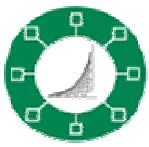
Laboratorio de Informática



MANUAL BÁSICO DE POSTGRESQL

ESCUELA COLOMBIANA DE INGENIERÍA
JULIO GARAVITO
LABORATORIO DE INFORMÁTICA
BOGOTÁ D. C.
2007-2

TABLA DE CONTENIDO

INTRODUCCIÓN

En este manual se encuentras las funciones básicas de PostgreSQL, como la autentificación ante PostgreSQL, el manejo de contraseñas, el manejo de usuarios (creación, modificación y eliminación) y sus privilegios, el manejo de grupos, (creación, eliminación), algunas otras funciones de PostgreSQL, como lo son los triggers, la creación, modificación y eliminación de tablas, creación y eliminación de bases de datos, los tipos de datos y los tipos numéricos.

Para cada sentencia o comando se muestra su sintaxis, y se parte del hecho de que la persona que haga uso de este manual, tiene conocimientos básicos, sobre el lenguaje SQL.

1. AUTENTICACIÓN EN POSTGRESQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales. Comenzó como un proyecto denominado *Ingres* en la Universidad Berkeley de California. *Ingres* fue desarrollado comercialmente más tarde por la Relational Technologies / Ingres Corporation.

A partir de PostgreSQL 7.1.x, los accesos de clientes basados en máquina (host) se encuentran especificados en el archivo *pg_hba.conf*. El archivo *pg_hba.conf* le permite establecer el tipo de autenticación basasda en máquina a ser usada. Esta autenticación es realizada antes de que PostgreSQL estblezca una conexión a la base de datos en cuestión, donde los permisis de usuarios serían relevantes.

El archivo *pg_hba.conf* está localizado en el directorio de datos de PostgreSQL (p.ej., */usr/local/pgsql/data/*), y es instalado automáticamente con la ejecución del comando *initdb* cuando PostgreSQL es instalado.

1.1 Autenticación de Contraseña

Las contraseñas de usuario son almacenadas en un texto plano en la tabla de sistema *pg_shadow*, pero sólo los superusuarios de PostgreSQL tienen permiso para ver la tabla *pg_shadow* y esta tabla además es accesible desde cualquier base de datos.

La estructura de la tabla es:

Columna	Tipo
usename	name
usesysid	integer
usecreatedb	boolean
usetrace	boolean
usesuper	boolean
usecatupd	boolean
passwd	text
valuntil	abstime

En dado caso que la contraseña no sea definida, por defecto el sistema asignara NULL.

2. ADMINISTRACIÓN DE POSTGRESQL

PostgreSQL almacena los datos de usuarios así como también los datos de los grupos dentro de sus propios catálogos de sistema. De esta manera, cualquier conexión a PostgreSQL debe ser realizada con un usuario específico, y cualquier usuario puede pertenecer a uno o más grupos definidos.

La tabla de usuarios en PostgreSQL controla los permisos de acceso y quién está autorizado a realizar acciones en el sistema, al igual qué las acciones puede realizar. Los grupos existen como un mecanismo para simplificar la ubicación de estos permisos. Tanto las tablas de usuarios como de grupos existen como objetos globales de base de datos, por consiguiente no están agregadas a ninguna base de datos en particular.

2.1 Gestión de Usuarios

Cada usuario tiene un ID de sistema interno en PostgreSQL (llamado *sysid*), así como una contraseña. El ID es utilizado para asociar objetos en una base de datos con su propietario

PostgreSQL crea por defecto a un *superusuario* llamado ***postgres***. Todos los demás superusuarios pueden ser creados por éste, o por cualquier otro *superusuario* creado posteriormente.

PostgreSQL proporciona dos métodos para la creación de usuarios de bases de datos. Cada uno de ellos requiere autenticación como superusuario.

Los métodos son:

- A través del uso del comando SQL CREATE USER.
- Un programa de línea de comandos llamado *createuser*

2.1.1 CREATE USER

La sintaxis para CREATE USER es:

```
CREATE USER nombre_usuario
[ WITH [ SYSID uid ]
     [ PASSWORD 'password' ] ]
[ CREATEDB | NOCREATEDB ]
[ CREATEUSER | NOCREATEUSER ]
[ IN GROUP groupname [, ...] ]
[ VALID UNTIL 'abstime' ]
```

A continuación se describe cada una de las partes de la sintaxis de CREATE USER:

- **SYSID uid**

  Especifica que el ID que va a definirse debe establecerse al valor de *uid*. Si se omite, un razonable y único valor numérico por defecto es escogido.

