PARTE 1

CREACIÓN DE ROLES Y ASIGNACIÓN DE PRIVILEGIOS

Creación del Script SQL

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.

```
CREATE USER 'superadministrador'@'localhost' IDENTIFIED BY 'superadmin123';
GRANT CREATE, DROP ON *.* TO 'superadministrador'@'localhost' WITH GRANT OPTION;
```

  Este usuario es capaz de crear bases de datos o eliminarlas en todo el servidor

- Administrador: Crear usuarios y procesos.

```
CREATE USER 'admin_usuario'@'localhost' IDENTIFIED BY 'admin123';
GRANT CREATE USER, PROCESS ON *.* TO 'admin_usuario'@'localhost';
```

  Este usuario es capaz de crear usuarios y gestionar los procesos del sistema

- CRUD: Insertar, actualizar y eliminar datos.

```
CREATE USER 'crud_usuario'@'localhost' IDENTIFIED BY 'crud123';
GRANT SELECT, INSERT, DELETE, UPDATE ON *.* TO 'crud_usuario'@'localhost';
```

  El usuario tiene acceso completo a las opciones CRUD en la base de datos

- CRU: Insertar y actualizar, pero sin eliminar.

```
CREATE USER 'cru_usuario'@'localhost' IDENTIFIED BY 'cru123';
GRANT INSERT, UPDATE ON *.* TO 'cru_usuario'@'localhost';
```

  El usuario tiene permisos para actualizar e insertar, pero no eliminarlos

- Solo Lectura: Realizar consultas a las tablas.

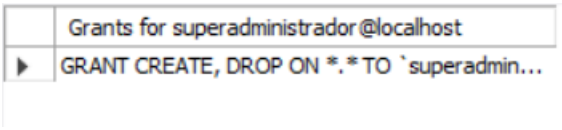
```
CREATE USER 'lectura_usuario'@'localhost' IDENTIFIED BY 'lectura123';
GRANT SELECT ON *.* TO 'lectura_usuario'@'localhost';
```

  El usuario tiene acceso a solo realizar consultas en la base de datos

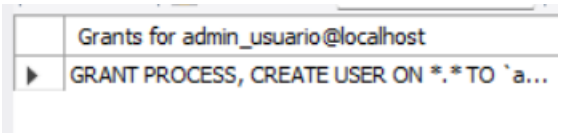
Verificación de Permisos

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

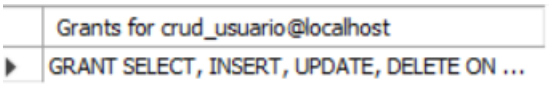
```
SHOW GRANTS FOR 'superadministrador'@'localhost';
SHOW GRANTS FOR 'admin_usuario'@'localhost';
SHOW GRANTS FOR 'crud_usuario'@'localhost';
SHOW GRANTS FOR 'cru_usuario'@'localhost';
SHOW GRANTS FOR 'lectura_usuario'@'localhost';
```



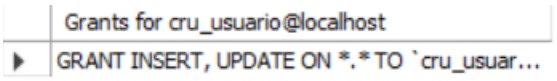
Grants for superadministrador@localhost
GRANT CREATE, DROP ON *.* TO `superadmin...



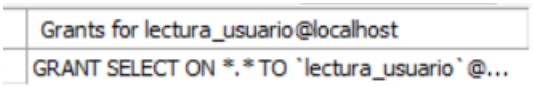
Grants for admin_usuario@localhost
GRANT PROCESS, CREATE USER ON *.* TO `a...



Grants for crud_usuario@localhost
GRANT SELECT, INSERT, UPDATE, DELETE ON ...



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



Grants for lectura_usuario@localhost
GRANT SELECT ON *.* TO `lectura_usuario`@...

PARTE 2 TRIGGERS

Objetivo:

Comprender la importancia de los *triggers* en bases de datos, cómo se aplican en diferentes escenarios, y reconocer áreas en las que su uso es crucial para la integridad de los datos y el control de los procesos.

Instrucciones:

1. Investigación sobre los *Triggers*:

Conceptos clave sobre los Triggers

2. **Tipos de triggers**:

- **BEFORE: Se ejecuta antes de que se realice una acción en la base de datos (INSERT, UPDATE, DELETE). Son útiles para validar datos antes de que se**

efectúe el cambio en la tabla.

