keep coding

Desarrollo web Full Stack

Modelado de datos y SQL

Días 2 y 3 – SQL

SQL
Introducción

DDL
Esquemas
Creación de tablas
Modificación de tablas
Borrado de tablas

DML
Inserción de datos
Extracción de datos
Actualización de datos
Borrado de datos

¿Qué es SQL?

SQL es un lenguaje utilizado para manipular la información dentro de bases de datos relacionales

Se divide en dos sub-lenguajes, como hemos visto anteriormente:

- DDL o Lenguaje de definición de datos
- DML o Lenguaje de manipulación de datos

El lenguaje es parcialmente estándar entre todos los SGBD, pero puede tener variaciones menores, sobre todo en funciones de agregado y tipos de datos.

En resumen, con DDL vamos a crear la estructura que ya planificamos con el modelo Entidad/Relación y utilizaremos DML para cargar, recuperar, actualizar y eliminar la información

SGBD - PostgreSQL

Para la clase de hoy vamos a utilizar como SGBD PostgreSQL.

Surge como parte del proyecto POSTGRES iniciado en 1986 en la Universidad de California, con más de 35 años de desarrollo a sus espaldas.

Tiene gran apoyo de la comunidad al ser un proyecto open source, funciona en la mayoría de los sistemas operativos y puede extenderse a través de Add-Ons com PostGIS para convertirla en una base de datos geoespacial.

Además, está disponible en cualquier entorno cloud como AWS o Google Cloud.

¿Por qué PostgreSQL y no otro SGBD? PostgreSQL es una navaja suiza y es aplicable en un amplio número de situaciones. En resumen, ofrece mejor rendimiento en operaciones de lectura/escritura que MySQL, pero peor rendimiento en operaciones de solo lectura, según AWS.

Cliente de SGBD

Un cliente **SGBD** es un software que nos permite acceder a una base de datos y utilizar sus prestaciones con una interfaz gráfica.

Para que nos entendamos, es un editor de código enfocado a bases de datos.

En esta clase utilizaremos **DBeaver**, que puede ser utilizado en la mayoría de **SGBD** del mercado.

Es libre y multiplataforma (Linux, MacOS y Windows).

DBeaver NO es una base de datos

DDL I – Bases de datos y esquemas

Un servidor de **PostgreSQL** puede contener a su vez diferentes bases de datos para dar servicio a diferentes clientes.

A su vez, cada base de datos albergará uno o varios esquemas. Un esquema hace referencia al conjunto de tablas utilizado por una aplicación.

El esquema por defecto de cada base de datos es **public**.

Este tipo de división es útil para limitar el acceso en diferentes roles de usuario. Nosotros no vamos a ver usuarios y roles en este curso, pero al menos vamos a crear un esquema para trabajar que sea distinto de **public**.

Para crear un esquema, se utiliza la sintaxis:

```
CREATE SCHEMA if not exists <nombre del esquema>;
```

Y se activa con:

```
SET SCHEMA '<nombre del esquema>';
```

*Prestar atención a las comillas simples de la activación, que no están en la creación. **No** es una errata. Cosas de PostgreSQL

DDL II – Creación de tablas

Las entidades se representan en tablas. Para crear una tabla, se utiliza la instrucción:

CREATE TABLE <nombre de la tabla>

Además del nombre de la tabla se pueden especificar los campos con su nombre, tipo de datos y demás descripciones. Por ejemplo, la creación de la tabla contacto en **PostgreSQL** sería:

```
CREATE TABLE contacto (
    dni varchar(9) PRIMARY KEY,
    nombre text NOT NULL,
    fecha_nacimiento date NOT NULL,
    id_via int NOT NULL,
    ext text NULL,
    CONSTRAINT via_contacto_fk FOREIGN KEY (id_via) REFERENCES via(id)
);
```

Para crear las relaciones se utiliza:

CONSTRAINT <nombre de la relación> FOREIGN KEY (<nombre del campo>) **REFERENCES** <tabla>(<campo de la tabla>).

Los tipos de datos disponibles en **PostgreSQL** están disponibles en la *documentación oficial*.

DDL III – Modificación de tablas

Para modificar las columnas de una tabla, se utiliza

ALTER TABLE <nombre de la tabla>

y la modificación que se necesite hacer. Por ejemplo, añadir una columna a contacto:

```
ALTER TABLE contacto ADD IBAN TEXT;
```

O eliminarla:

```
ALTER TABLE contacto DROP COLUMN IBAN;
```

También podemos añadir en este momento nuevas constricciones (constraints), como claves foráneas:

```
alter table TABLA add constraint NOMBRE_RESTRICCION
foreign key (CAMPO_TABLA) references TABLA_RELACIONADA(CAMPO_TABLA_RELACIONADA);
```

DDL IV – Borrado de tablas

Para eliminar **permanentemente** una tabla, se utiliza el comando DROP TABLE <nombre de la tabla>.

