ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

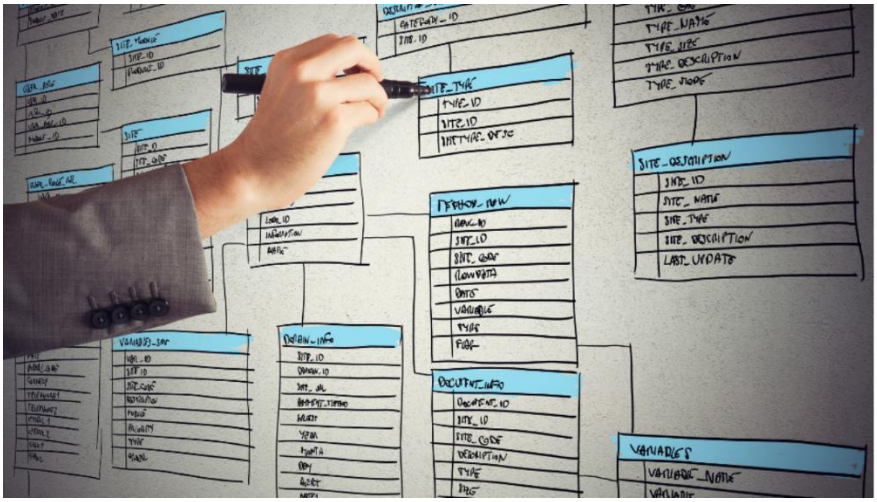
BASE DE DATOS

PROFESOR:	Ing. Yadira Franco R
PERÍODO ACADÉMICO:	2024-B

TAREA

TÍTULO:

INVESTIGACIÓN Y PRACTICA



Estudiante

Cristian Tambaco

2024-B

PARTE 1

CREACIÓN DE ROLES Y ASIGNACIÓN DE PRIVILEGIOS

Creación del Script SQL

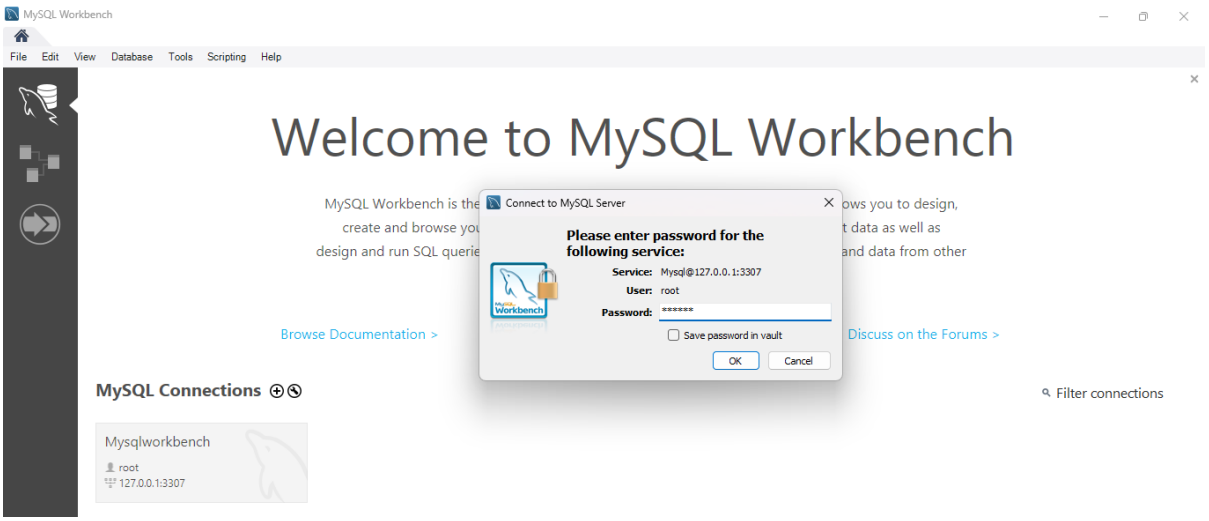
Enlace Git Hub:

https://github.com/CristianTambaco/Deber_ROLES---TRIGGER.git

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`).

