



PostgreSQL

Introducción a PostgreSQL

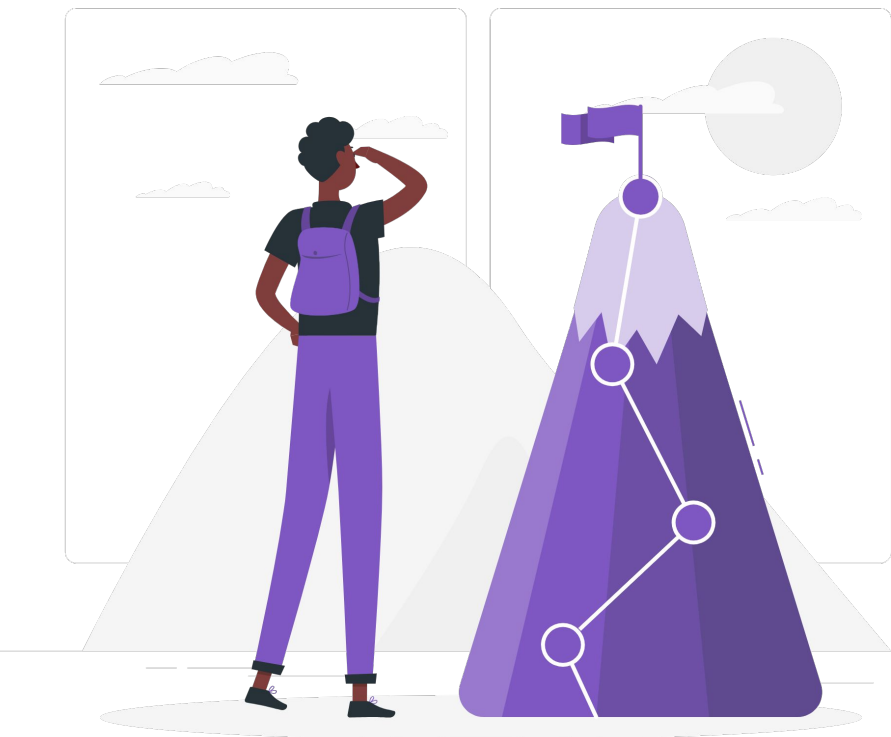
DEV.FX
DESARROLLAMOS(PERSONAS);

Elaborado por: César Guerra

www.cesarquerra.mx



Buy me a coffee



Objetivos de la Sesión

- Entender qué es PostgreSQL y sus ventajas.
- Instalar PostgreSQL.
- Aprender a iniciar y usar la interfaz de pgAdmin para acceder a nuestras bases de datos en PostgreSQL.
- Aprender a ejecutar Querys en pgAdmin.
- Crear bases de datos con pgAdmin.
- Crear tablas por medio de Querys.
- Comenzar a poblar la base de datos.



PostgreSQL

History of PostgreSQL

1986-
1985

1986-
1994

1994-
1995

INGRES

POSTGRES

POSTGRES95

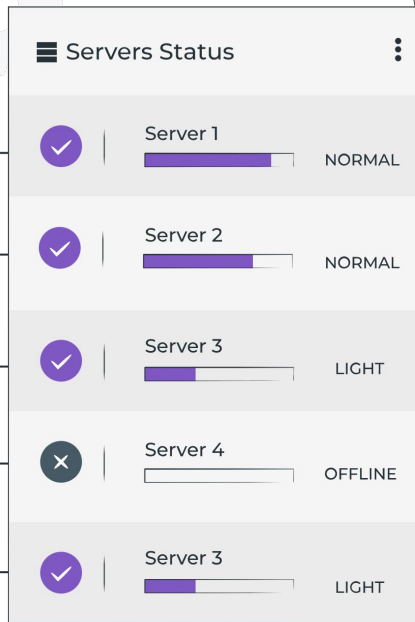
¿Qué es PostgreSQL?

PostgreSQL es un sistema de base de datos relacional de objetos de código abierto que **utiliza y amplía el lenguaje SQL**.

Los orígenes de PostgreSQL se remontan a 1986 como parte del proyecto POSTGRES de la Universidad de California en Berkeley y cuenta con más de 30 años de desarrollo activo en la plataforma central.



PostgreSQL



Ventajas de PostgreSQL

Instalación ilimitada. No hay costo asociado a la licencia del software, licencia BSD

Soporte (PostgreSQL Global Development Group): Su desarrollo no es controlado por una empresa, si no por una comunidad de desarrollares y organizaciones comerciales.

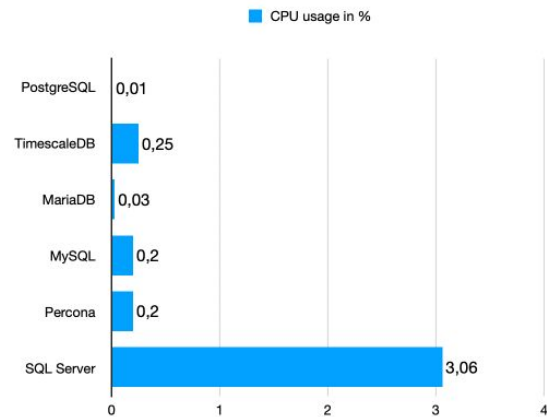
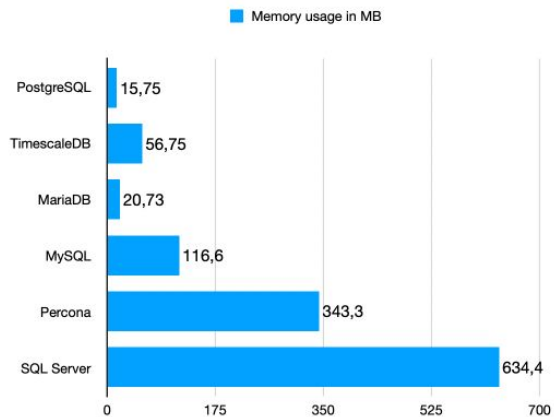
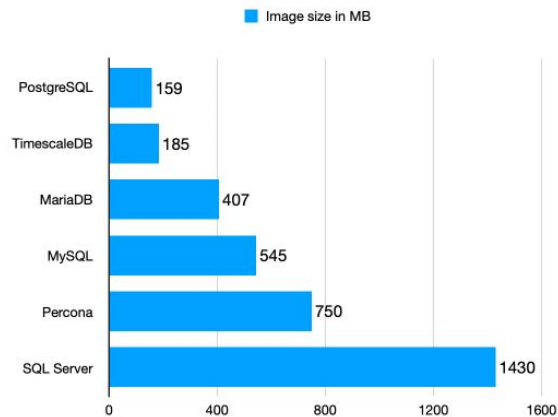
Ahorros considerables en costos de operación

Estabilidad y Confiabilidad: no presenta caídas en varios años de operación.

Extensible, código disponible sin costo.

Multiplataforma, disponible en casi cualquier Unix (34 plataformas en la última versión estable)

