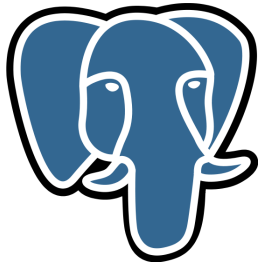
# SQL

KEEPCODING
Tech School

# Herramientas de trabajo

# PostgreSQL



- Es un SGBD relacional como puede ser Sql Server u Oracle, pero sin coste y de código libre
- Se puede instalar en Linux, macOS, Windows y Solaris

https://www.postgresql.org

# Dbeaver Community

- Es un cliente para conectarse a la mayoría de servidores de base de datos como PostgreSQL, MySQL, SQL Server, Oracle, DB2, Sybase, Access, SQLite etc.
- Es Libre y multiplataforma (Linux, macOS y Windows)

https://dbeaver.io

# Conexión a Postgree

- Conectamos con el cliente dbeaver a la bbdd
- Entorno de trabajo dbeaver
  - Conexiones
  - Editor SQL

# SQL

# ¿Que es SQL?

- Definición: Structure Query Languaje
- Lenguaje para el manejo de los SGBD como Oracle, Sql Server, Posgree, MySQL etc.
- Permite crear los modelos del diseño ER. A este tipo de SQL se le denomina DDL (Lenguaje de definición de datos)
    - Ejemplo crear una tabla, definir la Primary Key, FK, Integridad referencial (relaciones entre tablas).
- Permite añadir datos, actualizar, eliminar y buscar. Este tipo de SQL se denomina DML (Lenguaje de manipulación de datos)
    - Permite hacer búsquedas de los datos que tenemos
    - Permite realizar inserciones, actualizaciones y eliminaciones de los datos de las tablas

# ¿Es SQL igual en todos los SGBD?

- Son iguales en  más de un 95%
- Cada SGBD tiene algunas variaciones sobre todo las privadas como Oracle y SQL Server
- SQL Estándar es igual en todos los SGBD

# SQL - DDL

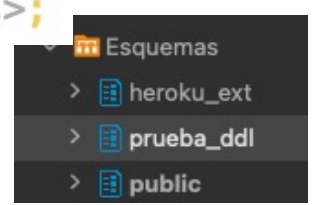

 

# Crear un esquema

Dentro de una base de datos se pueden organizar las tablas por esquemas según muchos criterios, dependiendo del sistema, tamaño de la bbdd, según el DBA, funcionalidad etc.

En nuestro caso vamos a crear un esquema denominado “pruebaDDL”

```
CREATE SCHEMA prueba_ddl AUTHORIZATION <<Usuario de base de datos>;
```



# Crear tabla – introducción

Creación de las tablas tal y como las hemos definido en el modelo ER

El comando SQL es :

