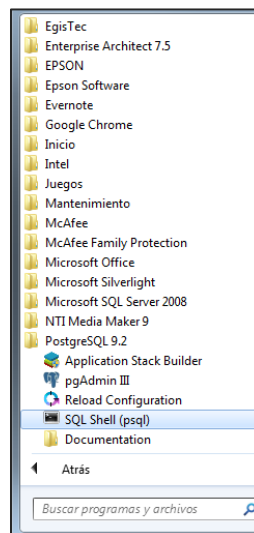
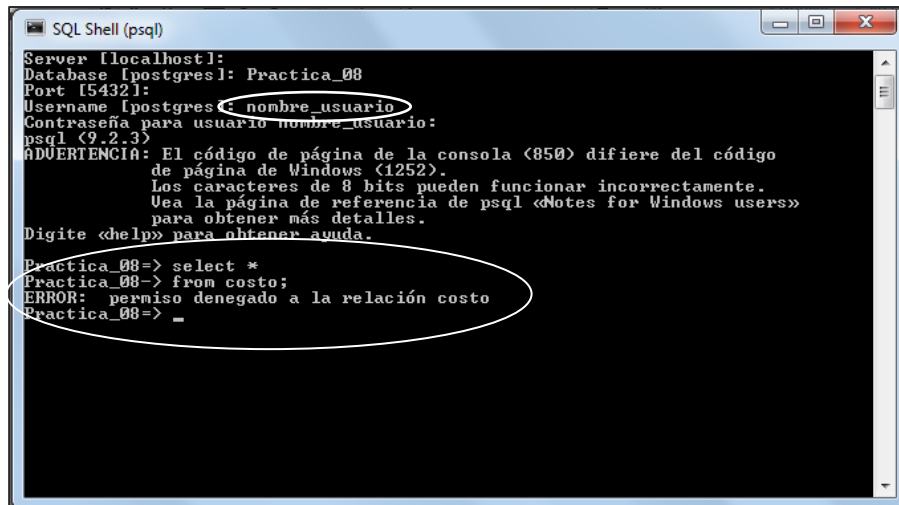


Figura 12.10 - Apertura de la consola de PostgreSQL en el menú de programas de inicio de Windows.

En la Figura 12.11 se muestra la consola de PostgreSQL ya con los datos de acceso reflejados. El nombre del servidor (por default localhost), la base de datos a la cual se quiere conectar (por default postgres), el puerto (por default 5432), el nombre de usuario (por default postgres) y por último el password (si es que fue declarado con anterioridad).





```
SQL Shell (psql)
Server [localhost]:
Database [postgres]: Practica_08
Port [5432]:
Username [postgres]: nombre_usuario
Contraseña para usuario nombre_usuario:
psql (9.2.3)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.
Practica_08=> select *
Practica_08-> from costo;
ERROR: permiso denegado a la relación costo
Practica_08=> _
```

Figura 12.11 - Ingreso a la base de datos Practica\_08 con el usuario “nombre\_usuario”.

Nótese que se ejecutó la misma consulta que en la Figura 12.9 obteniendo el mismo resultado.

### 3.3. Respaldo y Restauración

Es probable que aún con estas normas de seguridad activadas para el SMBD, existan problemas de pérdida o corrupción de los datos almacenados debido a múltiples factores tanto humanos como del entorno. Por ejemplo, una descarga eléctrica que afecte los servidores o el hurto físico de éstos mismos, entre otras.

Para mantener la mayor cantidad de información asegurada, los SMBD permiten respaldar la información almacenada en bases de datos, para su posterior restauración. Estos respaldos pueden contener la información de la base de datos completa o de tablas en particular. Dependiendo del SMBD que se utilice, se puede escoger también la opción para añadir la información inherente al esquema, es decir, la manera en la que está construida una tabla o la base de datos.

#### 3.3.1. Respaldo

La manera más sencilla que presenta PostgreSQL para crear respaldos se muestra a continuación. Para comenzar se debe de abrir pgAdminIII e ingresar normalmente. Ver Figura 12.12.

