Cheatsheet for MySQL

Primeros pasos

Instalar Lamp Server

- `sudo apt update`
- `sudo apt upgrade`
- `sudo apt install lamp-serverˆ` Instala el stack Linux, Apache, MySQL, PHP/Perl/Python

http://127.0.0.1 / http://localhost

Configurar usuario principal en MySQL

- `sudo mysql -u root` y posteriormente `mysql -u root -p`
- `ALTER USER 'root'@'localhost' IDENTIFIED WITH BY 'new-password';`

O se crea un nuevo usuario:

- `CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';`
- `GRANT ALL ON *.* TO 'usuario'@'localhost';`

Phpmyadmin

`sudo apt install phpmyadmin`

Acceso: http://localhost/phpmyadmin

Trabajando con CLI

Comandos SQL esenciales para trabajar desde CLI:

- mysql: desde terminal, ejecuta el cliente CLI de MySQL.
- SHOW: Muestra información de algo determinado:
- databases
- tables
- grants
- USE: Selecciona una base de datos.

- DESCRIBE: muestra la estructura de una tabla.
- Acceso al cliente CLI: `mysql -u nombre_usuario -p` e ingresamos contraseña.
- mostrar bases de datos disponibles: `show databases;`
- seleccionar una base de datos: `use nombre_base;`
- mostrar tablas disponibles: `show tables;`
- mostrar permisos de un usuario: `show grants for 'nuevo_usuario'@'localhost';`
- Ver la estructura de la tabla `describe user_type`

DDL (Data Definition Language)

Permite definir las estructuras que almacenan los datos:

- CREATE: Creación de bases de datos, tablas, usuarios...

```
CREATE DATABASE name;

CREATE USER 'nombre_usuario' IDENTIFIED BY 'contraseña';

CREATE TABLE login(
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(32) NOT NULL,
    PRIMARY KEY(id)
);

CREATE TABLE users(
    id int NOT NULL AUTO_INCREMENT,
    name varchar(256) NOT NULL,
    username varchar(256) NOT NULL,
    login_id int,
    PRIMARY KEY(id),
    FOREIGN KEY (user) REFERENCES login(id)
);
```

- ALTER: Modificar la estructura. Una columna, una tabla o añadir una clave Foranea.

```
ALTER TABLE sales ADD date_of_purchase DATE;
```

```sql
ALTER TABLE sales DROP (COLUMN) date_of_purchase;

ALTER TABLE sales CHANGE name new_name VARCHAR(32);

ALTER TABLE sales MODIFY COLUMN name VARCHAR(32);

ALTER TABLE old_table RENAME new_table;

ALTER DATABASE mi_basededatos READ ONLY = 1

ALTER USER 'nombre_usuario'@'localhost' IDENTIFIED BY 'nueva clave';

ALTER TABLE offers ADD CONSTRAINT pieces_fk FOREIGN KEY (piece_id) REFERENCES pieces (id) ON
```

- DROP: Eliminar objetos.

```sql
DROP TABLE my_tabla;

DROP DATABASE my_database;

DROP USER 'nombre_usuario'@'localhost';
```

- TRUNCATE: Elimina los registros de la tabla, pero mantiene la estructura.

DML (Data Manipulation Language)

DML o Lenguaje de manipulación de datos se utiliza para gestionar datos dentro de las tablas:

- SELECT: Obtener registros de una tabla.
- INSERT: Insertar registros en la tabla.
- UPDATE: Modifica datos existentes dentro de la tabla.
- DELETE: Elimina todos los registros de una tabla pero sin eliminar los espacios asignados.

