

ESCUELA POLITECNICA NACIONAL

Nombre: Richard Padilla

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.

Verificación de Permisos

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



```
1  -- Richard Padilla
2  • CREATE USER 'SuperAdministrador'@'localhost' IDENTIFIED BY 'SuperAdmin_123!';
3  • CREATE USER 'Administrador'@'localhost' IDENTIFIED BY 'Admin_456!';
4  • CREATE USER 'CRUD'@'localhost' IDENTIFIED BY 'Crud_789!';
5  • CREATE USER 'CRU'@'localhost' IDENTIFIED BY 'Cru_101!';
6  • CREATE USER 'SoloLectura'@'localhost' IDENTIFIED BY 'ReadOnly_202!';
7
8  -- SuperAdministrador: Crear y eliminar bases de datos
9  • GRANT CREATE, DROP ON *.* TO 'SuperAdministrador'@'localhost';
10
11 -- Administrador: Crear usuarios y procesos
12 • GRANT CREATE USER, PROCESS ON *.* TO 'Administrador'@'localhost';
13
14 -- CRUD: Insertar, actualizar y eliminar datos
15 • GRANT INSERT, UPDATE, DELETE ON *.* TO 'CRUD'@'localhost';
16
17 -- CRU: Insertar y actualizar datos, pero sin eliminar
18 • GRANT INSERT, UPDATE ON *.* TO 'CRU'@'localhost';
19
20 -- SoloLectura: Realizar consultas a las tablas
21 • GRANT SELECT ON *.* TO 'SoloLectura'@'localhost';
22
23 -- Aplicar cambios
```

```

15 • GRANT INSERT, UPDATE, DELETE ON *.* TO 'CRUD'@'localhost';
16
17 -- CRU: Insertar y actualizar datos, pero sin eliminar
18 • GRANT INSERT, UPDATE ON *.* TO 'CRU'@'localhost';
19
20 -- SoloLectura: Realizar consultas a las tablas
21 • GRANT SELECT ON *.* TO 'SoloLectura'@'localhost';
22
23 -- Aplicar cambios
24 • FLUSH PRIVILEGES;
25
26 -- Verificar permisos para cada usuario
27 • SHOW GRANTS FOR 'SuperAdministrador'@'localhost';
28 • SHOW GRANTS FOR 'Administrador'@'localhost';
29 • SHOW GRANTS FOR 'CRUD'@'localhost';
30 • SHOW GRANTS FOR 'CRU'@'localhost';
31 • SHOW GRANTS FOR 'SoloLectura'@'localhost';
32

```

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. ¿Qué es un Trigger?

Un Trigger (o disparador) es un conjunto de instrucciones que se ejecuta automáticamente en respuesta a ciertos eventos en una base de datos. Los triggers se asocian a tablas o vistas y se activan cuando ocurre un evento específico, como una inserción, actualización o eliminación de datos.

2. Tipos de triggers:

- a. **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.
- o **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.
- o **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:

- a. **INSERT:** Cuando se agrega un nuevo registro a la tabla.
- o **UPDATE:** Cuando se modifica un registro existente en la tabla.
- o **DELETE:** Cuando se elimina un registro de la tabla.

4. Contexto de los triggers:

- a. **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.
- o **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

1. Funcionamiento:

Los triggers se activan según las siguientes condiciones:

2. Eventos:

- **INSERT:** Cuando se añade un nuevo registro.
- **UPDATE:** Cuando se actualizan datos existentes.
- **DELETE:** Cuando se eliminan registros.

3. Momentos:

- **BEFORE:** Se ejecuta antes de que ocurra el evento. Se utiliza para validar o modificar los datos antes de que se apliquen.
- **AFTER:** Se ejecuta después de que el evento haya tenido lugar. Ideal para auditorías o registros de cambios.
- **INSTEAD OF:** Sustituye la acción que activa el evento. Común en vistas.