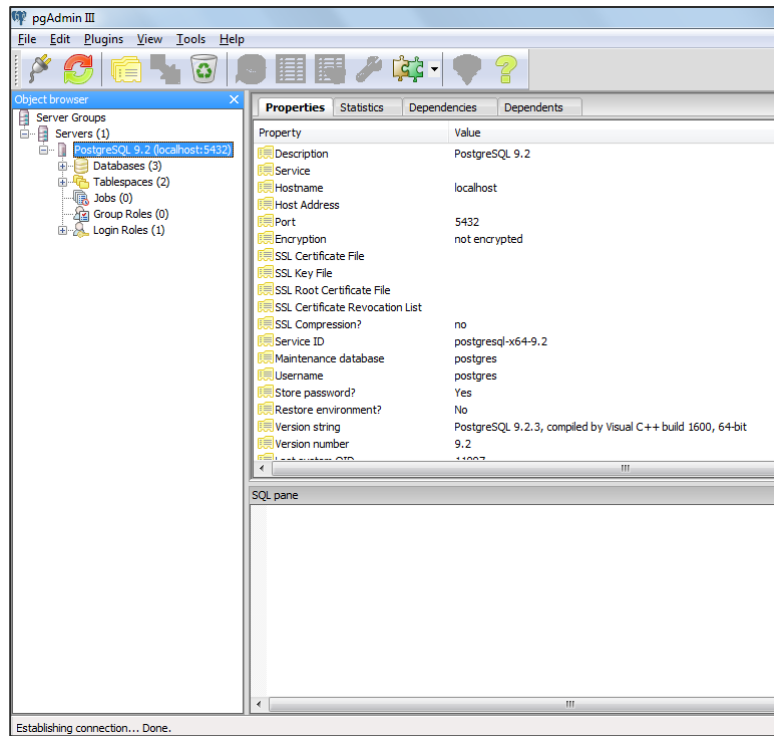


Figura 12.12 - Respaldo de bases de datos (1).

Para restaurar una base de datos completa o una tabla, se selecciona del listado *Databases* de PostgreSQL la opción que buscamos, es decir, el nombre de la base de datos a respaldar y se oprime el botón secundario del mouse. Para después seleccionar la opción *Backup* (respaldar). Si sólo se requiere respaldar una tabla, se debe seleccionar la base de datos a la cual pertenece y más adelante se seleccionará la tabla específica. Ver Figura 12.13.

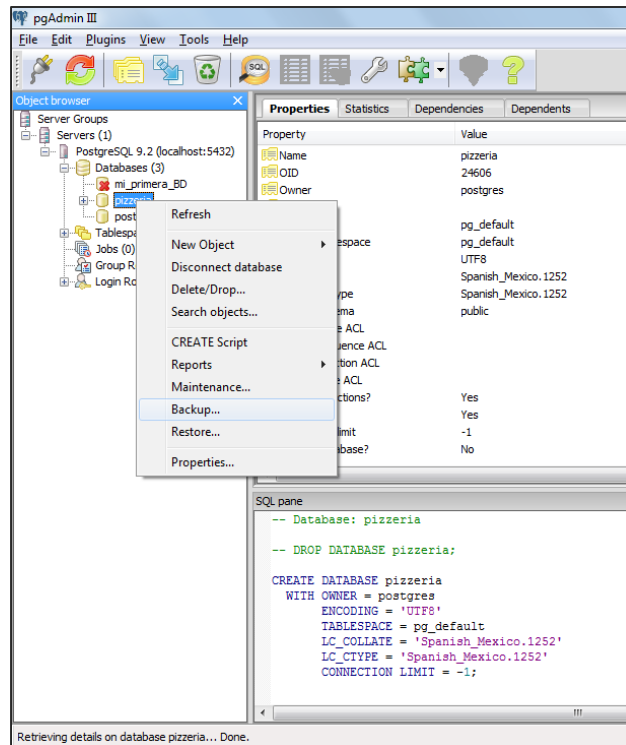


Figura 12.13 - Respaldo de bases de datos (2).