Este proceso no se puede deshacer, así que utilizar con mucho cuidado.

```
DROP TABLE contacto;
```

DML I – Inserción de datos

Para insertar datos en las tablas se utiliza el comando INSERT INTO <nombre de la tabla> (columna1, columna2, …, columnaN) y una colección con los valores a insertar:

- Con VALUES (valor_columna1, valor_columna2, …, valor_columnaN)
- Como resultado de un SELECT (lo veremos más adelante)

```
INSERT INTO telefono (id_contacto, telefono) VALUES (1, 666111333);
```

Se pueden insertar, por tanto, varias líneas de golpe (de hecho, es más eficiente). Para hacer esto, pueden añadirse varios bloques entre paréntesis separados por comas:

INSERT INTO <nombre-de-tabla> (atributo_1, atributo_2,…,atributo_N) VALUES (x1, x2,…,xN), (y1, y2,…,yN)…;

DML II – Extracción de datos I

Para obtener los datos de las tablas se utiliza la instrucción

SELECT columna1, columna2, ..., columnaN <nombre de la tabla>:

```
SELECT id_contacto, telefono FROM telefono;
```

Además, se puede filtrar con WHERE y ordenar con ORDER BY:

```
SELECT id_contacto, telefono FROM telefono WHERE id_contacto = 1 ORDER BY telefono;
```

Se pueden encadenar condiciones en la parte de WHERE con:

- And
- Or
- Not

Y se pueden agrupar condiciones entre paréntesis

DML III – Extracción de datos II

Al obtener datos con SELECT, también podemos agrupar por valores coincidentes con GROUP BY y además utilizar funciones de agregación, como COUNT(), MAX(), MIN(), AVG()…etc

Por ejemplo, obtener cuantos números de teléfono distintos tiene un contacto:

```
SELECT id_contacto, COUNT(telefono) FROM telefono GROUP BY id_contacto;
```

También existen funciones que se pueden añadir como columnas al extraer de un SELECT. Se puede personalizar también el nombre de la columna al extraer información con AS (incluso en muchas ocasiones sin ni siquiera AS):

```
SELECT DATETIME('now') fecha_y_hora;
```

Puedes consultar más funciones de agregación, funciones útiles para trabajar con texto y con fechas.

DML IV – Actualización de datos

Para actualizar los datos se utiliza la instrucción:

UPDATE <nombre de la tabla> SET columna1='valor columna1', ..., columnaN='valor columnaN'
WHERE <condiciones como en SELECT>

```
UPDATE telefono SET telefono = '111222333' WHERE id_telefono = 1;
```

¡CUIDADO! UPDATE **actualizará todas** las líneas coincidentes con las condiciones WHERE. ¡Tenlo en cuenta!

DML V – Borrado de datos

Por último, para borrar información se utiliza:

DELETE FROM <tabla> WHERE <condiciones>:

```
DELETE FROM telefono WHERE id_telefono = 1;
```

¡CUIDADO! **DELETE borrará todas** las líneas coincidentes con las condiciones WHERE.Y funciona sin WHERE, destruyendo la tabla completa ¡Tenlo en cuenta!

Al borrar datos que impliquen tablas relacionadas, puede establecerse el tipo de borrado al crear la unión entre tablas (CONSTRAINT):

- CASCADE (En cascada): Borra los registros completos también de las tablas relacionadas
- RESTRICT (Restricción): No deja borrar.
- SET NULL (Déjalo en blanco): Deja en blanco el dato en la tabla relacionada
- SET DEFAULT (Fija el valor por defecto): Permite dejar un valor por defecto en lugar de Nulo

DML VI – Unión de tablas

Para extraer datos de varias tablas, se utiliza JOIN. Existen diferentes tipos de unión, siendo los más comunes:

- **INNER JOIN**: Devuelve solamente los elementos coincidentes en las dos tablas.
- **LEFT/RIGHT JOIN**: Devuelve los elementos coincidentes en las dos tablas y la tabla completa del lado que especifiques, derecha o izquierda.

Puedes ver todos en este enlace:

https://upload.wikimedia.org/wikipedia/commons/9/9d/SQL_Joins.svg

www.keepcoding.io cursos@keepcoding.io (+34) 916 33 1779