**UNIDAD 3: PostgreSQL**

1. [UNIDAD 2: Introducción](https://learn.nextu.com/mod/page/view.php?id=10253&pid=P_WEB_DATABASE)

**MySQL**

Con la idea de ampliar las opciones de herramientas que puedes utilizar para implementar tu solución de base de datos relacional, en esta unidad trabajaremos con el sistema manejador de base de datos PostgreSQL. En esta unidad buscamos que puedas aplicar los conceptos de bases de datos, que ya has aprendido en el curso usando este manejador, con la ayuda de la línea de comando y la herramienta gráfica PgAdmin. La idea es que te vayas familiarizando con la utilización de los conceptos de bases de datos relacionales y su representación en PostgreSQL. Seguimos usando nuestro enfoque teórico-práctico, para que con la ayuda de talleres y ejercicios puedas explorar y practicar los comandos y sentencias explicadas en video y lecturas.

**Objetivos de aprendizaje**

1. Interpretar las diferentes opciones para definir usuarios y sus correspondientes permisos con la finalidad de restringir el acceso a tus datos a diferentes tipos de usuarios. Crear claves primarias y foráneas y sus restricciones a una solución de base de datos en MySQL.
2. Diferenciar los distintos tipos de funciones predefinidas en PostgreSQL con la finalidad de realizar operaciones más avanzadas sobre nuestra base de datos.
3. Crear vistas en PostgreSQL con la finalidad de visualizar el contenido de nuestras tablas más apropiadamente de acuerdo con el alcance de las consultas.
4. Aplicar restricciones y usar secuencias.

2. Lección 1: Conociendo PostgreSQL

[2.1. PSQL](https://learn.nextu.com/mod/lesson/view.php?id=10261&pid=P_WEB_DATABASE)

Acceder por consola con el comando:

Cliente – Servidor

Cliente: psql

Comando:

>psql -U postgres -W -h localhost

Argumentos

-W 🡪 es para que el sistema pida la clave

-h 🡪 localhost o IP

-d 🡪 Nombre de la base de datos para ingresar directamente en la conexión

Comandos postgres:

Postgres=# \l 🡪 Para listar las DBs

Postgres=# CREATE DATABASE practicando; 🡪 Para crear una DB con el nombre practicando

Postgres=# \c practicando; 🡪 Para cambiar a la base de datos practicando (todo en posgres tiene un usuario y una contraseña)

practicando=# \dt 🡪 Para listar tablas

practicando=# CREATE TABLE usuarios(

practicando(# id SERIAL, 🡪 Incrementar la clave y no nulo

practicando(# nombre VARCHAR NOT NULL); 🡪 Para crear tablas

practicando=# \d+ usuarios 🡪 Describe la tabla y para salir se presiona q

practicando=# INSERT INTO usuarios (nombre) VALUES (‘Luis’); 🡪 Para Insertar datos

practicando=# \q 🡪 Para salir del prompt de PostgreSQL

[2.2. Taller 1](https://learn.nextu.com/mod/lesson/view.php?id=10261&pageid=46661&pid=P_WEB_DATABASE)