CREATE TABLE <nombre tabla> (

);

```
create table test(
    col1 <tipo dato>,
    col2 <tipo dato>
);
```

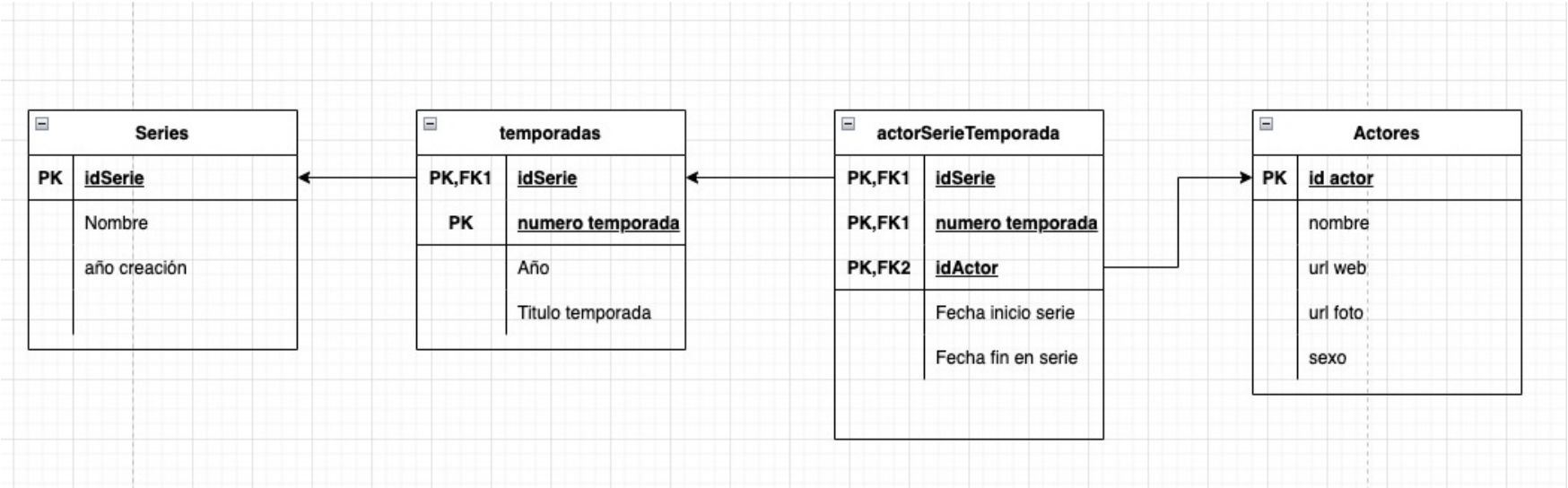
# Crear tabla – tipos de datos

Ver esta web

- Cadena:
  - varchar(n): Longitud variable de n caracteres
  - text: cadena caracteres de longitud variable ilimitada
- Fechas → date : fecha de un calendario (año, mes , dia)
- Números
  - integer : Numero entero normal
  - numeric(p,s): Numero exacto de enteros P y numero de decimales S.
  - Serial: para crear un autonumérico

# Crear tabla - Comando

Vamos a crear las tablas del modelo de datos creado el primer día



# Borrar un objeto

Para borrar una tabla o cualquier tipo de objeto en una base de datos usamos el comando “DROP

```
drop table series;
```

```
DROP SCHEMA prueba_ddl;
```

Nota: Se borra la estructura , en este caso la tabla y los datos. Es un comando que hay que tener mucho cuidado al lanzarlo.

# Crear tabla – control de nulos

Hay que indicar si los campos son nulos o no y se ponen después del tipo de dato de una columna

- NOT NULL (no puede ser nulo)
- NULL (permite campo nulo sin valor)

Los campos / columnas que sean PK no pueden ser nunca NULL

Nota: NULL es un concepto de base de datos que indica que un valor de una columna para un registro o fila, está vacío

# Crear tabla – Clave primaria

Para definir una clave primaria, se define un tipo de constraint o restricción de la tabla.

Se define al final de las columnas en el formato:

constraint <id clave primaria único > primary key (<columna que es PK>)

Nota: Se recomienda poner al identificador de la clave primaria el nombre de la tabla y después concatenado “_PK”

# Modificar Tabla

Se puede añadir/eliminar una columna nueva a una tabla con el comando “ALTER TABLE”.

Hay que tener en cuenta que si se añade una columna nueva  y la tabla ya tiene datos no puede crearse not null. Después habrá que rellenar los datos de la nueva columna en caso de que existan datos en la tabla.

Al eliminar una columna se eliminan los datos de esa columna.

Nota: estas acciones no permiten volver atrás en caso de error, por lo que antes de nada hacer una copia de seguridad de la tabla o base de datos.

Documentación alter table

# Modificar Tabla

También se puede usar el alter table para añadir las claves primarias y FK, después de crear las tablas.

```
create table series(
 idserie varchar(10) not null,
 nombre varchar(200) not null,
 anio_creation integer not null
 );

alter table series
 add constraint series_PK primary key(idserie)
```

=

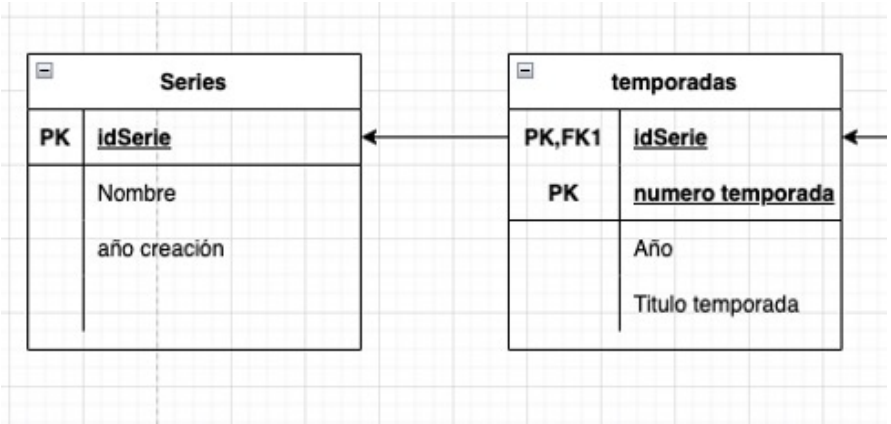
```
create table series(
idserie varchar(10) not null,
nombre varchar(200) not null,
anio_creation integer not null,
constraint series_PK primary key(idserie)
);
```