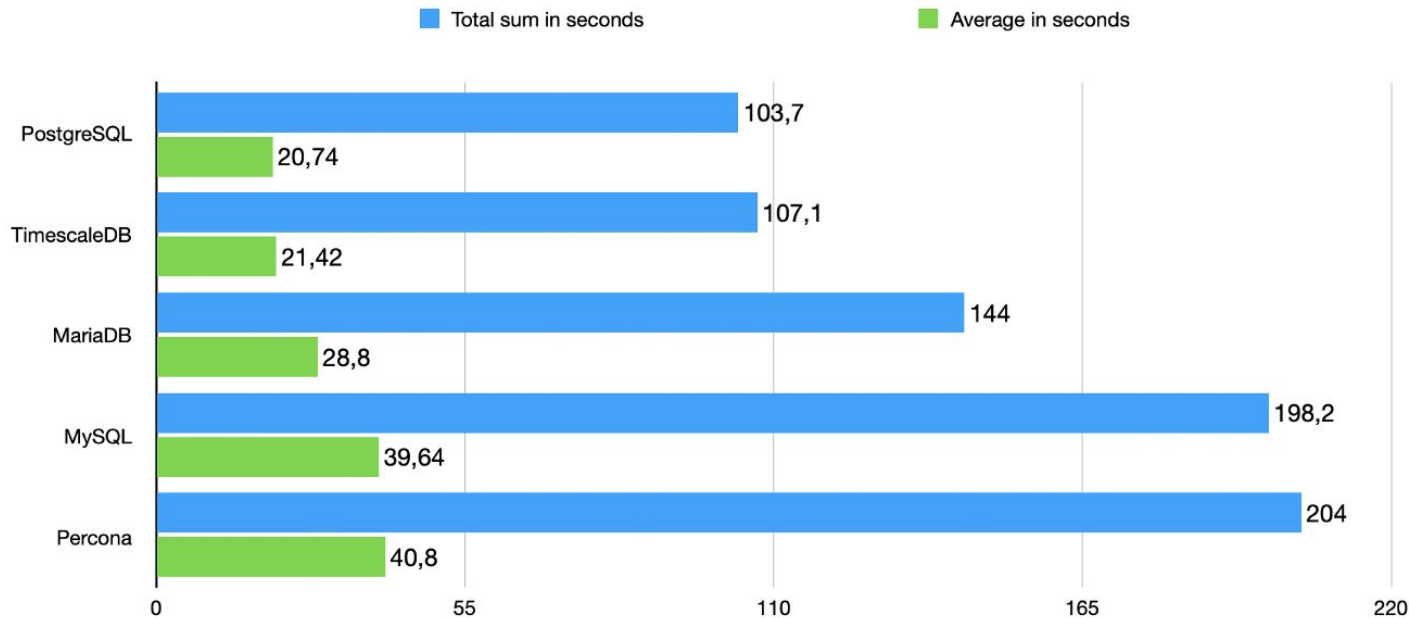
Comparativa de Consumo de Recursos



<https://itnext.io/benchmark-databases-in-docker-mysql-postgresql-sql-server-7b129368eed7>

Rendimiento al Insertar Registros

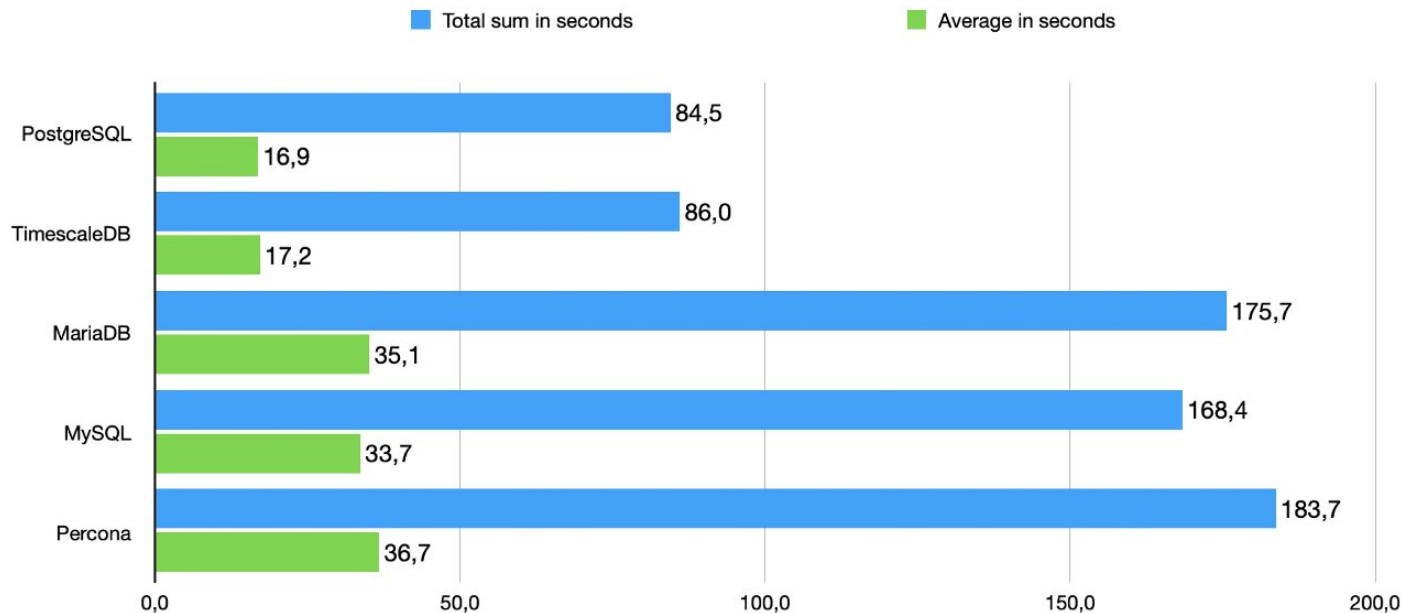
Inserción de 5 bloques de 10,000 registros cada uno:



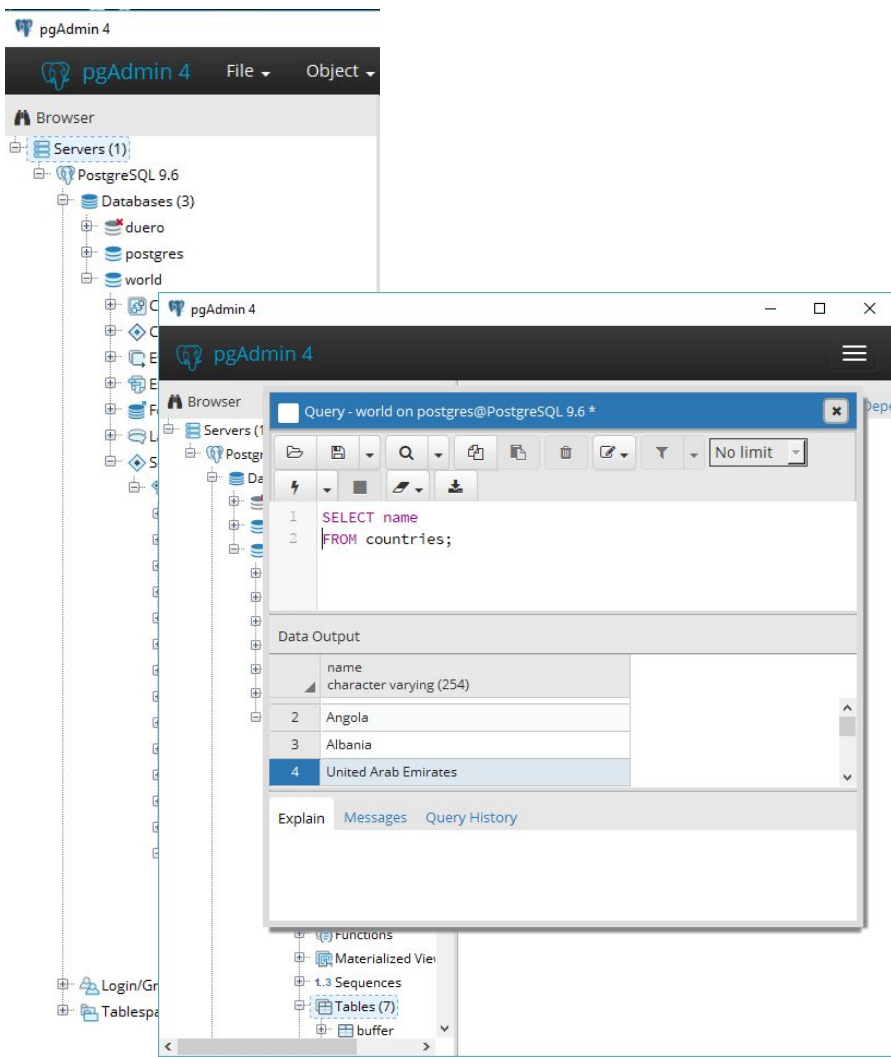
(menos es mejor)

Rendimiento al Realizar Consultas

5 rondas de 2000 iteraciones de average, sum, data.



(menos es mejor)



Gestión de PostgreSQL con pgAdmin

pgAdmin es una herramienta indispensable para gestionar y administrar PostgreSQL.

pgAdmin 4 es una versión robusta lanzada después de 14 años de desarrollo de sus antecesores y se ha construido usando Python y Javascript/jQuery.

pgAdmin 4 se incluye con los instaladores de PostgreSQL desde la versión 9.6.1 y desde la versión 9.6.6 se incluye pgAdmin 4 versión 2.0.