Descargar e instalar PostgreSQL

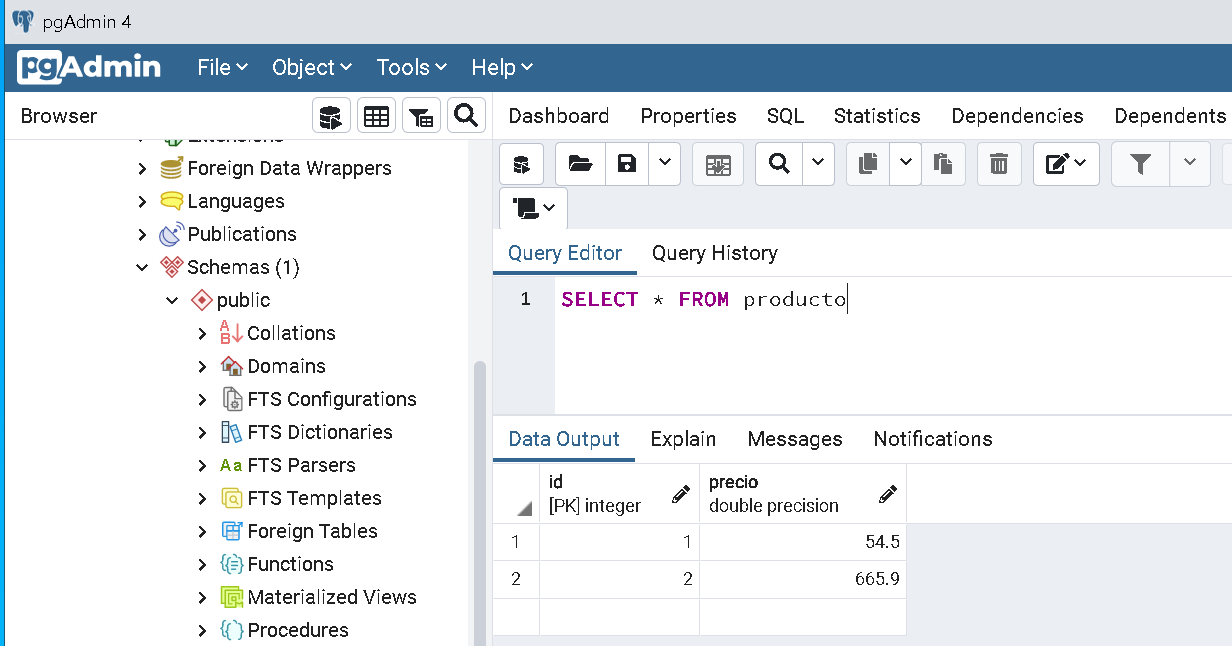
<https://www.postgresql.org/download/>

<https://www.pgadmin.org/>

2.3. Taller 2

2.4. PGAdmin

2.5. Taller 3



2.6. ¿Sabías qué?

**Tipos de Datos en PostgreSQL**

¡Bienvenido!

En esta lectura te proporcionamos un resumen de los tipos de datos más utilizados en PostgreSQL, categorizados en tres grupos:



**Strings**

| **Tipo** | **Descripción** |
| --- | --- |
| **CHAR(largo)** | Secuencia de caracteres (string) de tamaño fijo |
| **VARCHAR(largo)** | Secuencia de caracteres (string) de tamaño variable |
| **TEXT** | String largos |
|  |  |



**Numéricos**

| **Tipo** | **Descripción** |
| --- | --- |
| **SMALLINT** | Entero desde -32768 a 32767 (usado sin signo para booleanos) |
| **INTEGER** | Entero desde -2147483648 a 2147483647 |
| **BIGINT** | Entero desde -9223372036854775808 a 9223372036854775807 |
| **FLOAT** | Decimal Pequeño (8 bytes) |
| **REAL** | Decimal Grande (8 bytes) |
| **NUMERIC(n,s)** | Decimal con n dígitos normales y sin dígitos en la parte decimal |



**Otros**

| **Tipo** | **Descripción** |
| --- | --- |
| **DATE** | Fecha |
| **TIMESTAMP** | Fecha y hora HH:MM:SS |
| **TIME** | Hora |
| **SERIAL** | Entero autoincremental |
| **BOOLEAN** | True o false (verdadero o falso) |
| **CIRCLE** | Círculos |
| **POINT** | Punto en el plano |
| **MONEY** | Dinero |
| **CIDR** | Dirección IP |

Esperamos que esta guía sea útil a la hora de escoger tus datos en PostgreSQL.

3. Lección 2: Triggers y Stored Procedures

[3.1. Triggers](https://learn.nextu.com/mod/lesson/view.php?id=10262&pid=P_WEB_DATABASE)

Click derecho en “Trigger Functions”.