# Claves primarias multiples

Para definir una clave primaria multiple se hace igual que cuando es una columna, pero añadiendo las columnas por comas.

Todas las columnas deben ser not null.



# Integridad referencial: FK

En la tabla de temporadas hay que añadir la integridad referencial para que al añadir el idserie, se verifique que exista en la tabla de series. Para ello vamos a añadir una restricción de clave foránea.

```
alter table <table>
add constraint <id FK>  foreign key(<columna tabla>)
references <table referenciar>(<columna>)
```

Si se quiere activar la eliminación en cascada se añade al final “on delete cascade”

```
alter table <table>
add constraint <id FK>  foreign key(<columna tabla>)
references <table referenciar>(<columna>) on delete
cascade
```

# SQL - DML

# DML – Lenguaje manipulación de datos

Esta parte del lenguaje SQL nos permite manipular los datos de las estructuras de datos creadas (tablas).

Los comandos que vamos a ver son:

- INSERT
- SELECT
- UPDATE
- DELETE

# Comando INSERT

Permite añadir datos a una tabla, respetando el diseño del modelo creado.

Data la tabla series, la sentencia SQL para añadir un registro sería:

```
INSERT INTO prueba_ddl.series
(idserie, nombre, anio_creation)
VALUES('', '', 0);
```

Donde entre paréntesis se le indica los campos de la tabla y en VALUES los valores en el mismo orden que el paréntesis.

# Comando SELECT

Me permite buscar y recuperar los datos de 1 o N tablas mediante una sentencia SQL SELECT

Sintaxis básica:

SELECT <DISTINCT> <columnas> <*>

FROM <tabla> <alias>

WHERE <condicion1>  <operadores lógicos AND/OR> <condición 2> …

ORDER BY <columna> ASC/DESC

Operadores lógicos:

- And = significa “y”
- Or = significa “o”
- Not significa “no”
- () Paréntesis de prioridades para agrupar clausulas

# Comando DELETE

Me permite buscar y eliminar los registros que cumplan el criterio de la condición WHERE. Sino se pone condición se borrará toda la tabla (ojo peligro)

Sintaxis básica:

DELETE

FROM <tabla>

WHERE <condiciones>

# Comando UPDATE

Me permite buscar los registros que cumplan el criterio de la condición WHERE y actualizar las columnas que queramos con los valores indicador. Sino se pone condición se actualizará toda la tabla (ojo peligro)

Sintaxis básica:

UPDATE  <tabla>

SET <columna> = <valor>, <columna> = <valor> …

WHERE <condiciones>

# Comando SELECT. - Agrupaciones

Nos permite agrupar datos de una tabla por ejemplo para calcular el numero de registros, el valor máximo y mínimo etc,

Se trata de una sentencia SQL SELECT pero añadiendo una nueva condición denominada GROUP

SELECT …. FROM.. WHERE…

**GROUP BY <columna1>,<columna2>…**

**HAVING <condiciones>**

ORDER BY …

NOTA: las columnas de agruación deben estar en la selección. El having es una condición WHERE de los datos ya agrupados. (filtros por agrupaciones): El having solo puede filtrar por los campos de SELECT.

# Comando SELECT – UNION ALL

Nos permite lanzar varias sentencias SQL SELECT y el resultado de las Sentencias vengan en un único resultado. Las condiciones para que funcione es:

- Mismo numero de campos en cada sentencia y el mismo tipo de dato
- Se pueden traer datos de la misma tabla o diferentes tablas

```
select idserie , titulo  from temporadas  where idserie  = '0001'
union all
select idserie , titulo  from temporadas  where idserie  = '0002'
```

```
select distinct idserie  from temporadas  where idserie  = '0001'
 union all
 select idserie From series s  where idserie  in ('0002','0003')
```

