**UD4 BASES DE DATOS: DDL Y DML**

**CREAR UNA BASE DE DATOS**

* **Con una sentencia**

En la pantalla de consultas de una tabla cualquiera de una base de datos cualquiera:

**CREATE DATABASE** **Nombre de la base de datos;**

* **Desde el Terminal**

createdb -U postgres -W -e **NombreBaseDeDatos**;

**BORRAR UNA BASE DE DATOS**

* **Con una sentencia**

En la pantalla de consultas de una tabla cualquiera de una base de datos cualquiera (y además no hay nadie conectada a esta):

**DROP DATABASE** **Nombre de la base de datos;**

* **Desde el Terminal**

dropb -U postgres -W **NombreBaseDeDatos**;

**CREACIÓN Y ELIMINACIÓN DE TABLAS E INSERCIÓN DE VALORES**

En la pantalla de consultas de nuestra base de datos:

**CREATE TABLE my\_first\_table (**

**first\_column text,**

**second\_column integer**

**);**

Para crear una tabla se sigue esta secuencia:

**CREATE TABLE NombreBaseDeDatos (nombreColumna tipoDato, otraColumna tipoDato);**

Para borrarla se hace de la siguiente forma:

**DROP TABLE NombreBaseDeDatos;**

Para dar valores a esas columnas escribimos la siguiente sentencia:

**INSERT INTO my\_first\_table**

**VALUES ('Hola', 1);**

Ahora first\_column tiene una fila en la que pone ‘Hola’ y second\_column una fila en la que pone 1.

* **Añadir valores por defecto**

**CREATE TABLE productos (**

**num\_producto integer,**

**nombre text,**

**precio numeric DEFAULT 9.99**

**);**

En este caso, si no le damos un valor en una fila a una columna, en ella aparecerá el valor por defecto que le hayamos dado nosotros en lugar de NULL.

**TIPOS DE DATOS**

**COLUMNAS GENERADAS**

**RESTRICCIONES**

**MODIFICAR TABLAS**

* **Añadir columnas**

**ALTER TABLE producto**

**ADD COLUMN marca\_producto VARCHAR(200),**

**ADD COLUMN fecha\_caducidad DATE;**

* **Borrar columnas**

**ALTER TABLE producto DROP COLUMN fecha\_caducidad;**

**MANIPULACIÓN DE DATOS**

Para crear una nueva fila, se utiliza el comando INSERT. El comando requiere el nombre de la tabla y los valores de las columnas. Por ejemplo, pensemos en la siguiente definición de tabla:

**CREATE TABLE producto (**

**num\_producto INTEGER,**

**nombre TEXT,**

**precio NUMERIC,**

**CONSTRAINT pk\_productos PRIMARY KEY (num\_producto)**

**);**

Un ejemplo de comando de inserción de una fila sería:

**INSERT INTO producto VALUES (1, 'Queso', 9.99);**

La siguiente sintaxis nos muestra cómo podemos modificar el orden de las columnas en la tabla a la hora de insertar.

**INSERT INTO productos (nombre, precio, num\_producto) VALUES ('Queso', 9.99, 1);**

Esto NO modifica de forma permanente el orden de las columnas, sino que nos permite indicar otro orden para los valores a insertar.

Si no tienes valores para todas las columnas, puedes omitir algunas de ellas. En ese caso, las columnas se llenarán con sus valores predeterminados (que pueden ser un valor predeterminado si dicha columna tiene una restricción DEFAULT o simplemente, un NULL por defecto). Por ejemplo:

**INSERT INTO productos (num\_producto, nombre) VALUES (1, 'Cheese');**

Se pueden insertar múltiples filas en un solo comando:

**INSERT INTO productos (num\_producto, nombre, precio) VALUES**

**(1, 'Queso', 9.99),**

**(2, 'Pan', 0.99),**

**(3, 'Leche', 1.25);**

* **Para cuando tengamos una columna con el tipo de dato SERIAL**

Este comando sirve para establecer el número siguiente de forma distinta a la autoincrementación:

**ALTER SEQUENCE producto\_num\_producto\_seq**

**RESTART WITH 1000;**

Con este, los enteros se autoincrementan de 10 en 10

**ALTER SEQUENCE producto\_num\_producto\_seq**

**INCREMENT BY 10;**

Existen muchos más comandos como estos…

**ACTUALIZACIÓN DE DATOS**

Para actualizar filas existentes, se usa **UPDATE**. Estas sentencias tienen tres partes:

* El nombre de la tabla y la columna para actualizar
* El nuevo valor de la columna
* Qué fila (s) actualizar

Para escoger qué fila (o filas) se van a actualizar hay que especificar qué condiciones debe cumplir una fila para que se actualice. La única forma de garantizar que solamente se actualice una fila concreta es declarando la condición basada en la clave primaria.

Por ejemplo, este comando actualiza todos los productos que tienen un precio de 5 para que tengan un precio de 10:

**UPDATE productos SET precio = 10 WHERE precio = 5;**

**TRANSACCIONES**

Una transacción empaqueta varios pasos en una operación, de forma que se completen todos o ninguno. Los estados intermedios entre los pasos no son visibles para otras transacciones ocurridas en el mismo momento.

Los pasos dentro de una transacción son varias sentencias SQL que deben de completarse todas para que queden registradas.

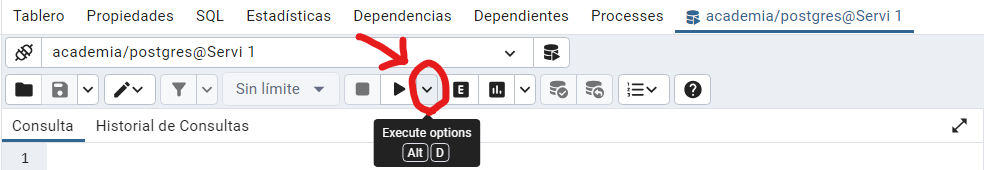
**Para comenzar una transacción utilizamos el comando BEGIN**.

**Para indicar al sistema que han terminado correctamente todas las sentencias SQL, utilizamos el comando COMMIT.**

Hay ocasiones en las que tenemos que desechar algunos de los pasos que se están realizando. **Para cancelar la transacción comenzada utilizamos el comando ROLLBACK.**

Para poder utilizar estos comandos mencionados (BEGIN, COMMIT y ROLLBACK) debemos de DESACTIVAR el **AUTOCOMMIT (ya que confirma y registra toda sentencia que quede ejecutada en la base de datos).**

Para ello clickamos en el botón de la flechita apuntando hacia abajo (situado a la derecha del botón de “Play”) y desactivamos la opción de Autocommit:

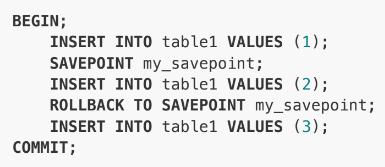


**El comando** **SAVEPOINT** **establece un nuevo punto de guardado en la transacción actual.** Un punto de guardado es una marca especial dentro de una transacción que permite revertir todos los comandos que se ejecutan después de que se estableció, restaurando el estado de la transacción a lo que era en el momento del punto de guardado.

Sintaxis: **SAVEPOINT nombreSavepoint;**

Podemos utilizar la variante **ROLLBACK TO SAVEPOINT para deshacer los cambios realizados hasta un determinado punto de guardado.** Los anteriores no se deshacen, pero quedarían pendientes de confirmación.

Un ejemplo:



La transacción anterior insertaría los valores 1 y 3, pero no el 2.

**Para establecer y destruir después un punto de guardado, podemos usar el comando RELEASE SAVEPOINT.**

**PAUTAS PARA CREAR UNA BASE DE DATOS**

1- Crear las tablas con todas las restricciones salvo las FK.

2- Insertamos los datos “actuales”.

3- Alterar las tablas para añadir las FK.

1- Crear las tablas con FK pero sin validar.