1. Primero se define el nombre del trigger
2. Luego se define el lenguaje para crear el trigger (opciones: internal, c o plpgsql)
3. Sí se selecciona plpgsql, se procede a ingresar el código(code), el cual debe ser un SQL (siempre se debe retornar un valor), por ejemplo:

BEGIN

INSERT INTO cambios\_de\_nombre (descripción, fecha) VALUES

(CONCAT(‘Cambio de nombre de: <’, OLD.nombre, ‘> a <’, NEW.nombre,’>’), NOW());

RETURN NEW;

END

1. Luego se debe ir a la tabla donde se desea que se dispare el trigger, se expande la tabla y se busca 🡪 Triggers, botón derecho y Trigger, se ingresa el nombre del trigger y luego en Definition se define cual es el Trigger que se desea disparar, por ejemplo:

Public.registrar\_cambio\_de\_nombre

1. Por último en la sección de Eventos se define cuando se quiere disparar el Trigger: BEFORE, AFTER o INSERT; UPDATE; DELETE or TRUNCATE, y se determina cual columna va a afectar (nombre)

[3.2. Ejercicio Práctico 1](https://learn.nextu.com/mod/lesson/view.php?id=10262&pageid=46671&pid=P_WEB_DATABASE)

3.3. Stored Procedures

Postgres no tiene el objeto SP definidos con ese nombre.

Lo que nos ofrece son las funciones, se pueden invocar cuando sea requerido

3.4. Ejercicio Práctico 2

3.5. ¿Sabías qué?

**Cápsula de conocimiento**

**Triggers y Funciones de PostgreSQL**

Aquí un resumen de algunos aspectos interesantes sobre triggers y funciones.

**Triggers**

1. Los triggers se implementan en dos partes:
   * La función que hace el cambio.
   * La asociación del trigger a la tabla.
2. El código asociado a la función debe estar entre **BEGIN** y **END**.
3. El objeto **NEW** representa al nuevo registro y **OLD** al registro antes de los cambios.
4. La función del trigger suele devolver el objeto **NEW** si lo modifica, y en caso contrario devuelve **NULL.**
5. Los triggers que se ejecutan antes (**BEFORE**) de una condición, pueden modificar los campos asignando **NEW.[campo] = [valor]**.
6. Los triggers que se ejecutan después (**AFTER**) de una condición, modifican los campos ejecutando un SQL como **UPDATE**, **INSERT**, etc.

**Funciones**

1. En PostgreSQL no existen Stored Procedures, en lugar de eso se usan las funciones de una forma general.
2. El código asociado a la función debe estar entre **BEGIN** y **END**.
3. Una función se puede definir con valor de retorno void, para lo cual no debe hacer **RETURN** al final.
4. Aunque la gente suele usar el lenguaje de PostgreSQL (plpgsql) para definir las funciones, estas pueden estar escritas en otros lenguajes como C, Python, Perl, Ruby, habilitando esos otros lenguajes.

Esperamos que esta información sea útil cuando utilices las funciones y triggers en PostgreSQL.

4. Lección 3: Funciones Predefinidas

[4.1. Funciones Matemáticas](https://learn.nextu.com/mod/lesson/view.php?id=10263&pid=P_WEB_DATABASE)

PGAdmin

Constantes:

Pi()

Sqrt() 🡪 Raíz cuadrada

Round() 🡪 Para redondear

4.2. Taller 1

4.3. Funciones de String y Fecha

CONCAT()

TRIM()

LOWER()

UPPER()

PostgreSQL a diferencia de MySQL si hace diferencia entre mayúsculas y minúsculas cuando se ejecuta un LIKE. Para no diferencias entre mayúsculas y minúsculas se usa la función ILIKE

Otra solución es utilizar el operador ~ para expresiones regulares, pero es sensible a mayúsculas y minúsculas, para no hacer sensible a mayúsculas y minúsculas se utiliza ~\*

Para hacer operaciones con fechas

Fecha + 20 🡪 En este caso se obtendrá la fecha 20 días después

EXTRACT(month FROM fecha) AS mes

Función para calcular la edad: AGE, requiere de una fecha

4.4. Taller 2

4.5. ¿Sabías qué?

4.6. Taller 3

5. Lección 4: Subqueries y Otros Objetos