Una vez seleccionada la opción de *Backup*, el SMBD abrirá una ventana que permitirá definir las características específicas bajo las cuales se requiere hacer el respaldo. En el campo *Filename* se captura el nombre que tendrá el archivo que contendrá el respaldo, así como la ruta donde se guardará éste archivo. *Format* es una lista desplegable que nos da la opción de elegir el formato que tendrá el archivo que estamos generando, es común que se elija la opción *Custom*. Ver Figura 12.14.

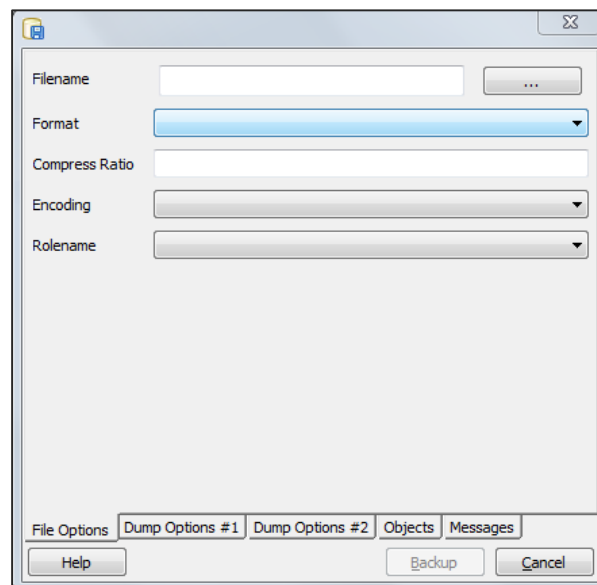


Figura 12.14 - Respaldo de bases de datos (3).

En la pestaña *Dump Options #1* se tiene la opción de escoger sólo el esquema o sólo los datos de la base o tabla que se pretenden respaldar. Si no se selecciona ninguna de estas opciones, el archivo se generará tanto con el esquema como con los datos. Ver Figura 12.15.

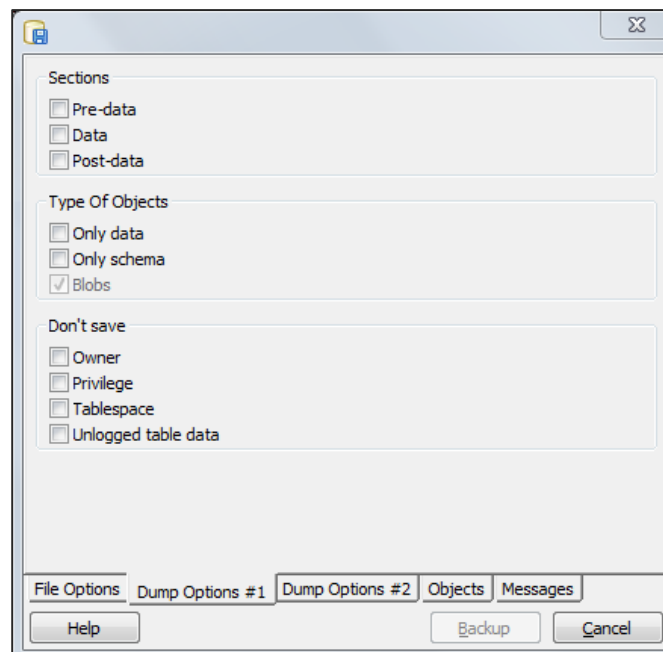


Figura 12.15. Respaldo de bases de datos (4).

Para la tercera pestaña *Dump Options #2* el SMBD ofrece la posibilidad de agregar al archivo las sintaxis para crear la base de datos (CREATE DATABASE) y para eliminarla (DROP DATABASE). Es común que estas sintaxis se usen cuando se pretende migrar la base de datos a otro servidor o computadora. Ver Figura 12.16.

