

## PARTE 1

### CREACIÓN DE ROLES Y ASIGNACIÓN DE PRIVILEGIOS

Escribe un script SQL que cree los cinco usuarios

Asegúrate de agregar comentarios que expliquen qué puede hacer cada usuario.

Usa contraseñas seguras y personalízalas si es necesario en la creación de usuarios.

- Inicia sesión con un usuario que tenga privilegios de super administrador (por ejemplo, root).

- CREAR ROLES

- Super Administrador: Crear y eliminar bases de datos.

```
-- Crear usuario Super Administrador
CREATE USER 'superadmin'@'localhost' IDENTIFIED BY 'SuperSecurePassword1!';
GRANT CREATE, DROP ON *.* TO 'superadmin'@'localhost';
-- Super Administrador puede crear y eliminar bases de datos.
```

- Administrador: Crear usuarios y procesos.

```
-- Crear usuario Administrador
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'AdminSecurePassword2!';
GRANT CREATE USER, PROCESS ON *.* TO 'admin'@'localhost';
-- Administrador puede crear usuarios y procesos.
```

- CRUD: Insertar, actualizar y eliminar datos.

```
-- Crear usuario CRUD
CREATE USER 'crud_user'@'localhost' IDENTIFIED BY 'CrudSecurePassword3!';
GRANT INSERT, UPDATE, DELETE ON *.* TO 'crud_user'@'localhost';
-- CRUD puede insertar, actualizar y eliminar datos.
```

- CRU: Insertar y actualizar, pero sin eliminar.

```
-- Crear usuario CRU
CREATE USER 'cru_user'@'localhost' IDENTIFIED BY 'CruSecurePassword4!';
GRANT INSERT, UPDATE ON *.* TO 'cru_user'@'localhost';
-- CRU puede insertar y actualizar, pero no eliminar.
```

- Solo Lectura: Realizar consultas a las tablas.

```
-- Crear usuario Solo Lectura
CREATE USER 'readonly_user'@'localhost' IDENTIFIED BY 'ReadOnlySecurePassword5!';
GRANT SELECT ON *.* TO 'readonly_user'@'localhost';
-- Solo Lectura puede realizar consultas a las tablas.
```

## Verificación de Permisos

Usa el comando `SHOW GRANTS FOR 'usuario'@'localhost';` para verificar qué permisos tiene cada usuario.

```
-- Verificación de permisos
SHOW GRANTS FOR 'superadmin'@'localhost';
SHOW GRANTS FOR 'admin'@'localhost';
SHOW GRANTS FOR 'crud_user'@'localhost';
SHOW GRANTS FOR 'cru_user'@'localhost';
SHOW GRANTS FOR 'readonly_user'@'localhost';
```

31 • `SHOW GRANTS FOR 'superadmin'@'localhost';`

<

Result Grid | Filter Rows:  | Export: | Wrap

Grants for superadmin@localhost
GRANT CREATE, DROP ON *.* TO `superadmin`...

32 • `SHOW GRANTS FOR 'admin'@'localhost';`

<

Result Grid | Filter Rows:  | Export: | \

Grants for admin@localhost
GRANT PROCESS, CREATE USER ON *.* TO `a`...

33 • `SHOW GRANTS FOR 'crud_user'@'localhost';`

<

Result Grid | Filter Rows:  | Export: | Wrap C

Grants for crud_user@localhost
GRANT INSERT, UPDATE, DELETE ON *.* TO `c`...

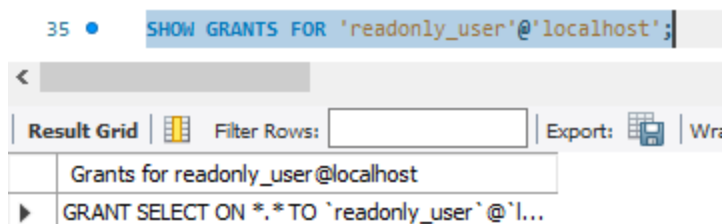
34 • `SHOW GRANTS FOR 'cru_user'@'localhost';`

<

Result Grid | Filter Rows:  | Export: | W

Grants for cru_user@localhost
GRANT INSERT, UPDATE ON *.* TO `cru_user`...

```
35 • SHOW GRANTS FOR 'readonly_user'@'localhost';
```



Grants for readonly_user@localhost
GRANT SELECT ON *.* TO 'readonly_user'@'localhost'...