- **AFTER**: Se ejecuta **después** de que la acción se haya realizado (INSERT, UPDATE, DELETE). Es útil cuando quieres registrar cambios o ejecutar acciones adicionales después de que se haya completado una operación.
- **INSTEAD OF**: Se utiliza principalmente en vistas, reemplazando la acción que se habría realizado por otra. Por ejemplo, en una vista, puedes usar un trigger `INSTEAD OF` para manejar `INSERT`, `UPDATE` o `DELETE` en lugar de modificar directamente las vistas.

3. **Eventos que pueden activar un trigger**:
    - **INSERT**: Cuando se agrega un nuevo registro a la tabla.
    - **UPDATE**: Cuando se modifica un registro existente en la tabla.
    - **DELETE**: Cuando se elimina un registro de la tabla.
4. **Contexto de los triggers**:
    - **NEW**: En triggers de tipo `INSERT` o `UPDATE`, puedes utilizar la palabra clave `NEW` para hacer referencia a los valores que van a insertarse o actualizarse en una fila. Es decir, los nuevos valores de una columna.
    - **OLD**: En triggers de tipo `DELETE` o `UPDATE`, puedes utilizar la palabra clave `OLD` para hacer referencia a los valores anteriores de una fila antes de la modificación o eliminación.

AMPLIAR INFORMACIÓN Y ENTENDER

- Definición y funcionamiento: Investigar qué son los *triggers* en bases de datos, cómo funcionan, y los diferentes tipos de *triggers* (por ejemplo, `BEFORE`, `AFTER`, `INSTEAD OF`).

  Permite ejecutar automáticamente un conjunto de instrucciones en respuesta a un evento específico que ocurre e una tabla o una vista. Primero se define qué evento activará el trigger, como un INSER, UPDATE o DELETE, luego se debe especificar si se debe ejecutar antes o después usando BEFORE o AFTER. Existen los triggers: BEFORE que se ejecuta antes que el evento ocurra, también está AFTER que se ejecuta después que el evento ocurra, e INSTEAD OF,

este trigger en MySQL no está disponible, y se ejecuta en lugar del evento.

- Importancia de su uso: Explicar por qué es importante usar *triggers* en una base de datos y cuáles son los beneficios que aportan, tales como la automatización de tareas, la integridad referencial, el control de cambios, entre otros.

  Son importantes porque permiten automatizar tareas, garantizando la consistencia e integridad de datos sin tener que intervenir manualmente. Los triggers nos permiten ejecutar acciones de forma automática en respuesta a diferentes eventos, también refuerzan las relaciones entre tablas, registran automáticamente los cambios en los datos y ayudan en la implementación de políticas de seguridad dentro de la base de datos.

- Ventajas y desventajas: Reflexionar sobre las ventajas (por ejemplo, evitar la corrupción de datos, asegurar reglas de negocio) y desventajas (por ejemplo, sobrecarga en el rendimiento) de utilizar *triggers*.

  Las ventajas que presentan son:

  - Automatización de procesos
  - Mantienen la integridad de los datos
  - Registra quién y cuándo los cambios se realizaron

  Las desventajas son:

  - Sobrecarga en el rendimiento
  - Dificultades en el depurado
  - Mantenimiento complejo
  - Depende del servidor de base de datos

5. Aplicaciones de *Triggers*:
   - Áreas de aplicación: Identificar en qué áreas de una base de datos se aplican *triggers*. Ejemplos comunes incluyen la validación de datos, la auditoría, el seguimiento de cambios y la implementación de reglas de negocio automáticas.
     - Mantenimiento de la integridad referencial: Aseguran relaciones entre

tablas de forma consistente

- Automatización de procesos: Se realizan procesos automáticos
- Optimización de flujos de trabajo: Se coordinan múltiples acciones