En resumen, nos va a permitir gestionar bases de datos de PostgreSQL de una forma más visual.

Instalación

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Descarga de PostgreSQL



Announcing BigAnimal: Fully managed PostgreSQL in the CloudPostgres Build 2021: 30 Nov to 1 Dec

DocsSign in

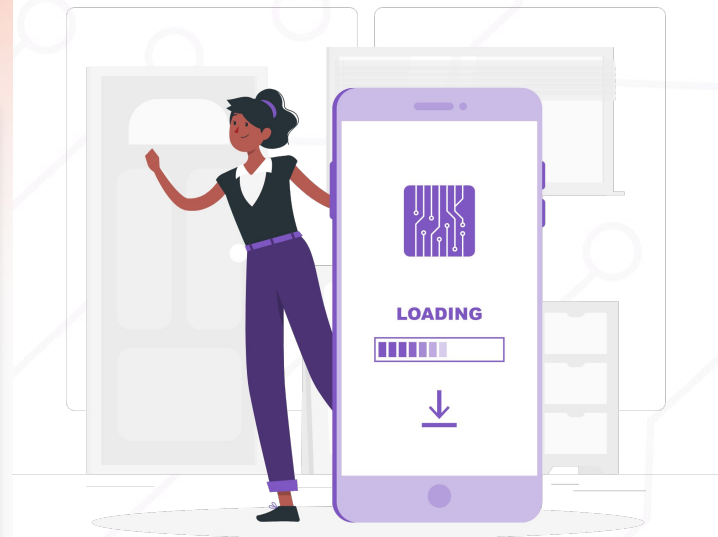
Why EDB?Cloud PostgreSQLPostgreSQL SoftwareServices & SupportResourcesPlansGet started

Contact us

Download PostgreSQL

Free PostgreSQL packages and installers from EDB

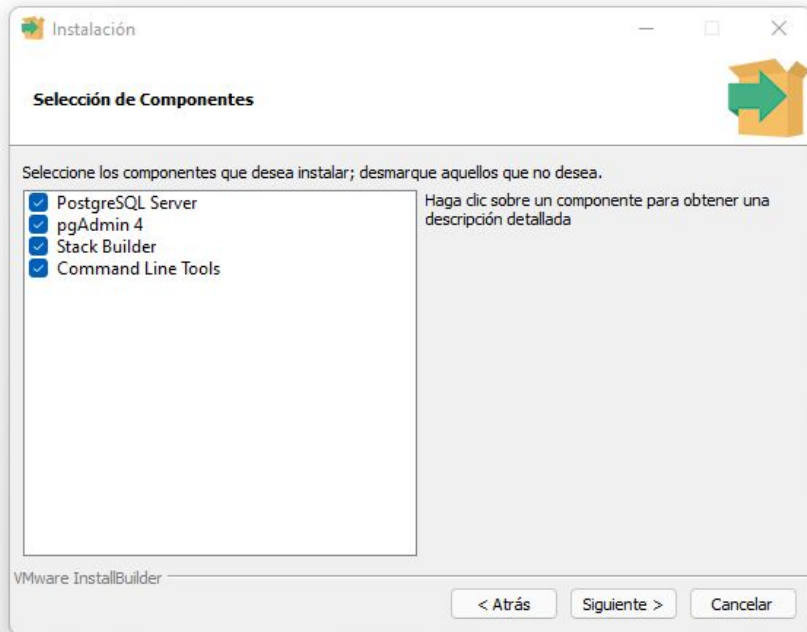
Versions	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
14.1	N/A	N/A	Download	Download	N/A
13.5	N/A	N/A	Download	Download	N/A
12.9	N/A	N/A	Download	Download	N/A
11.14	N/A	N/A	Download	Download	N/A
10.19	Download	Download	Download	Download	Download
9.6.24 (Not supported)	Download	Download	Download	Download	Download
9.5.25 (Not supported)	Download	Download	Download	Download	Download
9.4.26 (Not supported)	Download	Download	Download	Download	Download
9.3.25 (Not supported)	Download	Download	Download	Download	Download



<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

DEV.F

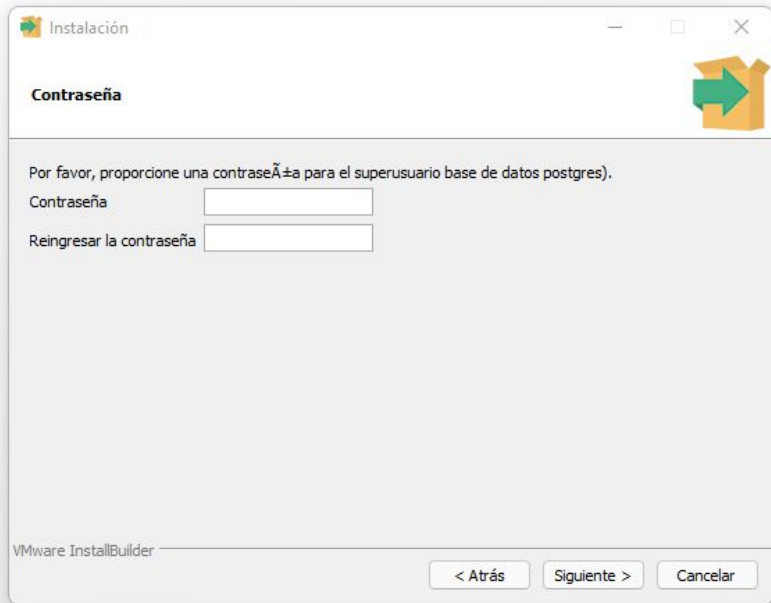
Instalar todo



Por defecto viene de esta forma, pero simplemente asegurarse que venga todo instalado.

Nos interesa sobre todo además de PostgreSQL Server tener el manejador pgAdmin 4.

Apuntar la Contraseña



Instalación

Contraseña

Por favor, proporcione una contraseña para el superusuario base de datos postgres).

Contraseña

Reingresar la contraseña

VMware InstallBuilder

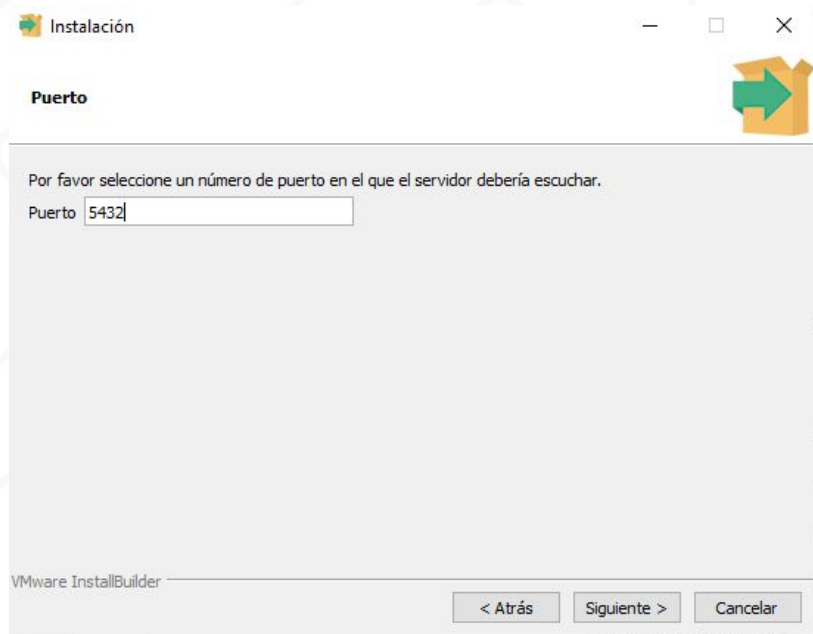
< Atrás Siguiente > Cancelar

IMPORTANTE

Durante la instalación se pedirá establecer una contraseña. Colocala y **APUNTALA.**

La contraseña será usada para iniciar sesión más adelante.

Confirmar el puerto



IMPORTANTE

Por defecto se instalará en el puerto **5432**.