- **PASSWORD 'password'**

  Establece la nueva contraseña del usuario. Si no se especifica, la contraseña por defecto es NULL.

- **CREATEDB | NOCREATEDB**

  Usando la palabra clave CREATEDB se le garantiza al nuevo usuario el privilegio de crear nuevas bases de datos, así como el de destruir las de su propiedad. Usando NOCREATEDB se deniega este permiso (que es lo que ocurre por defecto).

- **CREATEUSER | NOCREATEUSER**

  Certifica el privilegio de crear nuevos usuarios. Si un usuario tiene los privilegios de crear a otros usuarios tendrá además todos los privilegios, en todas las bases de datos (incluyendo los permisos para crear una base de datos, aunque se haya especificado NOCREATEDB). NOCREATEUSER explícitamente fuerza a la situación por defecto, que deniega el privilegio.

- **IN GROUP nombre_grupo [, ...]**

  Añade al nuevo usuario al grupo llamado *nombre_grupo*. Pueden ser especificados múltiples nombres de grupo, separándolos mediante comas. El o los grupos deben existir para que funcione la condición.

- **VALID UNTIL 'abstime'**

  Establece que la contraseña del usuario expirará el *abstime*, el cual debe ser un formato reconocible de fecha/hora (timestamp). Tras esa fecha, la contraseña se resetea, y la expiración se hace efectiva.

- **VALID UNTIL 'infinity'**

  Establece validez permanente para la contraseña del usuario.

2.1.2 createuser

El script *createuser* es ejecutado directamente desde la línea de comandos, y puede operar de dos formas.

Si se utiliza sin argumentos, él interactivamente le pedirá el nombre de usuario y cada uno de los privilegios que se le van a asignar. Alternamente, puede optar por especificar las opciones y el nombre del usuario a ser creado en la misma línea de comandos.

La sintaxis de *createuser* es:

createuser [opciones] [nombre_usuario]

El *nombre_usuario* en la sintaxis representa el nombre del usuario que va a crear. Reemplace *opciones* con una o más de las siguientes:

- ***-d, -createdb*** Equivalente a la palabra clave CREATEDB. Permite al nuevo usuario crear bases de datos.

- ***-D, -no-createdb*** Equivalente a la palabra clave NOCREATEDB Explícitamente indica que el nuevo usuario no puede crear bases de datos.

- ***-a, -adduser*** Equivalente a la palabra clave CREATEUSER Perimte al nuevo usuario la creación de otros usuarios, y asigna el status de superusurario al usuario.

- ***-A, -no-adduser*** Equivalente a la palabra clave NOCREATEUSER. Explícitamente indica que el nuevo usuario no es superusuario.

- ***-i SYSID, -sysid=SYSID*** Establece el nuevo ID de sistema del usuario a SYSID.

- ***-P, -pwprompt*** Resulta en una petición de introducción de contraseña, permitiéndole establecer la contraseña del nuevo usuario.

- ***-h NOMBRE_MAQUINA, -host=NOMBRE_MAQUINA*** Especifica desde qué NOMBRE_MAQUINA se conectará, además de la local (localhost), o la máquina definida por la variable de entorno PGHOST.

- ***-p PUERTO, -port=PUERTO*** Especifica que la conexión de base de datos se realizará por el puerto PUERTO, en vez de por el puerto por defecto.

- ***-U NOMBRE_USUARIO, -username=NOMBRE_USUARIO*** Especifica que NOMBRE_USUARIO será el usuario que conecte a PostgreSQL (por defecto se conecta usando el nombre de usuario del sistema).

- ***-W, -password*** Resulta en una petición de contraseña para el usuario que conecta, lo cual ocurre automáticamente si el archivo *pg_hba.conf* está configurado para no confiar en la máquina solicitante.