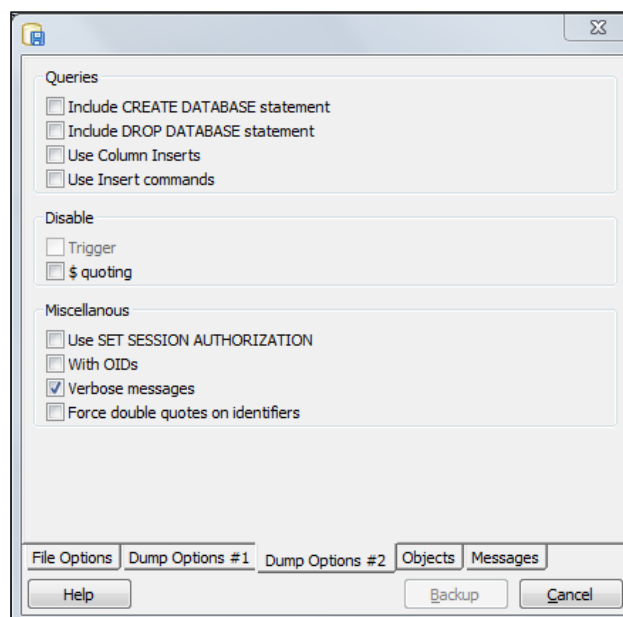


Figura 12.16 - Respaldo de bases de datos (5).

En la pestaña *Objects* podemos seleccionar las tablas que se desean respaldar, desde una hasta todas. Ver Figura 12.17.

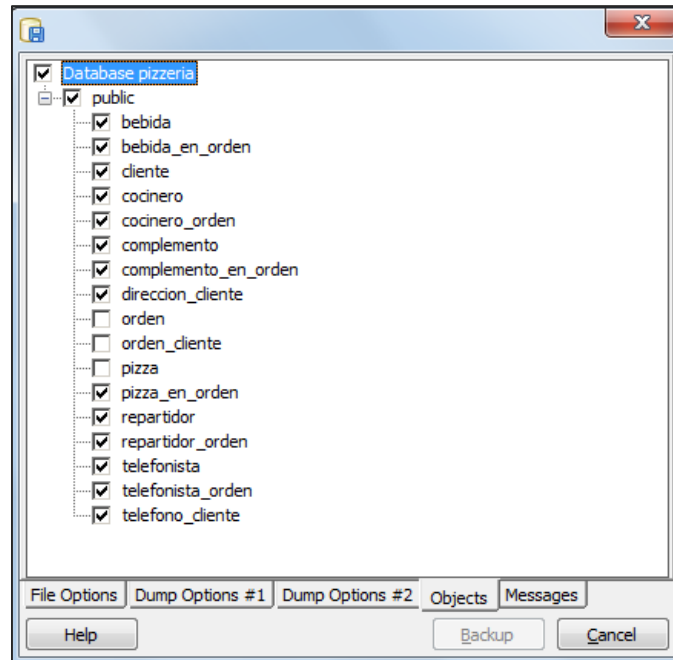


Figura 12.17 - Respaldo de bases de datos (6).

Cuando se haya finalizado la selección de las características, bastará con oprimir el botón *Backup* y el SMBD, en la pestaña *Messages*, regresará el estatus de la acción del respaldo. Ver Figura 12.18.

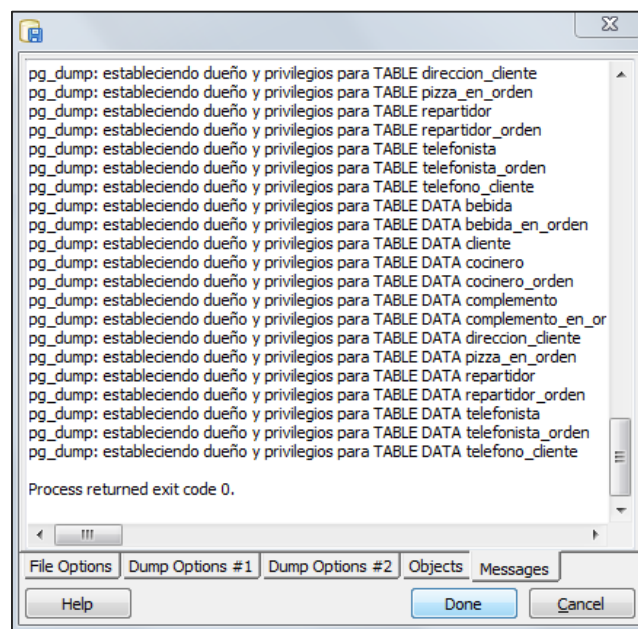


Figura 12.18 - Respaldo de bases de datos (7).