# Comando CREATE TABLE xx AS

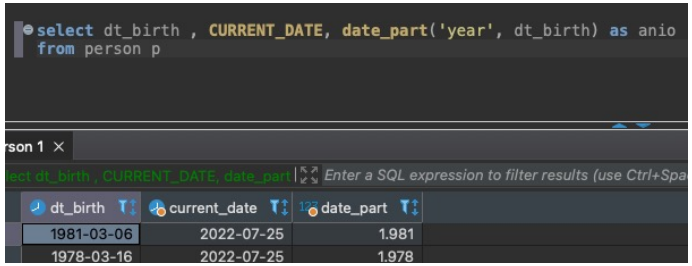
Nos permite crear una tabla  que necesitamos para procesos intermedios, cálculos intermedios, generación de informes y un largo etc., a partir de una sentencia SQL.

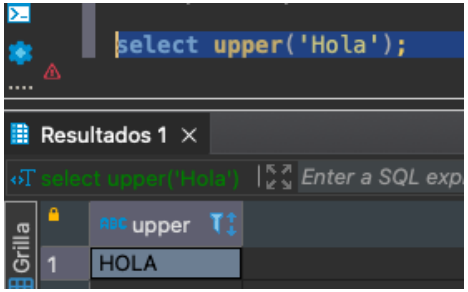
```
create table prueba_ddl.temportadas_aux1
as
select idserie , titulo  from temporadas  where idserie  = '0002'
```

# FUNCIONES SQL

Funciones que tiene SQL que ayudan a formatear información para extraerla, filtrarla, conversiones, manejo cadena  etc. mejor ver documentaciones por internet porque hay muchas.

Ver funciones de fecha  Ver funciones cadena



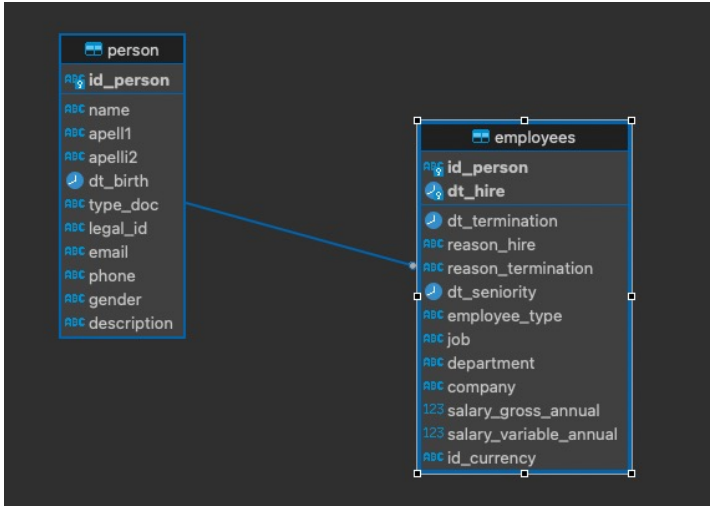



# EJERCICIOS SQL

# Modelo de Empleados en public

- Revisamos el modelo de persona
- Revisamos el modelo de empleados
- Revisamos el Script de creación de ambas tablas y carga de datos.



# Ejercicios 1

¿Sentencia SQL que calcule el total de personas que hay?

```sql
select  count(*)
from person
```

¿Sentencia SQL que calcule el numero de personas por cada genero (campo gender)?

```sql
select  count(*), gender
from person
group by gender
```

¿Cuántas personas hay mayores cuya fecha de nacimiento sea menor del 01/01/1970?

```sql
select  count(*)
from person
where dt_birth  < '1970-01-01'
```

Nota: Para filtrar fechas se puede poner como una cadena en formato ‘yyyy-mm-dd’

# Ejercicios 2

```sql
select  count(*)
from employees
where employee_type = 'PROFESIONAL'
```

¿Cuántos empleados de tipo PROFESIONAL” hay en la tabla employees?

¿Calcula el numero de empleados por cada tipo de empleado?

¿Calcula la media de los salario bruto anual(salary_gross_annual) de todos los empleados?

Calcula la media del salario bruto anual por departamento. Y también por tipo empleado , departamento y puesto. Ordenado todo por tipo empleado, departamento y puesto

Tipo Empleado = columna employee_type

# KEEPCODING

## Tech School

Madrid | Barcelona | Bogotá

**Jose Luis Bustos Esteban**
**bejl@outlook.es**