2.2 Modificación de Usuarios

Los usuarios existentes sólo pueden ser modificados por superusuarios PostgreSQL mediante el comando SQL ALTER USER, donde su sintaxis es:

```
ALTER USER nombre_usuario
        [ WITH PASSWORD 'password' ]
        [ CREATEDB | NOCREATEDB ]
        [ CREATEUSER | NOCREATEUSER ]
        [ VALID UNTIL 'abstime' ]
```

Aquí nuevamente se hace uso de las palabras claves utilizadas en CREATE USER.

2.3 Eliminación de Usuarios

Al igual que en la creación de usuarios, en la eliminación de usuarios también existen dos formas de hacerlo:

- El comando DROP USER
- El programa *psql*

La sintaxis para DROP USER es:

```
DROP USER nombre_usuario
```

La sintaxis para *psql* es:

```
[~]$ psql -U manager template1
Welcome to psql, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
     \h for help with SQL commands
     \? for help on internal slash commands
     \g or terminate with semicolon to execute query
     \q to quit
```

template1=# DROP USER salesuser;
DROP USER

2.4 Manejo de Grupos

Cualquier superusuario puede crear un nuevo grupo en PostgreSQL con el comando CREATE GROUP. Aquí tiene la sintaxis de CREATE GROUP:

CREATE GROUP nombre_grupo
[WITH [SYSID groupid]
[USER username [, ...]]]

En la sintaxis, *nombre_grupo* es el nombre del grupo a ser creado. debe iniciar por una carácter alfabético, y no puede exeder los 31 caracteres de longitud. El proporcionar la palabra clave WITH permite especificar cualquiera de los atributos opcionales. Si desea especificar el ID de sistema a usar con el nuevo grupo, utilice la palabra clave SYSID para especificar el valor de *groupid*. Use la palabra clave USER para incluir a uno o más usuarios al grupo en tiempo de creación. Separe los distintos usuarios mediante comas.

Adicionalmente, el usuario PostgreSQL y las tablas de grupos operan separadamente las unas de las otras. Esta separación que los ID de usuarios y grupos puedan ser idénticos dentro del sistema PostgreSQL.

Y para la eliminación de un grupo se usa el comando SQL DROP GROUP. Su sintaxis es:

DROP GROUP nombre_grupo

Para añadir o eliminar un usuario a un grupo se usa el comando SQL ALTER GROUP, especificando si es para añadirlo ADD y si es para eliminarlo DROP, y sin son varios usuarios simplemente se separan por ‘,’.

Su sintaxis es:

ALTER GROUP nombre_grupo { ADD | DROP } USER username [, ...]

3. OTRAS FUNCIONES BÁSICAS DE PostgreSQL

Estas son algunas otras funciones a tener en cuenta para el manejo de bases de datos en PostgreSQL.

- **Abort:** Aborta la transacción en curso

- **Modificar Grupo**:

  MODIFICAR GRUPO *nombre* AÑADIR USUARIO *nombre de usuario* [, ...]MODIFICAR GRUPO *nombre* ELIMINAR USUARIO *nombre de usuario* [, ...]

- **CREATE TABLE:** Crea una nueva tabla

  CREATE [TEMPORARY | TEMP] TABLE table (
  column type
  [NULL | NOT NULL] [UNIQUE] [DEFAULT value]
  [column_constraint_clause | PRIMARY KEY } [...]]
  [, ...]
  [, PRIMARY KEY (column [, ...])]
  [, CHECK (condition)]
  [, table_constraint_clause]

) [INHERITS (inherited_table [, ...])]

- **Modificar Tabla:**

  MODIFICAR TABLA *tabla* [*]
  AÑADIR [COLUMNA] *columna tipo*
  MODIFICAR TABLA *tabla* [*]
  MODIFICAR [COLUMNA] *columna* { SET DEFAULT *valor* | DROP DEFAULT }
  MODIFICAR TABLA *tabla* [*]
  RENOMBRAR [COLUMNA] *columna* A *nueva columna*
  MODIFICAR TABLA *tabla*
  RENOMBRAR A *nueva tabla*

- **Modificar usurario:**

  MODIFICAR USUARIO *nombre de usuario*[WITH PASSWORD '*palabra clave*'][CREATEDB | NOCREATEDB][CREATEUSER | NOCREATEUSER][VALID UNTIL '*abstime*']

- **BEGIN:** comienza una transacción en modo encadenado

BEGIN [WORK | TRANSACTION]

- **CLUSTER:**

  CLUSTER *indexname* ON *table*

- **COMMIT:** Realiza la transacción actual.