El comando *pg\_dump* que aparece en los mensajes de salida de la Figura 12.18 se utiliza para extraer una base de datos de PostgreSQL y almacenarla en un archivo. Esta opción se utiliza en la consola de PostgreSQL.

La última línea del proceso muestra el código *exit code 0*, lo que significa que el archivo de respaldo se creó de manera exitosa. Ver Figura 12.18.

Si se llegara a mostrar cualquier otro código, implicará que el archivo de respaldo se creó correctamente. Oprimiendo *Done* habrá finalizado la acción de respaldo.

### **3.3.2. Restauración**

Si en un determinado momento la base de datos o alguna de sus tablas sufrieran algún inconveniente, el SMBD puede llamar y ejecutar el archivo que se generó con la acción de respaldo del punto anterior.

Es necesario que la base de datos que se pretende restaurar se encuentre creada dentro de PostgreSQL para poderla asociar con el archivo de respaldo. En el caso que la base de datos haya sido eliminada, bastará con crearla nuevamente para restaurar los datos y tablas que se encuentren en el archivo de respaldo.

De la misma manera que en el respaldo, se deberá seleccionar del listado *Databases* la base que queremos restaurar y con el botón secundario hacer click, para escoger del menú la opción *Restore* (restaurar). Ver Figura 12.13.

Se deberá seleccionar el archivo de respaldo que se creó en la sección anterior dando click en el botón que contiene tres puntos que se encuentra a la derecha del cuadro de texto *Filename*. La ruta que tenemos que especificar es la misma ruta donde se guardó el archivo de respaldo. Después de seleccionar las características que se requieren para la restauración de manera similar al respaldo oprimir el botón *Restore*. Ver Figura 12.19.

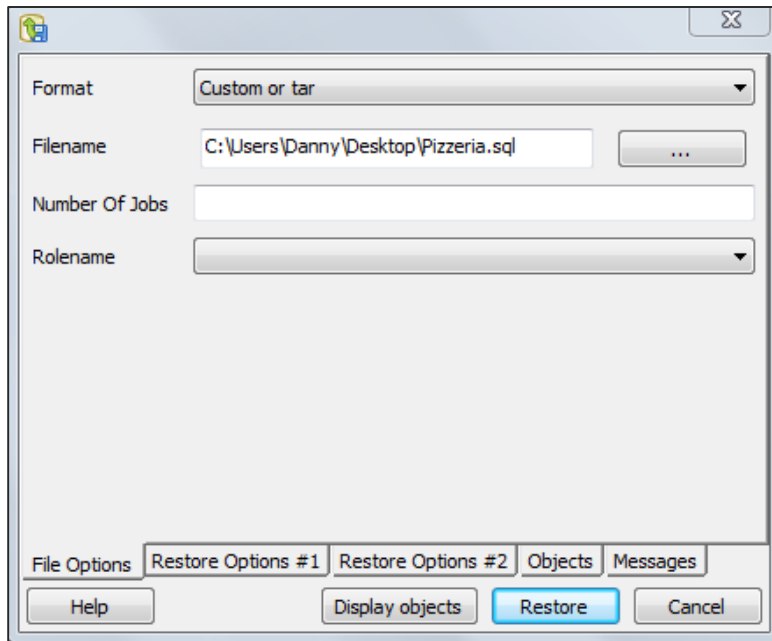


Figura 12.19 - Restauración de bases de datos (1).

PostgreSQL abrirá el archivo y procederá a ejecutar el código que se encuentra en éste. El código que presenta este archivo está escrito en DDL (Data Definition Language).

De ésta manera se habrá restaurado la base de datos que se seleccionó. El SMDB regresará una lista de mensajes en la pestaña *Messages* con el status de la restauración. Ver Figura 12.20.