SELECT

- Seleccionar todos los campos: `SELECT * FROM nombre_tabla;`
- Seleccionar algunos campos: `SELECT nombre, apellidos FROM nombre_tabla;`
- Ordenar los resultados por orden ascendente o descendente: `SELECT * FROM nommbre_tabla ORDER BY id DESC;`
- Mostrar una cantidad concreta de resultados: `SELECT * FROM nombre_tabla ORDER BY id ASC LIMIT 1;`
- Filtrar la consulta para encontrar un valor igual: `SELECT * FROM nombre_tabla WHERE id = 1;`
- Filtrar la consulta para encontrar valores diferente: `SELECT * FROM nombre_tabla WHERE id <> 1;`
- Filtrar la consulta para encontrar un valor menor o igual: `SELECT * FROM nombre_tabla WHERE id >= 1;`
- Filtrar la consulta si cumple una u otra condición: `SELECT * FROM nombre_tabla WHERE id >= 1 OR nombre = 'Pedro';`
- Filtar la consulta si cumple varias condiciones: `SELECT * FROM nombre_tabla WHERE id >= 1 AND nombre = 'Pedro';`
- Buscar coincidencias que contengan un texto: `SELECT * FROM user_type WHERE nombre LIKE '%pep%';`

SELECT tablas relacionadas con JOIN

- Consultar dos tablas unidas por su clave foranea incluyendo solo matches conectadas: `SELECT * FROM nombre_tabla INNER JOIN otra_tabla ON otra_tabla.id = nombre_tabla.estado;`
- Todas las filas conectadas y todas las filas sueltas de la tabla de la izquierda (primera). Valores null en las celdas de la otra tabla. `SELECT * FROM nombre_tabla LEFT JOIN otra_tabla ON otra_tabla.id = nombre_tabla.estado;`
- Todas las conectadas + las sueltas de la tabla de la derecha: `SELECT * FROM nombre_tabla RIGHT JOIN otra_tabla ON otra_tabla.id = nombre_tabla.estado;`
- Todos las conectadas y no conectadas. No en mysql. `SELECT * FROM nombre_tabla FULL JOIN otra_tabla ON otra_tabla.id = nombre_tabla.estado;`

```
SELECT name, last_name, brand, price FROM clients left join bikes on bikes.id = clients.bike
```

Contador de registros

- Devolver un valor número con la cantidad de registros: `SELECT COUNT(*) FROM nombre_tabla;`

Alias para columnas

- Asignar un alias a una columna: `SELECT id as identificador, nombre as como_me_llamo FROM nombre_tabla;`

Subqueries Ejecutar consultas dentro de otras consultas:

```
SELECT nombre, apellidos FROM usuarios WHERE id IN
    (SELECT usuario
        FROM
            registros
        WHERE
            id = 10);
```

INSERT

```
INSERT INTO nombre_tabla SET id=NULL, nombre='Alfredo';
```

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

All columns

```
INSERT INTO nombre_tabla values("value1", "value2");
```

UPDATE

- Actualizar datos:

```
UPDATE nombre_tabla SET nombre = 'Antonio' WHERE id = 1;
```

DELETE

- borrar todos los datos de una tabla:

```
DELETE FROM nombre_tabla;
```

- borrar un dato:

```
DELETE FROM nombre_tabla WHERE id=1;
```

Borra también relaciones en cascada en ambos casos si la restricción está configurada así.

Comandos empleados en clase

```
INSERT INTO bikes (model, brand, type, price)
            values("ECaliber", "Trek", "Road", 9999);

INSERT INTO repairs VALUES(NULL, 3, 3, 170, 4);

UPDATE clients SET phone_number="+34 3231231" WHERE id=1;

SELECT COUNT(*) FROM bikes WHERE type="Mountain";

SELECT MAX(price), name FROM parts;

SELECT * FROM parts WHERE
    price = (SELECT MIN(price) FROM parts);

DELETE FROM parts WHERE id IN (1, 2);

SELECT parts.name FROM parts
    JOIN repairs ON repairs.part_id = parts.id
    JOIN clients ON repairs.client_id = clients.id
    WHERE clients.id = 1;

SELECT model, parts.name, discount FROM bikes
    JOIN clients ON clients.bike_id = bikes.id
    JOIN repairs ON clients.id = repairs.client_id
    JOIN parts ON parts.id = repairs.part_id
    JOIN offers ON offers.part_id = parts.id;
```