Esto normalmente funciona por defecto a menos que tengamos dicho puerto en uso por otra aplicación o instalación.



pgAdmin

Management Tools for PostgreSQL

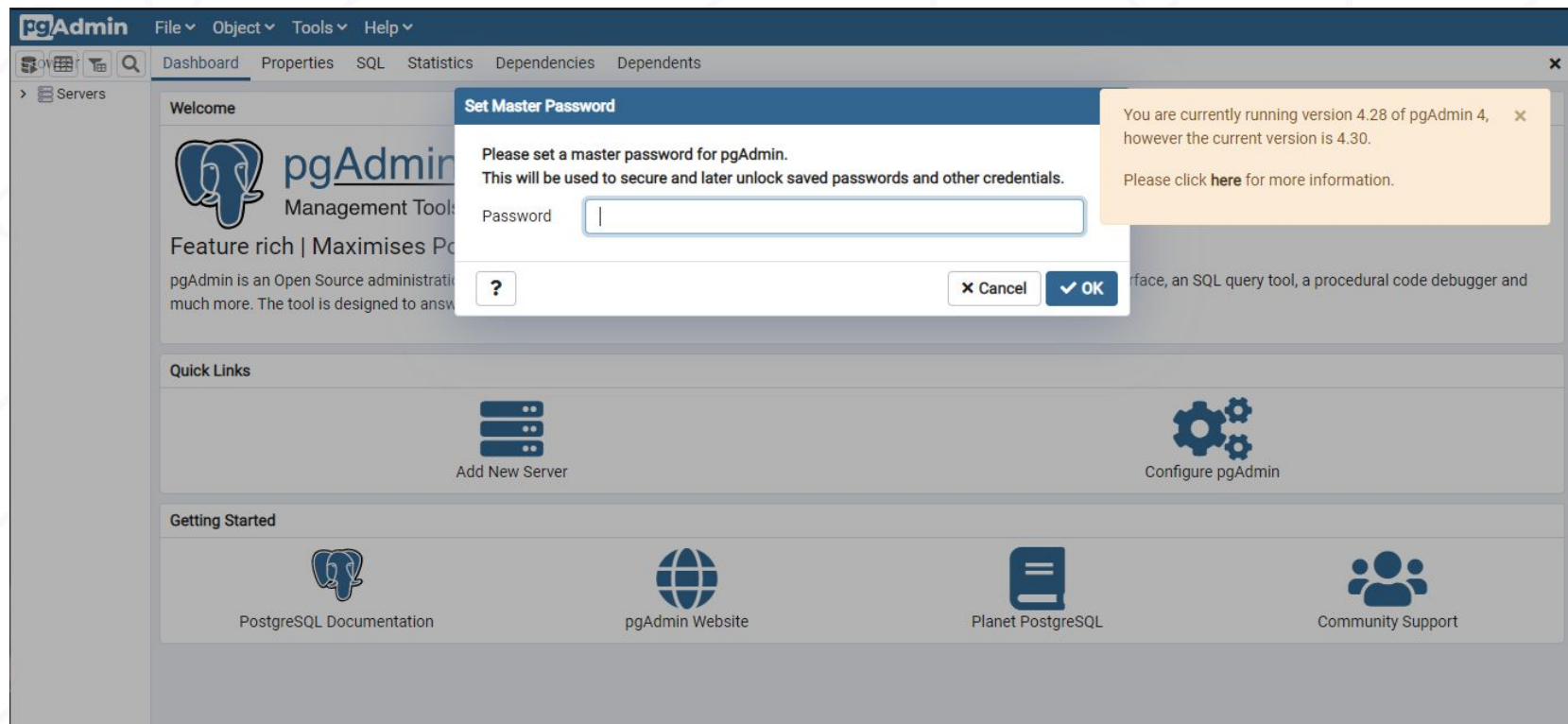
Waiting for pgAdmin 4 to start...

Uso de Postgres Con pgAdmin

DEV.FX
DESARROLLAMOS(PERSONAS);

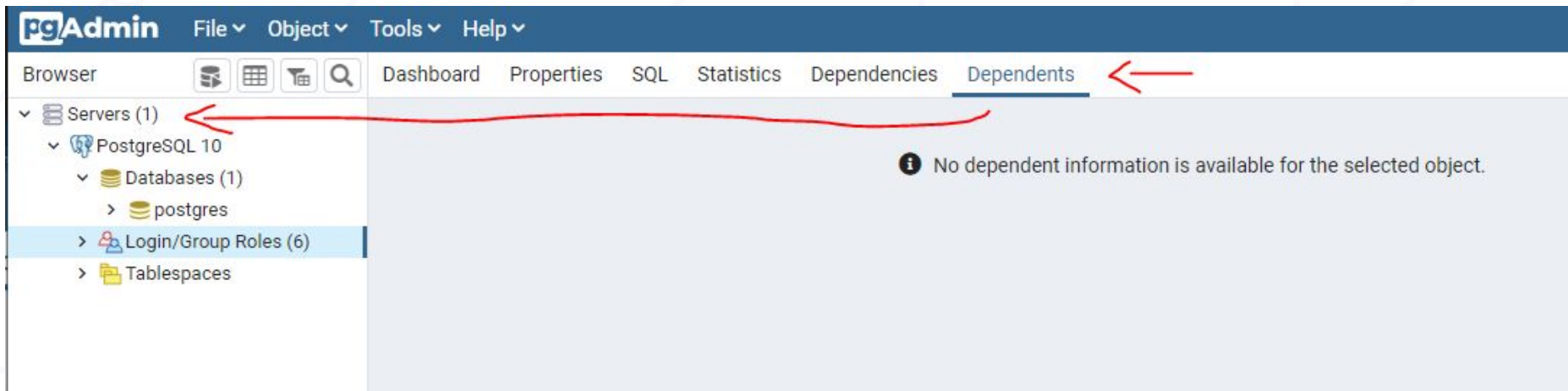
dev

Acceso a PGAdmin



Al ejecutar PGAdmin nos pedirá una contraseña. Aquí colocamos la que definimos al momento de la instalación

Ver los servidores y Bases de Datos: Dependents



Para ver los servidores, bases de datos y sus tablas, ir a la pestaña **Dependents** y Desplegar la parte de **Servers**.

Nota: En ocasiones es posible que vuelva a pedir la contraseña.

Creación de Base de Datos

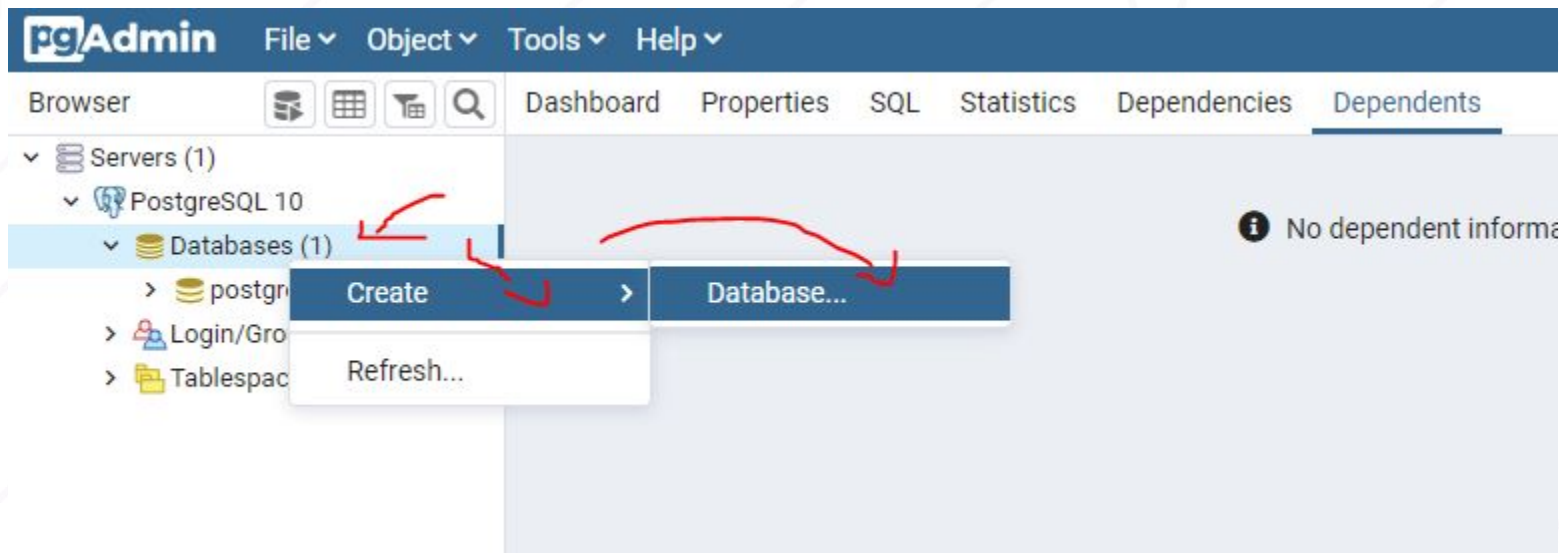
DEV.F
DESARROLLAMOS(PERSONAS);