Tipos de Triggers:

- **BEFORE TRIGGERS:** Útiles para verificar restricciones o modificar datos antes de la operación.
- **AFTER TRIGGERS:** Ideales para registrar cambios o notificar eventos después de la operación.
- **INSTEAD OF TRIGGERS:** Permiten modificar la acción predeterminada, especialmente útil en vistas.

3. Ventajas y Desventajas

Ventajas:

1. **Automatización:** Reducen la carga de trabajo manual.
2. **Asegurar reglas de negocio:** Implementan validaciones directamente en la base de datos.
3. **Auditoría:** Proveen un historial de cambios en las tablas.

4. **Integridad referencial:** Garantizan la consistencia de los datos entre tablas relacionadas.

Desventajas:

1. **Sobrecarga en el rendimiento:** Los triggers mal diseñados pueden ralentizar las operaciones.
2. **Complejidad:** El uso excesivo puede hacer que el sistema sea difícil de entender y mantener.
3. **Comportamiento oculto:** La lógica de los triggers puede no ser evidente, lo que complica el rastreo de errores.
4. **Ciclos infinitos:** Pueden causar bucles si no se diseñan cuidadosamente.

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:

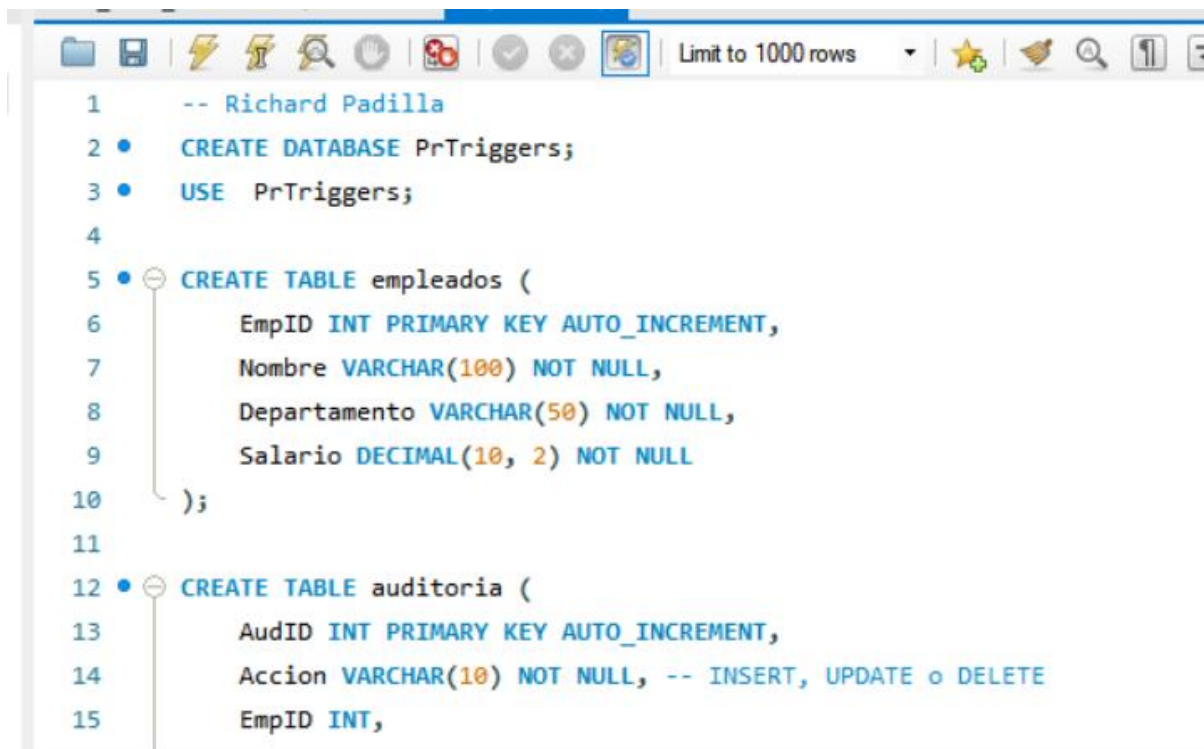
- `AudID` (ID del registro de auditoría)
- `Accion` (Tipo de operación: INSERT, UPDATE, DELETE)
- `EmpID` (ID del empleado afectado)
- `Nombre` (Nombre del empleado)
- `Departamento` (Departamento del empleado)
- `Salario` (Salario del empleado)
- `Fecha` (Fecha y hora de la operación)