## PARTE 2 TRIGGERS

Los triggers, también conocidos como disparadores, son procedimientos almacenados que se ejecutan automáticamente en respuesta a ciertos eventos en una base de datos. Estos eventos pueden incluir operaciones como INSERT, UPDATE o DELETE en una tabla. Los triggers se definen en una tabla específica y permiten automatizar acciones, garantizar la integridad de los datos y aplicar reglas de negocio sin la necesidad de que el cliente o la aplicación lo controle manualmente. Un trigger se ejecuta como parte de la transacción que provoca el evento, lo que garantiza que la acción se realice de manera sincronizada con el cambio de datos.

Existen tres tipos principales de triggers según su momento de ejecución:

**BEFORE:** Se ejecutan antes de que se realice la operación en la base de datos. Son útiles para validar datos o aplicar modificaciones antes de que el cambio tenga lugar.

**AFTER:** Se ejecutan después de que la operación se haya completado. Son empleados para registrar cambios, auditar acciones o realizar operaciones dependientes de datos actualizados.

**INSTEAD OF:** Estos triggers reemplazan la acción que se habría realizado, y son utilizados principalmente en vistas para personalizar el comportamiento de operaciones como INSERT, UPDATE o DELETE.

Los triggers pueden activarse mediante tres eventos principales:

**INSERT:** Cuando se agrega un nuevo registro a una tabla.

**UPDATE:** Cuando se modifica un registro existente.

**DELETE:** Cuando se elimina un registro de una tabla.

El contexto del trigger está definido por las palabras clave NEW y OLD:

NEW: Representa los valores nuevos en una fila durante una operación de tipo INSERT o UPDATE.

OLD: Representa los valores previos en una fila durante una operación de tipo UPDATE o DELETE.

El uso de triggers en una base de datos es fundamental para garantizar la integridad y la automatización de procesos. Entre sus principales beneficios se encuentran:

**Automatización de Tareas:** Los triggers pueden realizar tareas repetitivas automáticamente, como la actualización de registros relacionados o el registro de auditorías.

**Integridad Referencial:** Ayudan a mantener la consistencia entre tablas relacionadas al aplicar reglas automáticamente.

**Control de Cambios:** Permiten monitorear y registrar cualquier modificación realizada en las tablas, lo que resulta útil en auditorías o seguimiento de datos.

**Aplicación de Reglas de Negocio:** Aseguran que las reglas definidas por el negocio se cumplan de manera consistente.

#### Ventajas y Desventajas

##### Ventajas:

**Automatización:** Eliminan la necesidad de programación manual para tareas específicas.

**Reducción de Errores:** Aseguran que las reglas y validaciones se apliquen de manera uniforme.

**Auditoría y Seguimiento:** Facilitan la creación de registros históricos de los cambios realizados en la base de datos.

**Seguridad:** Pueden prevenir operaciones no autorizadas o inconsistentes.

##### Desventajas:

**Complejidad:** La configuración de triggers puede hacer que el sistema sea más difícil de entender y depurar.

Impacto en el Rendimiento: Si no se diseñan correctamente, los triggers pueden ralentizar las operaciones de la base de datos debido a la sobrecarga de procesos.

Dependencia de la Base de Datos: Los triggers son específicos de la base de datos y dificultan la migración o el cambio de tecnología.

Áreas Comunes de Aplicación:

Validación de Datos: Verificar que los datos cumplan con ciertas reglas antes de ser insertados o actualizados.

Auditoría: Registrar automáticamente los cambios realizados en los datos.

Seguimiento de Cambios: Mantener un historial detallado de las modificaciones realizadas en una tabla.

