Ejercicio 4:

sql

CREATE TABLE productos (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(255),

categoria VARCHAR(100),

precio DECIMAL(10,2)

);

-- Insertar 100,000 registros de ejemplo (puede hacerse con un script o generador de datos)

Sql

INSERT INTO productos (nombre, categoria, precio)

VALUES ('Iphone', 'Electronica', 1290.00);

-- (repetido o generado automáticamente hasta llegar a 100.000 filas)

> Consulta sin indice:

-- Ejecutar EXPLAIN para ver el plan de ejecución

sql

EXPLAIN SELECT \* FROM productos WHERE categoria = 'Electronica';

OBSERVACIÓN: Sin índice, recorre toda la tabla , esto genera un alto costo de lectura cuando son muchos datos.

CREAMOS UN ÍNDICE:

sql

CREATE INDEX idx\_categoria ON productos(categoria);

Consultamos con el índice:  
-- Volver a ejecutar el EXPLAIN

sql

EXPLAIN SELECT \* FROM productos WHERE categoria = 'Electronica';

OBSERVACIÓN:

Al usar el índice (idx\_categoria) mejora el tiempo de búsqueda

La creación de índices en campos utilizados frecuentemente en cláusulas WHERE mejora significativamente el desempeño de consultas en bases de datos grandes.

**Ejercicio 8: Seguridad y Auditoría**

**Armamos la tabla:**

-- Crear tabla clientes

sql

CREATE TABLE clientes (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(255),

email VARCHAR(255)

);

-- Crear tabla de auditoría

sql

CREATE TABLE auditoria\_clientes (

id INT AUTO\_INCREMENT PRIMARY KEY,

cliente\_id INT,

accion VARCHAR(50),

fecha TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

OBS: Agregamos trigger para la auditoría ( Cada vez que se inserta, actualiza o borra un cliente, se registra automáticamente en la tabla auditoria\_clientes.)

-- Trigger para INSERT

sql

CREATE TRIGGER auditoria\_insert

AFTER INSERT ON clientes

FOR EACH ROW

INSERT INTO auditoria\_clientes (cliente\_id, accion)

VALUES (NEW.id, 'INSERT');

-- Trigger para registrar UPDATES

sql

CREATE TRIGGER auditoria\_update

AFTER UPDATE ON clientes

FOR EACH ROW

INSERT INTO auditoria\_clientes (cliente\_id, accion)

VALUES (NEW.id, 'UPDATE');

-- Trigger para DELETE/borrar

sql

CREATE TRIGGER auditoria\_delete

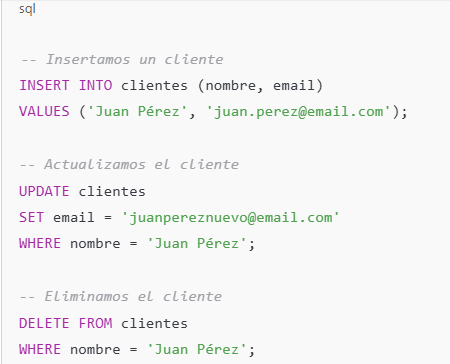
AFTER DELETE ON clientes

FOR EACH ROW

INSERT INTO auditoria\_clientes (cliente\_id, accion)

VALUES (OLD.id, 'DELETE');

Ejemplo de uso:



Conclusión:

Los triggers nos permiten registrar automáticamente las modificaciones sin necesidad de intervención manual. Esto es fundamental para mantener la **trazabilidad** y **seguridad** de la información en bases de datos críticas.

También se recomienda aplicar también restricciones o políticas de acceso sobre las tablas de auditoría para evitar manipulaciones no deseadas.

Ejercicio 9: Backup y Restore

Creación del backup en MySQL :

bash

mysqldump -u usuario -p basededatos > backup.sql

**Explicación de cada parte:**

* mysqldump: herramienta de línea de comandos para exportar bases de datos.
* -u usuario: nombre del usuario de MySQL que tiene permisos de lectura.
* -p: solicita la contraseña del usuario.
* basededatos: nombre de la base de datos a respaldar.
* >: redirige la salida al archivo backup.sql.

En resumen esto: Crea un archivo llamado backup.sql que contiene todas las instrucciones SQL necesarias para recrear la base de datos.

Simulamos perdida de datos :

sql

DROP DATABASE basededatos;

Restauramos de la base de datos:

Primero, se debe volver a crear la base de datos.

Bash

mysql -u usuario -p

Una vez dentro ejecutamos lo siguiente:

sql

CREATE DATABASE basededatos;

EXIT;

Observación: Si la base ya existiera, debería eliminarse antes (DROP DATABASE basededatos;).

Restaurar el contenido del backup

Bash

mysql -u usuario -p basededatos < backup.sql

Acá lo que hacemos es conectarnos a MySQL con el usuario especificado, le indicamos que la base de datos destino es basededatos y cargamos el contenido de backup.sql en esa base.

La base de datos basededatos es recreada con todas sus tablas, registros, índices y relaciones.