PostgreSQL

Taller # 1

"Uso y Manejo del Sistema Gestor de Bases de Datos PostgreSQL"

Luis García
Computación

ldgarc@gmail.com

PostgreSQL

Contenido

- Conexión
- PostgreSQL
- Creación de Tablas (*Create Table...*)
- Reglas y Restricciones (*Check – Foreign Keys)*
- Inserción de Datos (*Insert Into...*)
- Actualización de Datos (*Update...*)
- Selección de Datos (*Select...*)
- Eliminación de Datos (*Delete From...*)
- Creación de Dominios (*Create Domain...*)
- Creación de Vistas (*Create View...*)
- Creación de Triggers (*Create Trigger...*)
- Eliminación (*Drop [Table – Domain – View – Trigger]...*)

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Conexión

- Se debe realizar a partir de algún programa que implemente el protocolo SSH (Secure Shell), por medio del cual es posible acceder a máquinas remotas a través de una red.

- Bajo el Sist. Operativo Windows, la conexión se puede llevar a cabo con el programa PUTTY.

- Bajo otros sistemas, como LINUX, sólo se deben instalar los paquetes correspondientes al protocolo SSH, los cuales permitan acceder al servidor de la Base de Datos.

NOTA: también se puede usar el sistema de manera local en ambos Sist. Operativos

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Conexión
[Putty]

1. En el menú de configuración seleccione la categoría Session.

2. Introduzca el nombre del dominio o IP en el campo HostName. *En nuestro caso es 150.186.36.174*

3. Seleccione el protocolo SSH.

4. Haga clic en Open

PostgreSQL

Conexión

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

5. Ahora aparecerá una consola solicitando el nombre de usuario, colocamos **estudiante** y como Password **facyt**, se verá lo siguiente:

6. Ahora para conectarse a la Base de Datos hacer:

psql -U usuarioNro -d bdNro

El Nro dependerá del número del grupo, y cuando le sea solicitado el password, este será **usuarioNro**.

NOTA: Si accede al servidor desde LINUX, llevar a cabo los pasos a partir del nro. 5.

PostgreSQL

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

- Sist. Manejador de Bases de Datos Objeto-Relacional (ORDBMS) desarrollado inicialmente en la Univ. de California, mantenido actualmente por el "PostgreSQL Global Development Group" bajo la ideología de Software Libre.

- Está basado en el estándar SQL99.

- Entre sus características más importantes están:

  - Manejo de Clases y Herencia
  - Funciones definidas por el Usuario
  - Disparadores (Triggers)
  - Mecanismos de Integridad Transaccional

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
  - Sintáxis
  - Tipos de Datos
  - Def. de Claves
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Creación de Tablas
CREATE TABLE

- El lenguaje SQL incluye un conjunto de comandos para definición de datos. A continuación se presentan los principales:

```
CREATE TABLE   nombre_tabla   // Tabla a crear
               ( atr_1 tipo_atr_1,
                 atr_2 tipo_atr_2, ...,
                 atr_k1 tipo_atr_k ) ; // Atributo de la tabla – tipo
```

- Ejemplo:

```
CREATE TABLE persona ( CI int4, nombre varchar (30), edad int2 );
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
  - Sintáxis
  - Tipos de Datos
  - Def. de Claves
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Creación de Tablas
CREATE TABLE

- Tipos de Datos: los principales tipos de datos usados en el lenguaje SQL son:

Tipo	Descripción
Int2	Entero de 2 bytes con signo
Int4	Entero de 4 bytes con signo
Char (n)-	Cadena de Caracteres de longitud n fija
Varchar (n)	Cadena de Caracteres de longitud n variable
Bool	Tipo Booleano
Date	Tipo Fecha, año/mes/día (2005/12/15)
Float	Número de Punto Flotante

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
  - Sintáxis
  - Tipos de Datos
  - Def. de Claves
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Creación de Tablas

CREATE TABLE

- **Definición de Claves:** existen distintas formas de definir claves:

1. CREATE TABLE persona (CI int4 **primary key**, nombre varchar (30));

2. CREATE TABLE persona (CI int4, nombre varchar (30), edad int2, **primary key** (CI));

3. CREATE TABLE ciudad (nombre_c varchar(30), pais varchar (30), continente varchar(20), habitantes int8, **primary key** (nombre_c, pais));

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
  - Foreign Keys
  - Constraints
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Reglas y Restricciones

Foreign Keys

- **Foreign Keys:** o claves foráneas, pueden ser definidas en PostgreSQL, de manera tal que se respeten y mantengan las referencias adecuadas.

Ejemplos:

1.CREATE TABLE persona (CI int4 **primary key**, nombre varchar (30));

2.CREATE TABLE dpto (Id char(5) **primary key**, Id_sup int4, **foreign key** (id_sup) *references* persona (CI));

3.CREATE TABLE obrero (CI int4 **primary key**, dpto char(5), **foreign key** (CI) *references* persona (CI), **foreign key** (dpto) *references* dpto (Id));

PostgreSQL

Reglas y Restricciones

Foreign Keys

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
  - Foreign Keys
  - Constraints
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

- *Integridad Referencial:* es importante mantener la integridad de las referencias en caso de que se presente la eliminación o actualización, esto de la siguiente manera.

Ejemplos:

```
1.CREATE TABLE persona ( CI int4 primary key, nombre varchar (30));

2.CREATE TABLE  dpto (  Id char(5) primary key,  Id_sup int4,
                        foreign key (Id_sup) references
                        persona (CI) on delete cascade
                                     on update cascade);
```

PostgreSQL

Reglas y Restricciones

Constraints

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
  - Foreign Keys
  - Constraints
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

- *Check:* es el *constraint* más utilizado. Permite validar que el valor de cierta columna de la tabla cumpla con una condición *booleana*. Ejemplos:

```
1.CREATE TABLE persona ( CI int4 primary key, nombre varchar (30)
                         edad int2, check (edad >0 and edad <100);

2.CREATE TABLE persona ( CI int4 primary key, nombre varchar (30)
                         edad int2, constraint edad_valida
                         check ( edad >0 and edad <100 );

3.CREATE TABLE persona ( CI int4 primary key, nombre varchar (30)
                         edad int2,
                         check ( edad >0 and edad <100 ) ,
                         check ( CI>350000 and CI <95000000) );
```

PostgreSQL

Reglas y Restricciones
Constraints

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
  - Foreign Keys
  - Constraints
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

- *Not Null:* otro *constraint* muy utilizado. Permite validar que el valor de cierta columna debe presentarse siempre en los valores a insertar.

Ejemplo:

```
CREATE TABLE persona ( CI int4 primary key,
                       nombre varchar (30) NOT NULL,
                       edad int2 NOT NULL,
                       check (edad >0 and
                              edad <100) );
```

PostgreSQL

Reglas y Restricciones
Constraints

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
  - Foreign Keys
  - Constraints
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

- *Unique:* indica que el valor de una o varias columnas de cada tupla tiene que ser distinto al valor de todas las demás tuplas almacenadas en esa tabla

Ejemplo:

```
CREATE TABLE usuario ( CI int4 primary key,
                       nombre varchar (30) NOT NULL,
                       edad int2 NOT NULL,
                       login char (5) UNIQUE,
                       passwd varchar(10),
                       check (edad >0 and
                              edad <100) );
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Inserción de Datos

INSERT

- La sentencia INSERT lleva a cabo la inserción de una tupla dentro de una tabla.

- Los principales elementos de esta sentencia son usados en la siguiente sintaxis:

```
INSERT INTO   nombre_tabla    // Tabla en la cual se insertará
              ( atr_1, atr_2, ..., atr_k )  // Atributos referenciados
VALUES        ( valor_atr1, valor_atr2, ..., valor_atr_k) ;
```

Ejemplo:

```
INSERT INTO persona (CI, nombre, edad)
VALUES              (3861859, 'Pedro', 22) ;
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Inserción de Datos

INSERT

- Existen además distintas maneras de insertar una nueva tupla en una tabla.

- Si se desea crear una tupla con valores para cada atributo de la tabla, se puede realizar la inserción de la siguiente manera (tomando el ejemplo anterior):

1. INSERT INTO persona VALUES (3861859, 'Pedro', 22) ; // Correcto

2. INSERT INTO persona VALUES ('Pedro', 22, 3861859) ; // Errado

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Inserción de Datos

INSERT

- En el caso en el que no se vayan a denotar valores para todos los atributos de la tupla a insertar, se deben definir explícitamente cuales son los valores a insertar para qué atributos.

```
1. INSERT INTO persona (CI, Nombre)
                VALUES (3291859, 'Pedro') ; // Está bien hecho

2. INSERT INTO persona (CI, Nombre)
                VALUES ('Pedro', 3291859) ; // Está errado
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Actualización de Datos

UPDATE

- La sentencia *UPDATE* modifica una o más tuplas dentro de una tabla.

