

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.
- Administrador: Crear usuarios y procesos.
- CRUD: Insertar, actualizar y eliminar datos.
- CRU: Insertar y actualizar, pero sin eliminar.
- Solo Lectura: Realizar consultas a las tablas.

```
1 • Create database Roles;
2 • use Roles;
3
4 • SHOW GRANTS FOR CURRENT_USER();
5
6 • CREATE USER 'super_admin'@'localhost' IDENTIFIED BY 'S3gur0P@ss_super';
7 • GRANT ALL PRIVILEGES ON *.* TO 'super_admin'@'localhost';
8 |
9
10 -- Crear un usuario Administrador para gestionar usuarios y procesos.
11 • CREATE USER 'admin'@'localhost' IDENTIFIED BY 'S3gur0P@ss_admin';
12 • GRANT CREATE USER, PROCESS ON *.* TO 'admin'@'localhost';
13 -- El usuario 'admin' puede crear usuarios y gestionar procesos en el servidor.
14
15 -- Crear un usuario CRUD para realizar operaciones completas en tablas.
16 • CREATE USER 'crud_user'@'localhost' IDENTIFIED BY 'S3gur0P@ss_crud';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Grants for root@localhost
▶	GRANT SELECT, INSERT, UPDATE, DELETE, CR...
	GRANT APPLICATION_PASSWORD_ADMIN,AU...
	GRANT PROXY ON ``@`` TO 'root'@'localho...

```
30 -- Aplicar cambios para asegurarse de que los p
31 • FLUSH PRIVILEGES;
32 |
```

Output

Action Output

	#	Time	Action
✖	5	12:53:16	GRANT ALL PRIVILEGES ON *.* TO 'super_admin'@'localhost'
✖	6	12:53:37	GRANT ALL PRIVILEGES ON *.* TO 'super_admin'@'localhost'
✔	7	13:08:48	SHOW GRANTS FOR CURRENT_USER()
✔	8	16:29:30	FLUSH PRIVILEGES

Verificación de Permisos

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

- 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*.

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.
- 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.

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:

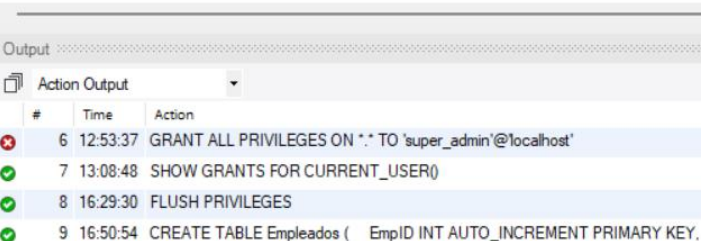
- Tipo de operación realizada (INSERT, UPDATE, DELETE)
- ID del empleado afectado
- Nombre y departamento del empleado
- Salario del empleado
- Fecha y hora en que se realizó la operación

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)

```
38 CREATE TABLE Empleados (  
39     EmpID INT AUTO_INCREMENT PRIMARY KEY,  
40     Nombre VARCHAR(100) NOT NULL,  
41     Departamento VARCHAR(100) NOT NULL,  
42     Salario DECIMAL(10, 2) NOT NULL  
43 );
```



#	Time	Action
6	12:53:37	GRANT ALL PRIVILEGES ON *.* TO 'super_admin'@'localhost'
7	13:08:48	SHOW GRANTS FOR CURRENT_USER()
8	16:29:30	FLUSH PRIVILEGES
9	16:50:54	CREATE TABLE Empleados (EmpID INT AUTO_INCREMENT PRIMARY KEY,

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)

```

46 CREATE TABLE Auditoria (
47     AudID INT AUTO_INCREMENT PRIMARY KEY,
48     Accion ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,
49     EmpID INT,
50     Nombre VARCHAR(100),
51     Departamento VARCHAR(100),
52     Salario DECIMAL(10, 2),
53     Fecha DATETIME DEFAULT CURRENT_TIMESTAMP
54 );

```

Output

Action Output

#	Time	Action
7	13:08:48	SHOW GRANTS FOR CURRENT_USER()
8	16:29:30	FLUSH PRIVILEGES
9	16:50:54	CREATE TABLE Empleados (EmpID INT AUTO_INCREMENT PRIMARY KE
10	16:51:57	CREATE TABLE Auditoria (AudID INT AUTO_INCREMENT PRIMARY KEY,

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.

Trigger para Insert