  COMMIT [WORK | TRANSACTION]

- **COPY:** Copia datos entre ficheros y tablas.

  COPY [BINARY] *table* [WITH OIDS]FROM { '*filename*' | stdin }[[USING] DELIMITERS '*delimiter*'][WITH NULL AS '*null string*']COPY [BINARY] *table* [WITH OIDS]TO { '*filename*' | stdout }[[USING] DELIMITERS '*delimiter*'][WITH NULL AS '*null string*']

- **CREATE AGGREGATE:** Define una nueva función de agregado

  CREATE AGGREGATE *name* [AS] (BASETYPE = *data_type* [, SFUNC1 = *sfunc1*, STYPE1 = *sfunc1_return_type*][, SFUNC2 = *sfunc2*, STYPE2 = *sfunc2_return_type*][, FINALFUNC = *ffunc*][, INITCOND1 = *initial_condition1*][, INITCOND2 = *initial_condition2*])

- **CREATE DATABASE:** Crea una nueva base de datos.

  CREATE DATABASE name [WITH LOCATION = 'dbpath']

- **CREATE FUNCTION:** Crea una nueva función.

  CREATE FUNCTION name ([ftype [, ...]])
  RETURNS rtype
  [WITH (attribute [, ...])]
  AS obj_file , link_symbol

  LANGUAGE 'C'

- **CREATE INDEX:** Crear un índice secundario.

  CREATE [UNIQUE] INDEX nombre_indice ON tabla
  [USING nombre_acceso] (columna [nombre_operador] [, ...])

  CREATE [UNIQUE] INDEX nombre_indice ON tabla

[USING nombre_acceso] (nombre_funcion(r">columnale> [, ...]) nombre_operador)

- **CREATE TRIGGER:** Crea un nuevo disparador

  CREATE TRIGGER name
  { BEFORE | AFTER } { event
  [OR ...] } ON table
  FOR EACH { ROW | STATEMENT } EXECUTE PROCEDURE
  ER">funcBLE>
  (arguments)

- **DELETE:** Borrar filas de una tabla

  DELETE FROM table [WHERE condition]

- **DROP AGGREGATE:** Elimina la definición de una función agregada

  DROP AGGREGATE name type

- **DROP DATABASE:** Elimina una base de datos existente

  DROP DATABASE name

- **DROP TABLE:** Eliminar tablas de una base de datos

  DROP TABLE nombre [, ...]

- **DROP TRIGGER:** Eliminar la definición de un disparador

  DROP TRIGGER nombre ON tabla

4. TIPOS DE DATOS PostgreSQL

Estos son algunos de los tipos de datos que hay en PostgreSQL.

Postgres Type	Description
bool	logical boolean (true/false)
box	rectangular box in 2D plane
char(n)	fixed-length character string
cidr	IP version 4 network or host address
circle	circle in 2D plane
date	calendar date without time of day
decimal	exact numeric for p <= 9, s = 0
float4/8	floating-point number with precision p
float8	double-precision floating-point number
inet	IP version 4 network or host address
int2 int4	signed two-byte integer signed 4-byte integer
int8	signed 8-byte integer
line	infinite line in 2D plane
lseg	line segment in 2D plane
money	US-style currency
numeric	exact numeric for p == 9, s = 0
path	open and closed geometric path in 2D plane
point	geometric point in 2D plane
polygon	closed geometric path in 2D plane
serial	unique id for indexing and cross-reference
time	time of day
timespan	general-use time span
timestamp	date/time
varchar(n)	variable-length character string

5. TIPOS NUMÉRICOS PARA PostgreSQL

Numeric Type	Storage	Description	Range
decimal	variable	User-specified precision	no limit
float4	4 bytes	Variable-precision	6 decimal places
float8	8 bytes	Variable-precision	15 decimal places
int2	2 bytes	Fixed-precision	-32768 to −32767
int4	4 bytes	Usual choice for fixed-precision	-2147483648 to +2147483647
int8	8 bytes	Very large range fixed-precision	+/- > 18 decimal places
numeric	variable	User-specified precision	no limit
serial	4 bytes	Identifer or cross-reference	0 to +2147483647