- En forma resumida, la sintaxis definida para esto es:

```
UPDATE  nombre_tabla          // Actualizar la tabla nombre_tabla
SET     atr_1 = valor_atr1, …, atr_k = valor_atrk  // Valores a modificar
WHERE   condicion_x ;         //  En las tuplas que cumplan con X
```

- Ejemplo:

```
UPDATE persona
SET     nombre = 'Johan',  edad = 25
WHERE   CI = 3291859;
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Selección de Datos

SELECT

- Es una de las sentencias principales, ya que a partir de ella se recuperan (o muestran) los datos almacenados en la Base de Datos.

- La sintaxis básica para su uso es:

```
SELECT  atr_1, atr_2, …, atr_k    // Atributos a mostrar
FROM    table_1, table_2, …, table_k // Tablas de los atributos utilizados
WHERE   condicion_x ;    // Condición que deben cumplir las tuplas
```

Ejemplo:

```
SELECT  CI, nombre
FROM    persona
WHERE   edad > 25 AND edad < 50;
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Eliminación de Datos

DELETE

- La sentencia *DELETE* elimina una o más tuplas dentro de una tabla, de acuerdo a cierta condición.

- La sintaxis básica para su uso es:

```
DELETE  FROM    nombre_tabla  // Eliminar de nombre_tabla
WHERE           condicion_x ;// La(s) tupla(s) que cumpla(n) con X
```

EJEMPLOS:

```
DELETE FROM     persona
WHERE           edad < 18;
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Creación de Dominios

CREATE DOMAIN

- Permite definir nuevos dominios para los valores de los datos, de manera tal que el usuario pueda definir de forma personalizada los valores esperados a partir de un tipo de dato primitivo.

- La sintaxis básica para su definición es:

```
CREATE DOMAIN  nombre_dom  // Crear el dominio nombre:_dom
AS             tipo_dato; // Tipo de dato correspondiente al dominio
```

EJEMPLOS:

```
CREATE  DOMAIN  nombre_corto  AS varchar (10);
CREATE  DOMAIN  nombre_largo  AS varchar (30);
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Creación de Dominios

CREATE DOMAIN

- De manera similar al caso del CREATE TABLE, es posible definir reglas sobre los valores de los datos de un dominio, como se muestra en los siguientes ejemplos:

```
CREATE  DOMAIN   tipo_cuenta  AS varchar(10)
    CHECK  ( VALUE = 'Ahorro'
             OR VALUE = 'Corriente' );


    CREATE  TABLE deposito ( nro_bauche integer,
                             cantidad     integer,
                             nro_cuenta  varchar(20),
                             tipo          tipo_cuenta);
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Creación de Vistas

CREATE VIEW

- Son consultas predefinidas, que se almacenan en la base de datos y se pueden utilizar a medida que sean necesarias, permitiendo reutilizar una misma consulta una y otra vez.

- Entre los principales usos de las vistas tenemos:

  - Ocultar la complejidad de los datos.
  - Simplificar comandos para el usuario.
  - Presentar los datos en una perspectiva diferente.
  - Guardar consultas que puedan ser útiles luego.

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Creación de Vistas

CREATE VIEW

- La sintaxis usada para la definición de vistas es la siguiente:

```
CREATE  VIEW    nombre_vista        // Vista a Crear
AS              select_x ;          //  SELECT a ejecutar
```

Ejemplo:

```
CREATE VIEW     hab_ciudad_america  AS
SELECT          C.nombre_c, C.habitantes
FROM            ciudad C
WERE            C.continente = 'america' and C.habitantes > 1999999
```

"Vista de todas las ciudades de América que poseen una población mayor o igual a 2.000.000"

PostgreSQL

Creación de Triggers

INTRODUCCIÓN

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

- PL/pgSQL: es un lenguaje procedimental cargable para PostgreSQL. Los objetivos propuestos para este consisten en permitir (entre otras cosas) que el mismo:

  - Sea usado para crear funciones y disparadores.
  - Adicione estructuras de control al lenguaje SQL.
  - Sea capaz de realizar cálculos complejos.
  - Herede tipos, funciones y operadores definidos por el usuario

Excepto por conversiones de E/S y funciones de cálculo para tipos definidos por el usuario, todo lo que puede definirse por medio de funciones en el lenguaje C puede definirse también con PL/pgSQL.

PostgreSQL

Creación de Triggers

INTRODUCCIÓN

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

- Funciones: son aquellas definidas por parte del usuario haciendo uso de un lenguaje procedimental (PL/pgSQL en nuestro caso).

- La sintaxis básica utilizada en su definición es la siguiente:

```
CREATE FUNCTION   nombre_func  (arg1 tipo_arg1,..., argk tipo_argk)
        RETURNS   tipo_ret
        AS        $$ definition $$
        LANGUAGE  lenguaje_usado;
```

PostgreSQL

+ **Agenda**

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

Creación de Triggers

INTRODUCCIÓN

- Ejemplos:

```
1. CREATE FUNCTION   concat_text  ( text, text ) RETURNS text  AS '
BEGIN
        RETURN $1 || $2;
 END;
' LANGUAGE plpgsql ;

2. CREATE FUNCTION   add  ( integer, integer ) RETURNS  integer
        AS'  select $1+$2; '
        LANGUAGE SQL;
```

PostgreSQL

+ **Agenda**

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

Creación de Triggers

CREATE TRIGGER

- Triggers: es un bloque de código en lenguaje PL/pgSQL, el cual se ejecuta implícitamente cuando ocurre un evento asociado a una tabla de la base de datos.

- Los triggers se lanzan automáticamente cuando se ejecuta una sentencia *INSERT, UPDATE* ó *DELETE* contra una tabla específica que se encuentre asociada al trigger en cuestión.

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

Creación de Triggers

CREATE TRIGGER

- *Momento de Ejecución*: es cuando se tiene que ejecutar el trigger con relación al evento. Puede ser antes (BEFORE) o después (AFTER). Este es el punto más importante en el momento de crear un trigger

- *Evento:* es la operación u operaciones que provocarán la ejecución del trigger ¿Cuáles son? Un INSERT, un UPDATE, un DELETE o incluso una combinación de ellas sobre una tabla.

- *Nombre de la tabla:* es simplemente la tabla sobre la que se crea el trigger (ON TABLE).

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

Creación de Triggers

CREATE TRIGGER

- *Tipo de trigger:* esto determinará cuantas veces debe ejecutarse el cuerpo del trigger cuando ocurre el evento (ROW, STATEMENT).

- *Cláusula WHEN:* especifica la restricción de un trigger. La condición se evalúa para cada registro para determinar si se ejecuta el cuerpo del trigger o no.

- *Cuerpo del trigger:* es aquí donde se escribe el código del trigger en PL/pgSQL. Se trata de escribir la(s) acción(es) a realizar. Es importante reseñar que dentro del cuerpo del trigger se tiene acceso a los valores nuevos (*NEW*) y viejos (*OLD*) del registro que se está procesando.

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

Creación de Triggers

CREATE TRIGGER

- La sintaxis básica utilizada para definir un Trigger es la siguiente:

```
CREATE TRIGGER    nomb_trigger
{ BEFORE | AFTER }
{ INSERT | UPDATE | DELETE }

ON      nombre_tabla
FOR  EACH { ROW | STATEMENT }
EXECUTE  PROCEDURE  nomb_func  ( arg_func );
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

Creación de Triggers

CREATE TRIGGER

- Ejemplo: Empleado (id int4, sueldo integer);

```
CREATE FUNCTION  validar_emp() RETURNS trigger  AS $val_emp$
BEGIN
        IF NEW.salario IS NULL THEN
           RAISE EXCEPTION 'El salario del empleado no debe ser nulo';
        ENDIF;

        IF NEW.salario < 0 THEN
           RAISE EXCEPTION 'El salario no debe ser negativo';
        ENDIF;

        RETURN NEW;
END;

$val_emp$          LANGUAGE  plpgsql;
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
  - PL / pgSQL
  - Funciones
  - Triggers
- Eliminación

Creación de Triggers

CREATE TRIGGER

- Una vez definida la función que se ejecutará al dispararse el trigger, se crea entonces el mismo:

```
CREATE TRIGGER validar_empleado
BEFORE INSERT OR UPDATE
ON Empleado FOR EACH ROW

EXECUTE PROCEDURE validar_emp();
```

PostgreSQL

+ Agenda

- Conexión
- PostgreSQL
- Creación de Tablas
- Reglas y Restricciones
- Inserción de Datos
- Actualización de Datos
- Selección de Datos
- Eliminación de Datos
- Creación de Dominios
- Creación de Vistas
- Creación de Triggers
- Eliminación

Eliminación

DROP

- Esta es la sentencia utilizada para eliminar cada uno de los objetos de la base de datos que han sido definidos o agregados por parte del usuario.

- La sintaxis según el caso es la siguiente:

```
DROP TABLE       nombre_tabla;
DROP CONSTRAINT  nombre_constraint;
DROP DOMAIN      nombre_dominio;
DROP VIEW        nombre_vista;
DROP FUNCTION    nombre_funcion;
DROP TRIGGER     nombre_trigger;
```

PostgreSQL

Preguntas

Observaciones

Enlaces Importantes:

PostgreSQL:
http://www.postgresql.org

Documentación PostgreSQL:
http://www.postgresql.org/files/documentation/pdf/8.2/postgresql-8.2-A4.pdf