dev

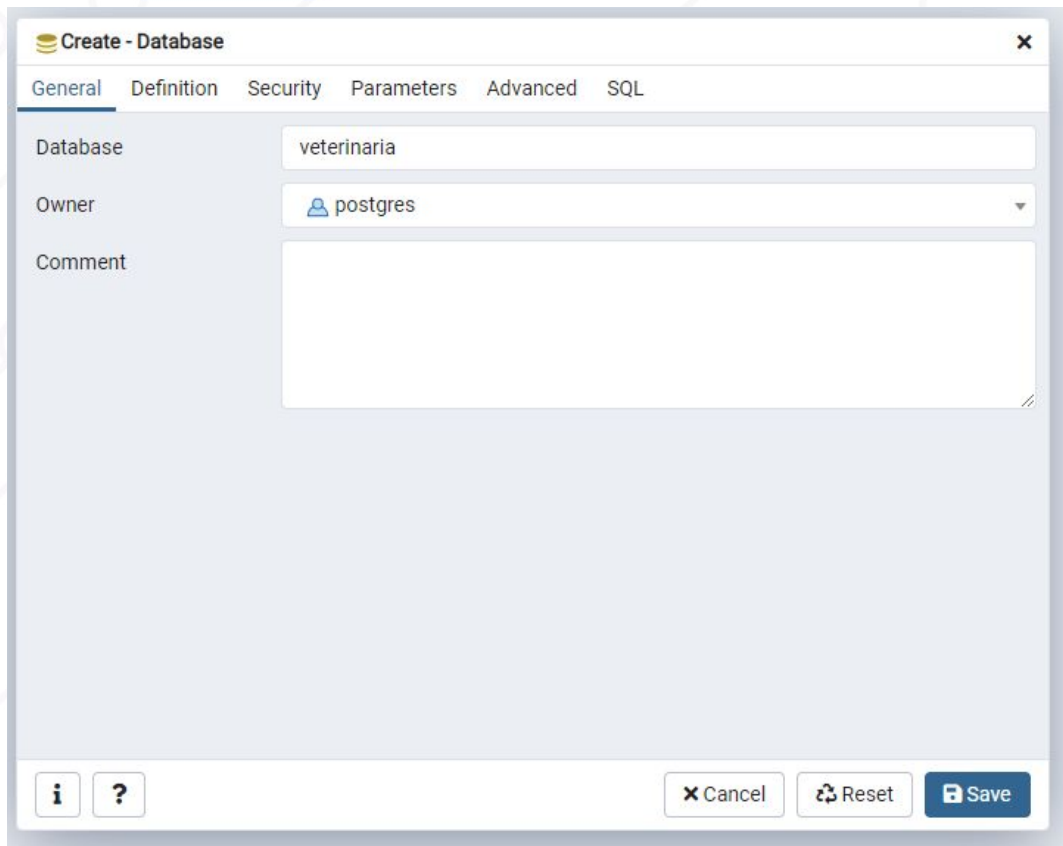
Creación de Base de Datos en PostgreSQL

Para crear una nueva base de datos:

Clic derecho en Databases -> Create -> Database



Creación de Base de Datos: Database & Owner



Create - Database

General Definition Security Parameters Advanced SQL

Database: veterinaria

Owner: postgres

Comment:

Buttons: [i] [?] [X Cancel] [Reset] [Save]

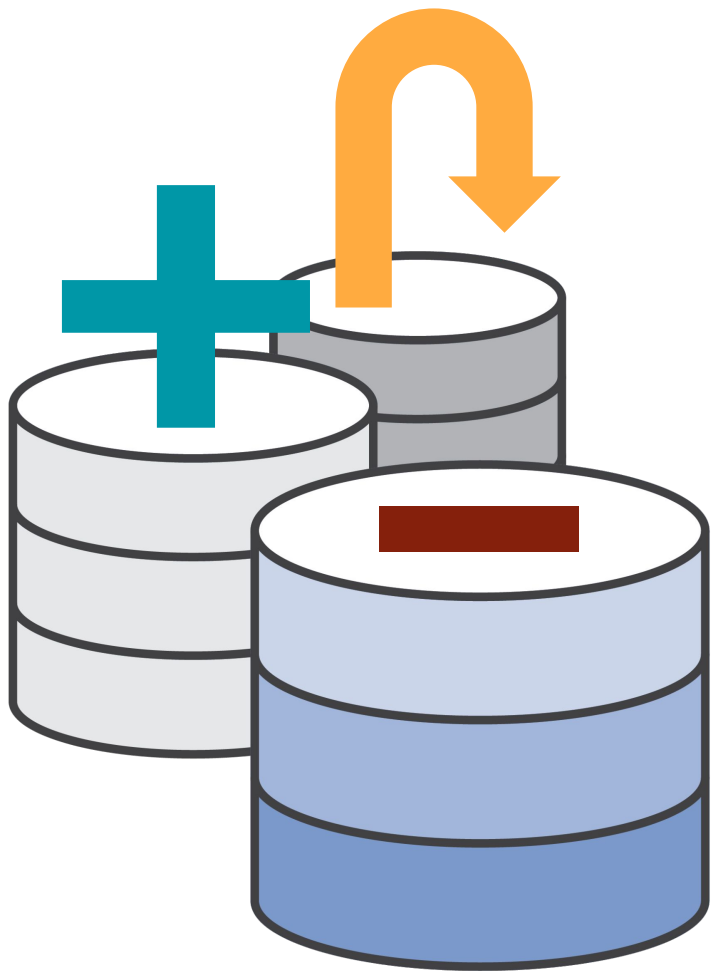
En la ventana siguiente indicamos el nombre de base de datos (**database**) y el usuario de la misma (**owner**).

El usuario **postgres** es el que se se crea por defecto y al cual asignamos contraseña al momento de la instalación.

Creación de Tablas

DEV.F
DESARROLLAMOS(PERSONAS);