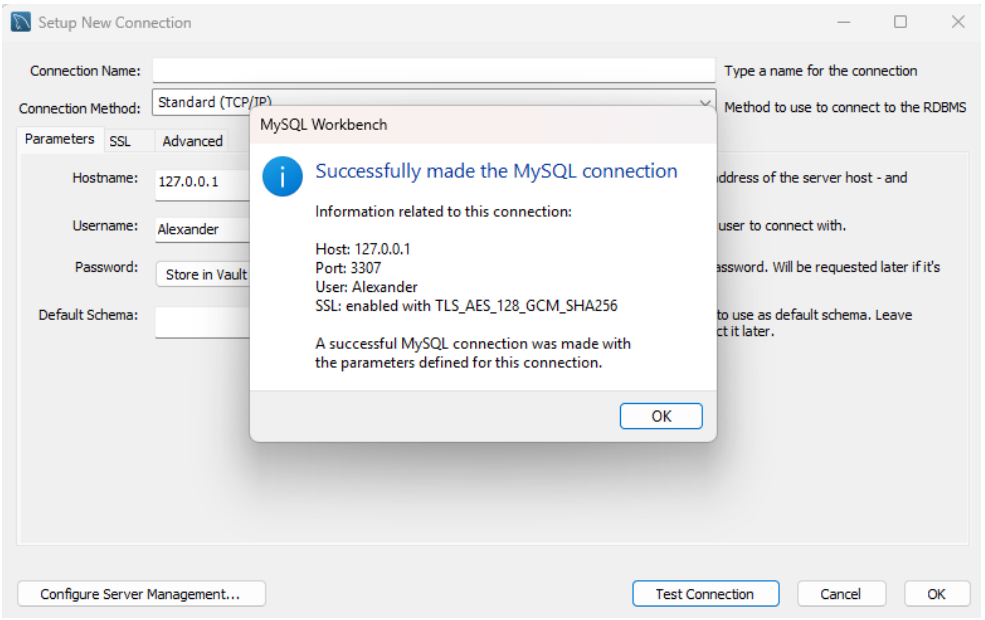


- CREAR ROLES
- Super Administrador: Crear y eliminar bases de datos.

-- CREAR ROLES
-- 1. Super Administrador: Crear y eliminar bases de datos.

-- Crear usuario con el rol 'Super Administrador'
CREATE USER 'Alexander'@'localhost' IDENTIFIED BY 'superadmin123';

-- Asignar rol 'Super_Administrador', donde puede crear y eliminar bases de datos
GRANT CREATE, DROP ON *.* TO 'Alexander'@'localhost';
FLUSH PRIVILEGES;