- Casos de uso específicos: Investigar ejemplos reales de empresas o sistemas que utilicen *triggers* para gestionar procesos como auditorías de registros, actualizaciones automáticas de información, control de integridad referencial, entre otros.
    - Auditorías de registro: Se usan en bancos y finanzas
    - Actualizaciones automáticas de información: Usado en el comercio electrónico, por ejemplo: Amazon
    - Control de integridad referencial: Usando en sistemas de gestiones universitarias

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)
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)

```sql
CREATE DATABASE triggers_tarea;

USE triggers_tarea;

CREATE TABLE Empleados (
    EmpID INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(100),
    Departamento VARCHAR(50),
    Salario DECIMAL(10, 2)
);

CREATE TABLE Auditoria (
    AudID INT PRIMARY KEY AUTO_INCREMENT,
    Accion ENUM('INSERT', 'UPDATE', 'DELETE'),
    EmpID INT,
    Nombre VARCHAR(100),
    Departamento VARCHAR(50),
    Salario DECIMAL(10, 2),
    Fecha DATE,
    FOREIGN KEY (EmpID) REFERENCES Empleados(EmpID)
);
```

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.

```sql
-- TRIGGER PARA EL EVENTO INSERT

DELIMITER $$

CREATE TRIGGER Auditoria_Empleados_Insertar
AFTER INSERT ON Empleados
FOR EACH ROW
BEGIN
    INSERT INTO Auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
    VALUES ('INSERT', NEW.EmpID, NEW.Nombre, NEW.Departamento, NEW.Salario, NOW());
END $$

DELIMITER ;
```

```sql
-- TRIGGER PARA EL EVENTO UPDATE

DELIMITER $$

CREATE TRIGGER Auditoria_Empleados_Actualizar
AFTER UPDATE ON Empleados
FOR EACH ROW
BEGIN
    INSERT INTO Auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
    VALUES ('UPDATE', NEW.EmpID, NEW.Nombre, NEW.Departamento, NEW.Salario, NOW());
END $$

DELIMITER ;


-- TRIGGER PARA EL EVENTO DELETE

DELIMITER $$

CREATE TRIGGER Auditoria_Empleados_Eliminar
AFTER DELETE ON Empleados
FOR EACH ROW
BEGIN
    INSERT INTO Auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
    VALUES ('DELETE', OLD.EmpID, OLD.Nombre, OLD.Departamento, OLD.Salario, NOW());
END $$

DELIMITER ;
```

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

```
INSERT INTO Empleados(Nombre, Departamento, Salario)
VALUES ("Pedro Pérez", "Ventas", 3000.00),
("Jose García", "Compras", 3500.00),
("Luis Gonzáles", "Negocios", 5000.00);
```

```
UPDATE Empleados SET Salario = 4500.00 WHERE EmpID = 1;
UPDATE Empleados SET Salario = 4000.00 WHERE EmpID = 2;
UPDATE Empleados SET Salario = 3000.00 WHERE EmpID = 3;

DELETE FROM Empleados WHERE EmpID = 1;
DELETE FROM Empleados WHERE EmpID = 2;
DELETE FROM Empleados WHERE EmpId = 3;
```

```
82    SELECT * FROM Auditoria;
83
84
85
```

Result Grid | Filter Rows: | Edit: | Export/Import:

AudID	Accion	EmpID	Nombre	Departamento	Salario	Fecha
1	INSERT	1	Pedro Pérez	Ventas	3000.00	2024-12-31
2	INSERT	2	Jose García	Compras	3500.00	2024-12-31
3	INSERT	3	Luis Gonzáles	Negocios	5000.00	2024-12-31
4	UPDATE	1	Pedro Pérez	Ventas	4500.00	2024-12-31
5	UPDATE	2	Jose García	Compras	4000.00	2024-12-31
6	UPDATE	3	Luis Gonzáles	Negocios	3000.00	2024-12-31
7	DELETE	1	Pedro Pérez	Ventas	4500.00	2024-12-31
8	DELETE	2	Jose García	Compras	4000.00	2024-12-31
9	DELETE	3	Luis Gonzáles	Negocios	3000.00	2024-12-31

PRESENTACIÓN

Compartir el Enlace

- Comparte el enlace del repositorio de GitHub con el instructor o en la plataforma donde se solicite.

Formato de Entrega:

- Documento escrito con la investigación y el estudio de caso.