2- Insertar datos.

3- Validar FK.

**Las FK no son NUNCA de tipo (SMALL/BIG) SERIAL.**

Tipos de datos (FECHAS): <https://docs.aws.amazon.com/es_es/redshift/latest/dg/r_Datetime_types.html#r_Datetime_types-timestamp>

* Crear y borrar tablas.

**Crear:**

**CREATE TABLE my\_first\_table (**

**first\_column text,**

**second\_column integer**

**);**

**Borrar: DROP TABLE my\_first\_table;**

* Tipos de datos:
  + Tipos numéricos.

**DECIMAL, NUMERIC.**

* + Tipos coma flotante.

**REAL, DOUBLE PRECISION.**

* + Tipo enteros.

**SMALLINT, INTEGER, BIGINT.**

* + Tipos seriales.

**SMALLSERIAL, SERIAL, BIGSERIAL.**

* + Cadenas de caracteres.

**CHAR(**n**), VARCHAR(**n**), TEXT.**

* + Fechas.

**TIME, DATE, TIMESTAMP, TIMETZ, TIMESTAMPTZ.**

* Columnas generadas.

**CREATE TABLE televisores (**

**diagonal\_in numeric,**

**diagonal\_cm numeric GENERATED ALWAYS AS (diagonal\_in \* 2.54) STORED**

**);**

* Restricciones:
  + Check.

**CREATE TABLE producto (**

**num\_producto INTEGER,**

**nombre TEXT,**

**precio NUMERIC,**

**precio\_descontado NUMERIC,**

**CONSTRAINT precio\_positivo CHECK (precio > 0),**

**CONSTRAINT precio\_descontado\_positivo CHECK (precio\_descontado > 0),**

**CONSTRAINT descuento\_valido CHECK (precio\_descontado < precio)**

**);**

* + DEFAULT (Se escribe a la hora crear tablas).

**CREATE TABLE productos (**

**num\_producto integer,**

**nombre text,**

**precio numeric DEFAULT 9.99**

**);**

* + NOT NULL (Se escribe a la hora crear tablas).

**CREATE TABLE producto (**

**num\_producto INTEGER NOT NULL,**

**nombre TEXT NOT NULL,**

**precio NUMERIC**

**);**

* + Unicidad.

**CREATE TABLE temperatura\_diaria (**

**id\_ciudad integer,**

**fecha date,**

**maxima numeric,**

**minima numeric,**

**media numeric,**

**CONSTRAINT ciudad\_fecha\_unicas UNIQUE (id\_ciudad, fecha)**

**);**

* + Clave primaria [PK].

**CREATE TABLE temperatura\_diaria (**

**id\_ciudad integer,**

**fecha date,**

**maxima numeric,**

**minima numeric,**

**media numeric,**

**CONSTRAINT pk\_temperatura\_diaria PRIMARY KEY (id\_ciudad, fecha)**

**);**

* + Clave externa [FK].

**ALTER TABLE items\_pedido**

**ADD CONSTRAINT pk\_items\_pedido PRIMARY KEY (num\_producto, id\_pedido),**

**ADD CONSTRAINT fk\_items\_pedido\_producto FOREIGN KEY (num\_producto)**

**REFERENCES producto ON DELETE RESTRICT,**

**ADD CONSTRAINT fk\_items\_pedido\_pedido FOREIGN KEY (id\_pedido) REFERENCES**

**pedido ON DELETE CASCADE**

**);**

* Políticas de borrado:
  + Cascada.
  + SET NULL.
  + SET DEFAULT.
  + No action.
  + RESTRICT.
* Añadir y eliminar columnas.
* Añadir y eliminar una restricción.
* Cambiar el tipo de dato.
* Renombrar columnas y tablas.
* Inserción de datos.
* Actualización de datos.
* Eliminación de filas.
* Transacciones.
  + BEGIN.
  + COMMIT.
  + ROLLBACK.
  + SAVEPOINT.
  + ROLLBACK TO SAVEPOINT.