dev



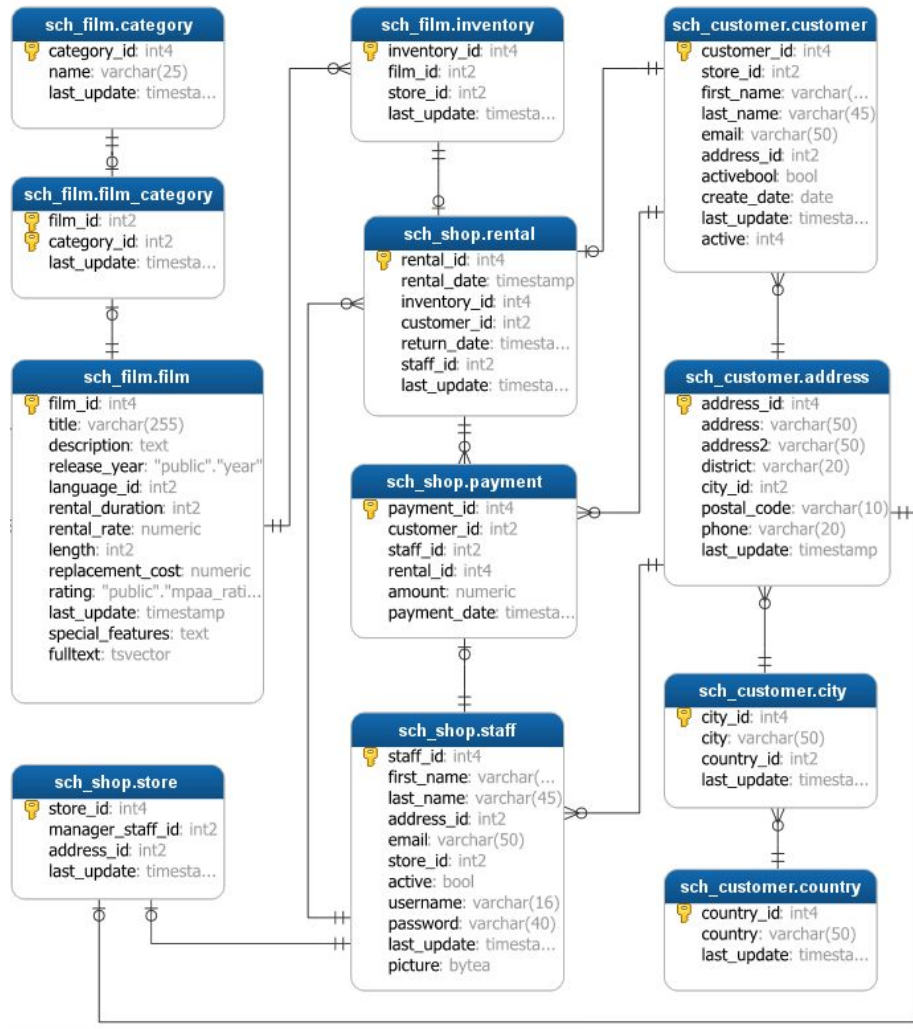
Lenguaje de Definición de Datos (DDL)

Este lenguaje permite a los programadores de un sistema gestor de base de datos, como Postgres, **definir las estructuras que almacenarán los datos así como los procedimientos o funciones que permitan consultarlos**. Para definir las estructura disponemos de tres sentencias:

CREATE: se usa para crear una base de datos, tabla, vistas, etc.

ALTER: se utiliza para modificar la estructura, por ejemplo añadir o borrar columnas de una tabla.

DROP: con esta sentencia, podemos eliminar los objetos de la estructura o toda la tabla.



Schema (Esquema)

El esquema de una base de datos (en inglés, database schema) describe la estructura de una base de datos.

En una base de datos relacional, **el esquema define sus tablas, sus campos en cada tabla y las relaciones entre cada campo y cada tabla.**

Tipos de Datos Números

■ 常用数据类型, 数字

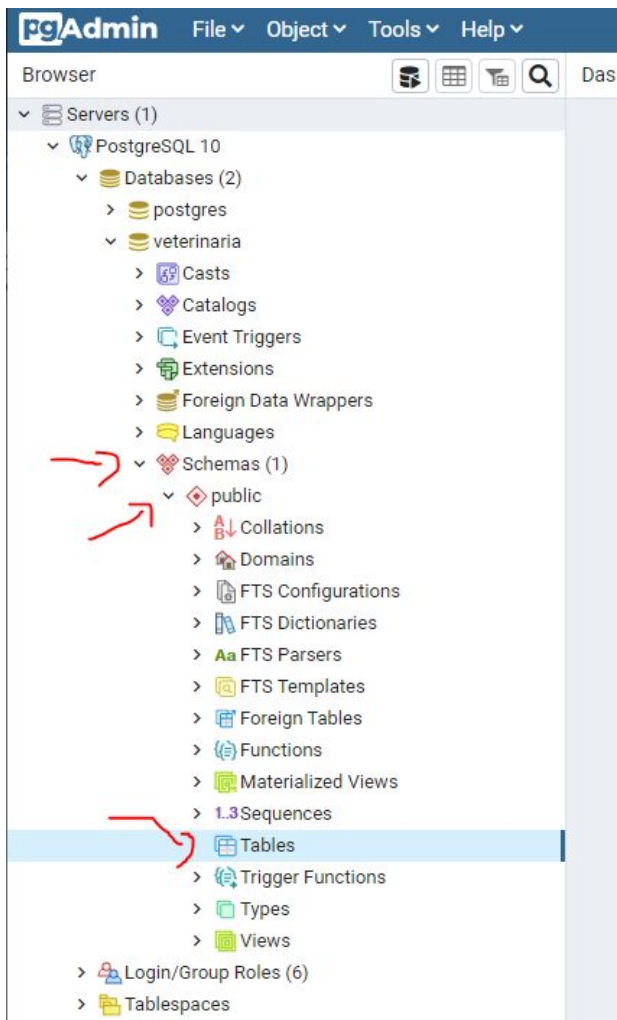
Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to 9223372036854775807
decimal / numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Tipos de Datos de Carácter

Name	Storage Size	Description
character varying(n), varchar (n)	variable(can store n chars)	variable-length with limit
character(n), char(n)	n chars	fixed-length, blank padded
text	variable	variable unlimited length
"char"	1 byte	single-byte internal type
name	64 bytes	internal type for object names

Tipos de Datos de Eventos

Name	Storage Size	Description	Low Value	High Value	Resolution
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond / 14 digits
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond / 14 digits
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
time [(p)] [without time zone]	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond / 14 digits
time [(p)] with time zone	12 bytes	times of day only, with time zone	00:00:00+14 59	24:00:00-1459	1 microsecond / 14 digits
interval [fields] [(p)]	12 bytes	time interval	-178000000 years	178000000 years	1 microsecond / 14 digits

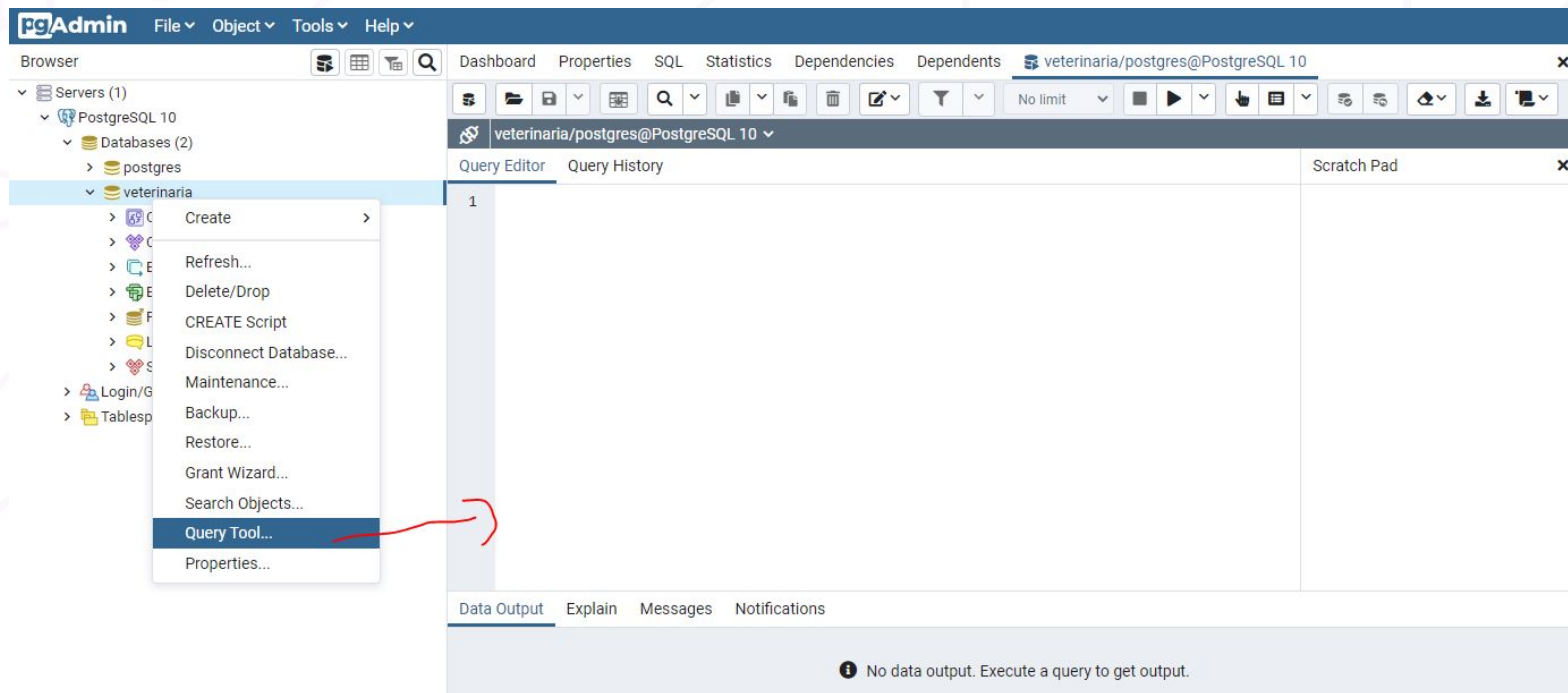


Visualizar las Tablas

Para poder visualizar las tablas que contiene una base de datos en pgAdmin, es necesario expandir la base de datos en cuestión y luego ir a:

Schemas -> public -> Tables

Query Tool



Es la herramienta que nos permitirá escribir consultas a la base de datos. Click derecho sobre el nombre de la base de datos y hacer clic en **Query Tool**.

```
CREATE TABLE films (  
  code      serial,  
  title     varchar(40),  
  did       integer,  
  date_prod date,  
  kind      varchar(10),  
  len       interval hour to minute,  
  PRIMARY KEY(code)  
);
```

<https://www.postgresql.org/docs/10/sql-createtable.html>

CREATE TABLE

Para crear una tabla, usamos la el comando **CREATE TABLE** seguido del nombre de la tabla y entre paréntesis comenzamos a colocar sus atributos.

Sobre nombrar tablas y atributos...

Es buena práctica que los nombres de tablas vayan en PLURAL y los atributos en SINGULAR.

Ejemplo: Tabla “Alumnos” con atributo “nombre”.

También hay quienes prefieren usar nombres de tablas solo en singular.

Demostración

Resolviendo el Ejercicio de Veterinaria

DEV.F.
DESARROLLAMOS(PERSONAS);

dev

Ejercicio: Veterinaria

Una veterinaria, requiere un sistema que le ayude a controlar la entrada de las mascotas a la veterinaria.

La veterinaria requiere almacenar el nombre de la mascota, el tipo de mascota, raza y edad.

Del dueño mascota se necesitan datos personales, cada mascota puede ser atendido por uno o varios veterinarios de los cuales necesitamos saber cédula profesional, nombre , edad años de experiencia.

Para este negocio, es necesario crear:

- 1. Diagrama Relacional**
- 2. Diagrama Entidad Relación**

Veterinaria: Diagrama E/R

owners	
owner_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)

pets	
pet_id	SERIAL
name	VARCHAR(150)
type	VARCHAR(150)
breed	VARCHAR(150)
age	INT
owner_id	INT

veterinarians	
veterinarian_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)
licence	VARCHAR(10)
years_experience	INT

attends	
id	SERIAL
pet_id	int
veterinarian_id	int

owners

owner_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)

Tabla owners

CREATE TABLE owners (

owner_id SERIAL PRIMARY KEY,

first_name VARCHAR(150) NOT NULL,

last_name VARCHAR(150) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL,


phone VARCHAR(15) NOT NULL

);

Dashboard Properties SQL Statistics Dependencies Dependents veterinaria/postgres@PostgreSQL 10

Query Editor Query History

```
1 CREATE TABLE owners (
2   owner_id serial PRIMARY KEY,
3   first_name VARCHAR(150) NOT NULL,
4   last_name VARCHAR(150) NOT NULL,
5   email VARCHAR(150) UNIQUE NOT NULL,
6   phone VARCHAR(15) NOT NULL
7 )
```



pets

pet_id	SERIAL
name	VARCHAR(150)
type	VARCHAR(150)
breed	VARCHAR(150)
age	INT
owner_id	INT

Tabla pets

```
CREATE TABLE pets (
```

```
    pet_id SERIAL PRIMARY KEY,
```

```
    name VARCHAR(150) NOT NULL,
```

```
    type VARCHAR(150) NOT NULL,
```

```
    breed VARCHAR(150) NOT NULL,
```

```
    age INT NOT NULL,
```

```
    owner_id INT NOT NULL,
```

```
    FOREIGN KEY (owner_id) REFERENCES  
    owners(owner_id)
```

```
);
```

veterinaria/postgres@PostgreSQL 10

Query Editor Query History

```
1 CREATE TABLE pets (  
2     pet_id SERIAL PRIMARY KEY,  
3     name VARCHAR(150) NOT NULL,  
4     type VARCHAR(150) NOT NULL,  
5     breed VARCHAR(150) NOT NULL,  
6     age INT NOT NULL,  
7     owner_id int NOT NULL,  
8     FOREIGN KEY (owner_id) REFERENCES owners(owner_id)  
9 )
```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 49 msec.

veterinarians

veterinarian_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)
licence	VARCHAR(10)
years_experience	INT

Tabla veterinarians

```
CREATE TABLE veterinarians (  
  
    veterinarian_id SERIAL PRIMARY KEY,  
  
    first_name VARCHAR(150) NOT NULL,  
  
    last_name VARCHAR(150) NOT NULL,  
  
    email VARCHAR(150) UNIQUE NOT NULL,  
  
    phone VARCHAR(15) NOT NULL,  
  
    licence VARCHAR(10) UNIQUE NOT NULL,  
  
    years_experience INT  
  
);
```

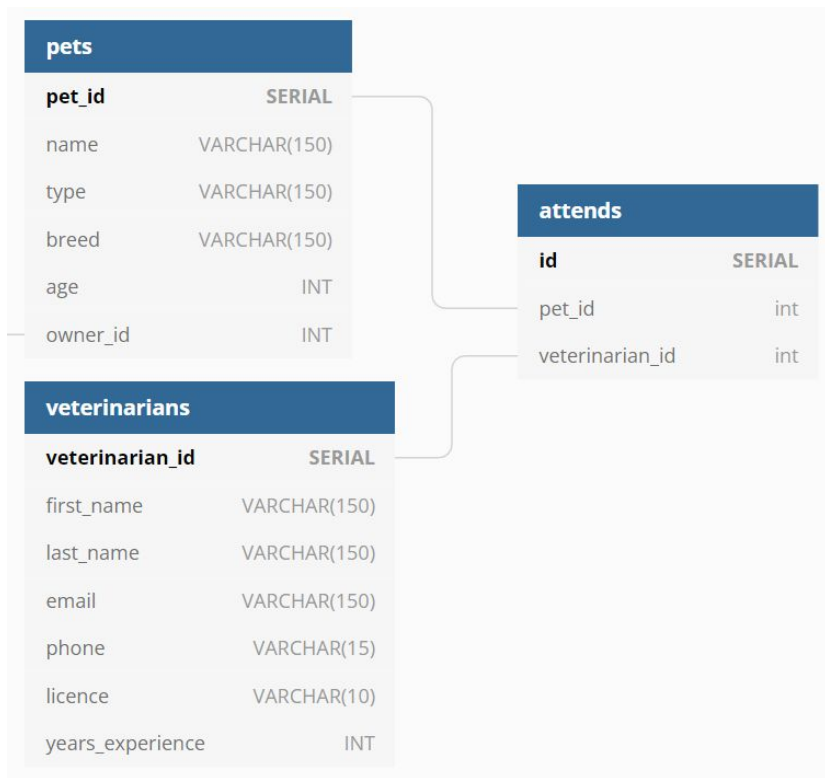


Tabla attends

```
CREATE TABLE attends (
```

```
    pet_id int NOT NULL,
```

```
    veterinarian_id int NOT NULL,
```

```
    FOREIGN KEY(pet_id) REFERENCES  
    pets(pet_id),
```

```
    FOREIGN KEY(veterinarian_id) REFERENCES  
    veterinarians(veterinarian_id),
```

```
    PRIMARY KEY(pet_id, veterinarian_id)
```

```
/* Composite Primary Key */
```

```
);
```

Composite Primary Key

La Composite Primary Key creada es un "hack" muy común para tablas que solo unen a 2 tablas.

Básicamente es la concatenación de 2 o más columnas para asegurar que una llave sea única.