Reglas de Negocio: Implementar reglas automáticas que deben cumplirse en toda operación.

Actualización de Tablas Relacionadas: Sincronizar datos entre tablas relacionadas.

Casos de Uso Específicos:

En empresas financieras, los triggers se utilizan para registrar automáticamente las transacciones realizadas por los usuarios.

En sistemas de inventario, se activan para actualizar la cantidad de productos disponibles después de cada venta o devolución.

En sistemas de gestión de recursos humanos, registran automáticamente las modificaciones realizadas en los datos de los empleados, como aumentos salariales o cambios de departamento.

## Enunciado de la Práctica:

### Objetivo:

Crear un **trigger** que registre todas las operaciones (insert, update, delete) realizadas en una tabla de empleados en una tabla de auditoría. El objetivo es llevar un historial detallado de las acciones realizadas, incluyendo el tipo de operación, los datos afectados y la fecha y hora en que ocurrió cada cambio.

### Descripción del Ejercicio:

Imagina que tienes una empresa que desea llevar un control detallado sobre los cambios realizados en los registros de sus empleados. Para ello, se necesita crear una tabla de auditoría que registre cualquier acción (inserción, actualización o eliminación) que se realice en la tabla de **Empleados**. Cada vez que se realice una operación sobre la tabla de empleados, el sistema debe registrar la siguiente información en la tabla de auditoría:

Tipo de operación realizada (INSERT, UPDATE, DELETE)

ID del empleado afectado

Nombre y departamento del empleado

Salario del empleado

Fecha y hora en que se realizó la operación

### Pasos para la práctica:

#### 1. **Crear la tabla de empleados** con los siguientes campos:

- `EmpID` (ID del empleado)
- `Nombre` (Nombre del empleado)
- `Departamento` (Departamento en el que trabaja el empleado)
- `Salario` (Salario del empleado)

```
-- Parte 2
> CREATE TABLE empleados (
    EmpID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Departamento VARCHAR(100) NOT NULL,
    Salario DECIMAL(10,2) NOT NULL
);
```

2. **Crear la tabla de auditoría** con los siguientes campos:

- AudID (ID del registro de auditoría)
- Accion (Tipo de operación: INSERT, UPDATE, DELETE)
- EmpID (ID del empleado afectado)
- Nombre (Nombre del empleado)
- Departamento (Departamento del empleado)
- Salario (Salario del empleado)
- Fecha (Fecha y hora de la operación)

```
> CREATE TABLE auditoria (
    AudID INT AUTO_INCREMENT PRIMARY KEY,
    Accion ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,
    EmpID INT,
    Nombre VARCHAR(100),
    Departamento VARCHAR(100),
    Salario DECIMAL(10,2),
    Fecha DATETIME NOT NULL
);
```

### 3. Crear el trigger:

- El trigger debe activarse **después** de realizar cualquier operación (INSERT, UPDATE o DELETE) sobre la tabla de empleados. El trigger debe insertar un nuevo registro en la tabla de auditoría cada vez que se realice una de estas operaciones.

### Requerimientos:

- El trigger debe registrar correctamente el tipo de operación realizada (INSERT, UPDATE, DELETE).
- El trigger debe almacenar el nombre del empleado, su departamento y salario.
- El trigger debe capturar la fecha y hora de la operación.
- Crear el trigger para auditar eliminaciones Y Ver los cambios realizados

### Verificar el funcionamiento

```

/*Verificar el funcionamiento*/
-- Inserción
INSERT INTO empleados (Nombre, Departamento, Salario) VALUES ('Juan Pérez', 'Ventas', 2500.00);

-- Actualización
UPDATE empleados SET Salario = 3000.00 WHERE EmpID = 1;

-- Eliminación
DELETE FROM empleados WHERE EmpID = 1;

104      -- Verificar auditoría
105  ●    SELECT * FROM auditoria;
106

```

[illegible]



Link del repositorio:

[https://github.com/Sebastian-Betancourt/Aplicacion\\_usuarios.git](https://github.com/Sebastian-Betancourt/Aplicacion_usuarios.git)