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](#)



```
1  -- Richard Padilla
2  • CREATE DATABASE PrTriggers;
3  • USE PrTriggers;
4
5  • CREATE TABLE empleados (
6      EmpID INT PRIMARY KEY AUTO_INCREMENT,
7      Nombre VARCHAR(100) NOT NULL,
8      Departamento VARCHAR(50) NOT NULL,
9      Salario DECIMAL(10, 2) NOT NULL
10 );
11
12 • CREATE TABLE auditoria (
13     AudID INT PRIMARY KEY AUTO_INCREMENT,
14     Accion VARCHAR(10) NOT NULL, -- INSERT, UPDATE o DELETE
15     EmpID INT,
```




```
22  -- Para insertar
23  DELIMITER $$
24
25  • CREATE TRIGGER trigger_auditoria_insert
26    AFTER INSERT ON empleados
27    FOR EACH ROW
28    BEGIN
29        INSERT INTO auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
30        VALUES ('INSERT', NEW.EmpID, NEW.Nombre, NEW.Departamento, NEW.Salario, NOW());
31    END$$
32
33  DELIMITER ;
34
35  -- Para actualizar
36  DELIMITER $$
```

```
34
35  -- Para actualizar
36  DELIMITER $$
37
38  • CREATE TRIGGER trigger_auditoria_update
39    AFTER UPDATE ON empleados
40    FOR EACH ROW
41    BEGIN
42        INSERT INTO auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
43        VALUES ('UPDATE', NEW.EmpID, NEW.Nombre, NEW.Departamento, NEW.Salario, NOW());
44    END$$
45
46  DELIMITER ;
47
48  -- Para eliminar
```

```

58
59 DELIMITER ;
60
61 • -- Insertar empleados
62 INSERT INTO empleados (Nombre, Departamento, Salario)
63 VALUES ('Juan Pérez', 'Finanzas', 5000.00);
64
65 -- aCTUALIZAR SALARIO DE LOS EMPLEADOS
66 • UPDATE empleados
67 SET Salario = 5500.00
68 WHERE EmpID = 1;
69
70 -- Eliminar un empleado
71 • DELETE FROM empleados
72 WHERE EmpID = 1;

```

SCHEMAS

Filter objects

- funciones_sql
- prtriggers
 - Tables
 - Views
 - Stored Procedures
 - Functions
- registros
- sys
- tienda_online

Limit to 1000 rows

```

62 INSERT INTO empleados (Nombre, Departamento, Salario)
63 VALUES ('Juan Pérez', 'Finanzas', 5000.00);
64
65 -- aCTUALIZAR SALARIO DE LOS EMPLEADOS
66 • UPDATE empleados
67 SET Salario = 5500.00
68 WHERE EmpID = 1;
69
70 -- Eliminar un empleado
71 • DELETE FROM empleados
72 WHERE EmpID = 1;
73
74
75 • SELECT * FROM auditoria;
76

```

Result Grid

AudID	Accion	EmpID	Nombre	Departamento	Salario	Fecha
1	INSERT	1	Juan Pérez	Finanzas	5000.00	2025-01-03 18:31:57
2	UPDATE	1	Juan Pérez	Finanzas	5500.00	2025-01-03 18:31:57
3	DELETE	1	Juan Pérez	Finanzas	5500.00	2025-01-03 18:31:57
	NULL	NULL	NULL	NULL	NULL	NULL

Administration Schemas

Information

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.