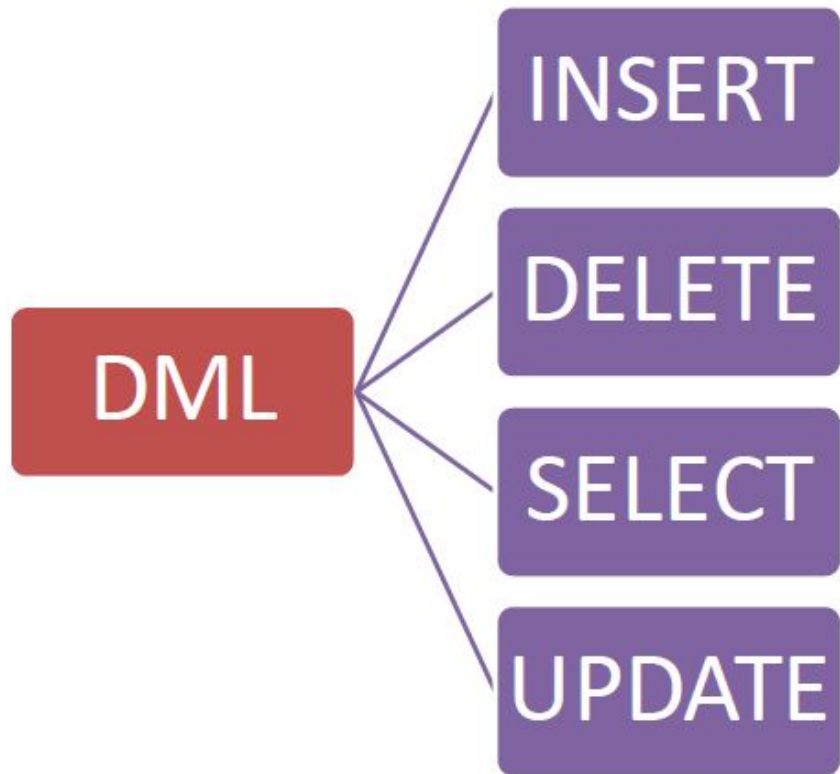
+Info:

<https://kb.objectrocket.com/postgresql/postgresql-composite-primary-keys-629>

Creando y Actualizando Información

DEV.F
DESARROLLAMOS(PERSONAS);

dev



Lenguaje de Manipulación de Datos (DML)

Utilizando instrucciones de SQL, permite a los usuarios introducir datos para posteriormente realizar tareas de consultas o modificación de los datos que contienen las Bases de Datos.

INSERT: con esta instrucción podemos insertar los valores en una base de datos.

UPDATE: sirve para modificar los valores de uno o varios registros.

SELECT: esta sentencia se utiliza para realizar consultas sobre los datos.

DELETE: se utiliza para eliminar las filas de una tabla.

SQL Insert Values Into a Table

Syntax - `INSERT INTO table_name
(column1_name, column2_name,...)
VALUES (column1_value, column2_value, ...)`

owners	
owner_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)

```
INSERT INTO owners (first_name, last_name,  
email, phone)  
VALUES  
( 'César', 'Guerra', 'cesar@correo.com', '55533  
34444' );
```

INSERT INTO

Con esta instrucción podemos insertar los valores en una base de datos.

Deberemos después de esta instrucción indicar el nombre de la tabla que afectaremos, y entre paréntesis los nombres de los atributos de la tabla que se afectarán.

Seguido de la instrucción VALUES qué indicará el contenido a colocar en cada atributo especificado anteriormente.

Nota: El orden de lo declarado en INSERT INTO se corresponde con lo colocado en VALUES.

SQL Insert Values Into a Table

Syntax - `INSERT INTO table_name
(column1_name, column2_name,...)
VALUES (column1_value, column2_value, ...)`

owners	
owner_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)

INSERT INTO owners
VALUES

(2, 'César', 'Guerra', 'cesar@correo2.com', '55
53334444')

INSERT INTO corto

Si se van a utilizar todos los valores, es posible omitir la parte de declarar los campos y colocar directamente los valores.

Sin embargo deberemos colocarlos en el mismo orden en que aparecen en la tabla y debe usarse con precaución, ya que al no declarar los nombres de los atributos podemos caer en errores.

SINTAXIS

```
DELETE FROM table_name  
WHERE condition;
```

owners	
owner_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)

```
DELETE FROM owners  
WHERE owner_id = 1
```

DELETE FROM

Delete nos permite borrar uno o más registros en una tabla donde se cumpla determinada condición.

Por seguridad, normalmente borramos poniendo como condición el ID de la tabla (para asegurarnos que solo borre específicamente un campo)

SINTAXIS

```
UPDATE table_name  
SET column1 = value1, column2 = value2,  
...  
WHERE condition;
```

owners	
owner_id	SERIAL
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(150)
phone	VARCHAR(15)

```
UPDATE owners  
SET first_name = 'Juan'  
WHERE owner_id = 1
```

UPDATE

Update permite actualizar la información contenida en una o más filas de la base de datos.



PostgreSQL

Ejercicios #2

Diseño SQL, Create, Insert

DEV.F.
DESARROLLAMOS(PERSONAS);

Control de Inventario

Un negocio quiere llevar el control de piezas que emplean en el día a día y los proveedores que suministran estas con las siguientes características:

- De cada proveedor conocemos su nombre, dirección, ciudad, provincia y un código de proveedor que será único para cada uno de ellos.
- Nos interesa llevar un control de las piezas que nos suministra cada proveedor.
- Es importante conocer la cantidad de las diferentes piezas que nos suministra y en qué fecha lo hace.
- Tenga en cuenta que un mismo proveedor nos puede suministrar una pieza con el mismo código en diferentes fechas.
- El diseño de la base de datos debe permitir almacenar un histórico con todas las fechas y las cantidades que nos ha proporcionado un proveedor.
- Una misma pieza puede ser suministrada por diferentes proveedores.
- De cada pieza conocemos un código que será único, nombre, color, precio y categoría.
- Pueden existir varias categorías y para cada categoría hay un nombre y un código de categoría único.
- Una pieza solo puede pertenecer a una categoría.

Realizar el modelo entidad-relación, documentar las queries para crear las tablas e insertar registros de ejemplo en PostgreSQL y ejecutarlas.

Concesionaria de Automóviles

Se desea diseñar una base de datos que sea de utilidad para concesionarios de automóviles. Un concesionario puede vender automóviles de varias marcas (por ejemplo, Audi y Volkswagen). Sobre los automóviles se desea mantener la siguiente información: marca, modelo, precio, descuento (si es que lo tiene) y los datos técnicos (caballos de fuerza, cilindrada, etc.). Para cada modelo de automóvil se quiere conocer las características de su equipamiento de serie (por ejemplo: airbag conductor y cierre centralizado), así como los extras que se pueden incluir (aire acondicionado, airbag acompañante, pintura metalizada, etc.) y el precio de cada uno de ellos. Notar que, lo que son características del equipamiento de serie de algunos modelos, son extras para otros modelos. Por ejemplo, hay modelos que llevan el airbag de serie mientras que otros lo tienen como un posible extra.

El concesionario tiene siempre automóviles de varios modelos en stock (cada uno se identifica por su número de bastidor). Éstos se pueden encontrar en su mismo local, o bien, en cualquiera de los servicios oficiales que dependen de él. Un servicio oficial es también una tienda de automóviles, pero depende de un concesionario que es el que le presta los automóviles para su exposición, y también se los vende. De cada servicio oficial se conoce el nombre, domicilio y RFC (número de identificación fiscal).

Cuando se vende un automóvil se quiere saber quién lo ha vendido: puede ser uno de los vendedores del concesionario o bien un servicio oficial. También se desea saber el precio que se ha cobrado por él y el modo de pago: al contado o mediante financiera. También se guardará información sobre los extras que se han incluido, precio de cada uno, la fecha de entrega, matrícula y si era de stock o se ha tenido que encargar a fábrica. De los vendedores se almacenarán los datos personales (nombre, INE, domicilio, etc.) y las ventas realizadas.

Realizar el modelo entidad-relación, documentar las queries para crear las tablas e insertar registros de ejemplo en PostgreSQL y ejecutarlas.



¡GRACIAS!



César Guerra



www.cesarguerra.mx