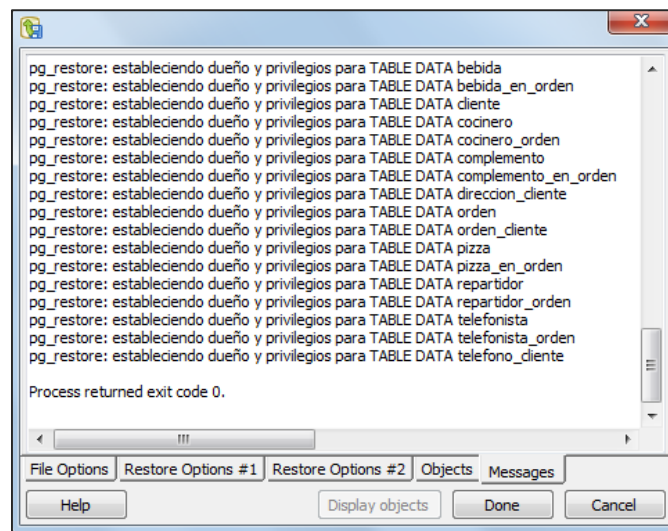


Figura 12.20 - Restauración de bases de datos (2).

La última línea del proceso muestra el código *exit code 0*, que significa que la restauración se efectuó de manera exitosa. Ver Figura 12.20.

Si se llegara a mostrarse cualquier otro código, implicará que la restauración no fue llevada a cabo correctamente. Oprimiendo el botón *Done* finalizará la acción de restauración.

#### 4. Ejercicios

*(NOTA: Resuelve los siguientes ejercicios en relación al proyecto que realizarás durante el curso, en dado caso que no tengas un proyecto, utiliza la información en el apéndice NFL-ONEFA parte 12 al final de esta práctica para realizarlos)*

1. Crea tres distintos usuarios con la siguiente combinación de permisos específicos sobre tablas distintas de tu base de datos y entrega la sintaxis que usaste.
  - MIGMOR: Otorga SELECT y DELETE.
  - ATD: Otorga INSERT y SELECT.
  - DOR: Otorga UPDATE, INSERT y DELETE.
2. Retira los permisos que se indican a cada uno de los tres usuarios que definiste en el punto 1 y entrega la sintaxis que usaste.
  - MIGMOR: Revoca DELETE.
  - ATD: Revoca INSERT.
  - DOR Revoca UPDATE e INSERT.
3. Crea un respaldo de toda tu base de datos y entrega el archivo que se generó.
4. Intencionalmente elimina 5 tablas de tu base de datos.
5. Restaura las tablas que eliminaste en el punto 4.
6. Intencionalmente elimina solo los registros de otras dos tablas de tu base de datos.
7. Restaura toda la base de datos mediante el archivo que generaste en el punto 3.
8. Valida que todos los datos y esquemas que eliminaste se hayan restaurado exitosamente.

#### Entregables requeridos:

- Usuarios creados cada uno con los permisos adecuados
- Respaldo completo de la base de datos



## 5. Apéndice NFL-ONEFA parte 12

Realiza los ejercicios de la sección 4 de ésta práctica sobre la base de datos NFL-ONEFA.

HINT: Como ayuda para la corrección de posibles errores, a continuación se presenta las sintaxis con las cual se podrían resolver los ejercicios presentados en la sección 4.

### Ejercicio 1.

```
CREATE USER MIGMOR;  
GRANT SELECT, DELETE  
ON Jugador, jugador_equipo, Equipo  
TO MIGMOR;
```

```
CREATE USER ATD;  
GRANT INSERT, SELECT  
ON Estadio, Ciudad  
TO ATD;
```

```
CREATE USER DOR;  
GRANT UPDATE, INSERT, DELETE  
ON Entrenador, equipo_entrenador, Equipo  
TO DOR;
```

### Ejercicio 2.

```
REVOKE DELETE  
ON Jugador, jugador_equipo, Equipo  
FROM MIGMOR;
```

```
REVOKE INSERT  
ON Estadio, Ciudad  
FROM ATD;
```

```
REVOKE UPDATE, INSERT  
ON Entrenador, equipo_entrenador, Equipo  
FROM DOR;
```