```

57 DELIMITER //
58 CREATE TRIGGER trg_auditoria_empleados_insert
59 AFTER INSERT ON Empleados
60 FOR EACH ROW
61 BEGIN
62     INSERT INTO Auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
63     VALUES ('INSERT', NEW.EmpID, NEW.Nombre, NEW.Departamento, NEW.Salario, NOW());
64 END//
65 DELIMITER ;

```

Output

Action Output

#	Time	Action
10	16:51:57	CREATE TABLE Auditoria (AudID INT AUTO_INCREMENT PRIMARY KEY, Accion ENUM('INSERT', 'UPD...
11	16:52:43	CREATE TRIGGER trg_auditoria_empleados AFTER INSERT OR UPDATE OR DELETE ON Empleados FOR EA...
12	16:53:34	CREATE TRIGGER trg_auditoria_empleados AFTER INSERT OR UPDATE OR DELETE ON Empleados FOR EA...
13	16:55:38	CREATE TRIGGER trg_auditoria_empleados_insert AFTER INSERT ON Empleados FOR EACH ROW BEGIN I...

Trigger para Update

```

67 DELIMITER //
68 • CREATE TRIGGER trg_auditoria_empleados_update
69 AFTER UPDATE ON Empleados
70 FOR EACH ROW
71 BEGIN
72     INSERT INTO Auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
73     VALUES ('UPDATE', NEW.EmpID, NEW.Nombre, NEW.Departamento, NEW.Salario, NOW());
74 END//
75 DELIMITER ;

```

Output

Action Output

#	Time	Action
✗ 11	16:52:43	CREATE TRIGGER trg_auditoria_empleados AFTER INSERT OR UPDATE OR DELETE ON Empleados FOR EA...
✗ 12	16:53:34	CREATE TRIGGER trg_auditoria_empleados AFTER INSERT OR UPDATE OR DELETE ON Empleados FOR EA...
✓ 13	16:55:38	CREATE TRIGGER trg_auditoria_empleados_insert AFTER INSERT ON Empleados FOR EACH ROW BEGIN I...
✓ 14	16:56:03	CREATE TRIGGER trg_auditoria_empleados_update AFTER UPDATE ON Empleados FOR EACH ROW BEGIN ...

Trigger para Delete

```

/b
77 DELIMITER //
78 • CREATE TRIGGER trg_auditoria_empleados_delete
79 AFTER DELETE ON Empleados
80 FOR EACH ROW
81 BEGIN
82     INSERT INTO Auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
83     VALUES ('DELETE', OLD.EmpID, OLD.Nombre, OLD.Departamento, OLD.Salario, NOW());
84 END//
85 DELIMITER ;

```

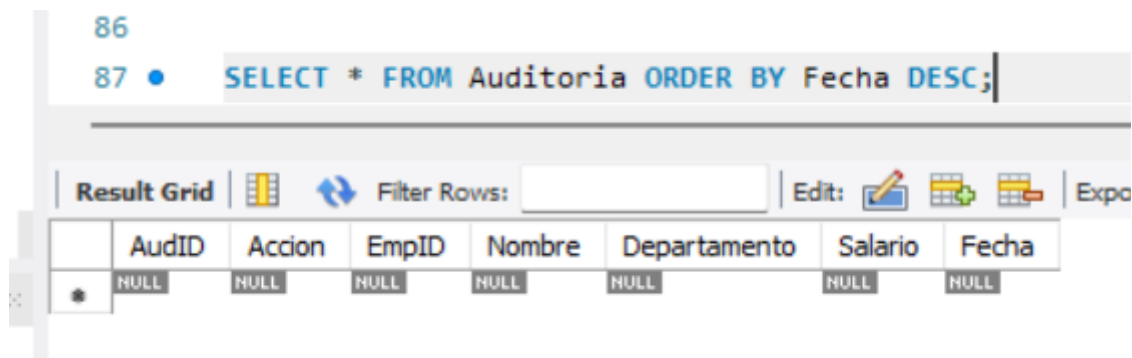
Output

Action Output

#	Time	Action	M
✗ 12	16:53:34	CREATE TRIGGER trg_auditoria_empleados AFTER INSERT OR UPDATE OR DELETE ON Empleados FOR EA...	Err
✓ 13	16:55:38	CREATE TRIGGER trg_auditoria_empleados_insert AFTER INSERT ON Empleados FOR EACH ROW BEGIN I...	0 n
✓ 14	16:56:03	CREATE TRIGGER trg_auditoria_empleados_update AFTER UPDATE ON Empleados FOR EACH ROW BEGIN ...	0 n
✓ 15	16:56:30	CREATE TRIGGER trg_auditoria_empleados_delete AFTER DELETE ON Empleados FOR EACH ROW BEGIN ...	0 n

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](#)



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.

Formato de entrega link de Github:

<https://github.com/EdwinSarango12/Roles-Tarea-BasesdeDatos-2024b.git>

Formato del documento en el aula virtual como PDF