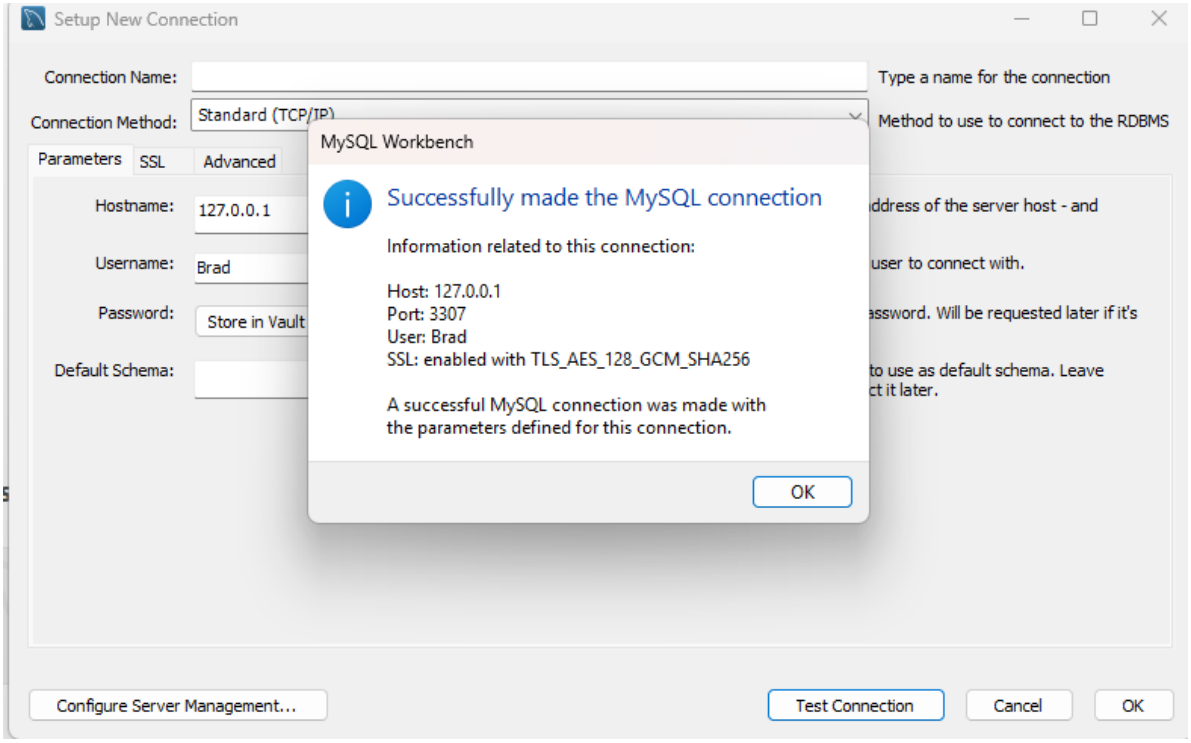


- Administrador: Crear usuarios y procesos.

-- 2. Administrador: Crear usuarios y procesos.

-- Crear usuario con el rol 'Administrador'
CREATE USER 'Brad'@'localhost' IDENTIFIED BY 'admin123';

-- Asignar rol 'Administrador', donde puede crear usuarios y procesos
GRANT CREATE USER, PROCESS, GRANT OPTION ON *.* TO 'Brad'@'localhost';
FLUSH PRIVILEGES;

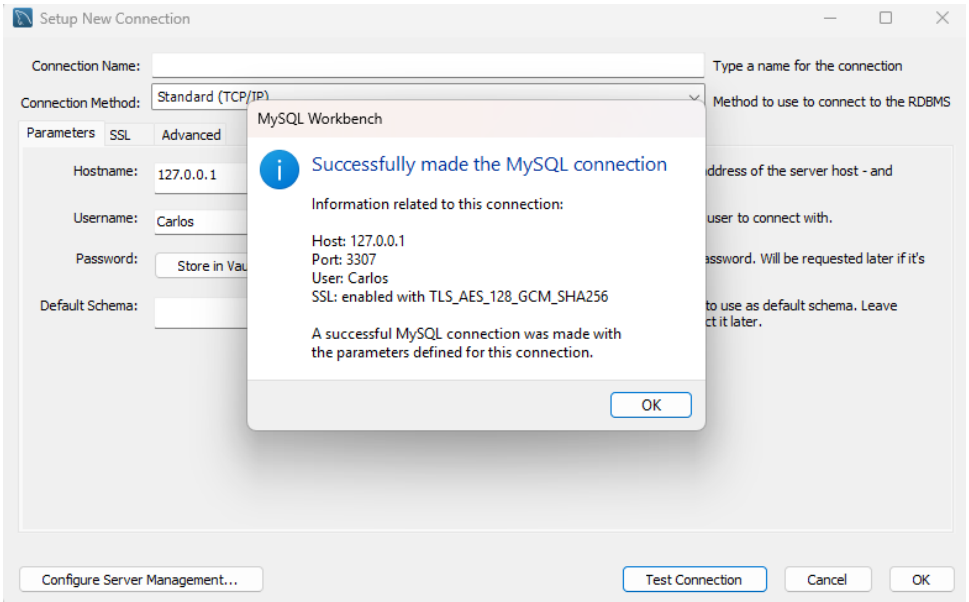


- CRUD: Insertar, actualizar y eliminar datos.

-- 3. CRUD: Insertar, actualizar y eliminar datos.

-- Crear usuario con el rol 'CRUD'
CREATE USER 'Carlos'@'localhost' IDENTIFIED BY 'crud123';

-- Asignar rol 'CRUD', donde puede seleccionar, insertar, actualizar
-- y eliminar en todas las tablas de la base de datos "hospital"
GRANT SELECT, INSERT, UPDATE, DELETE ON hospital.* TO 'Carlos'@'localhost';
FLUSH PRIVILEGES;

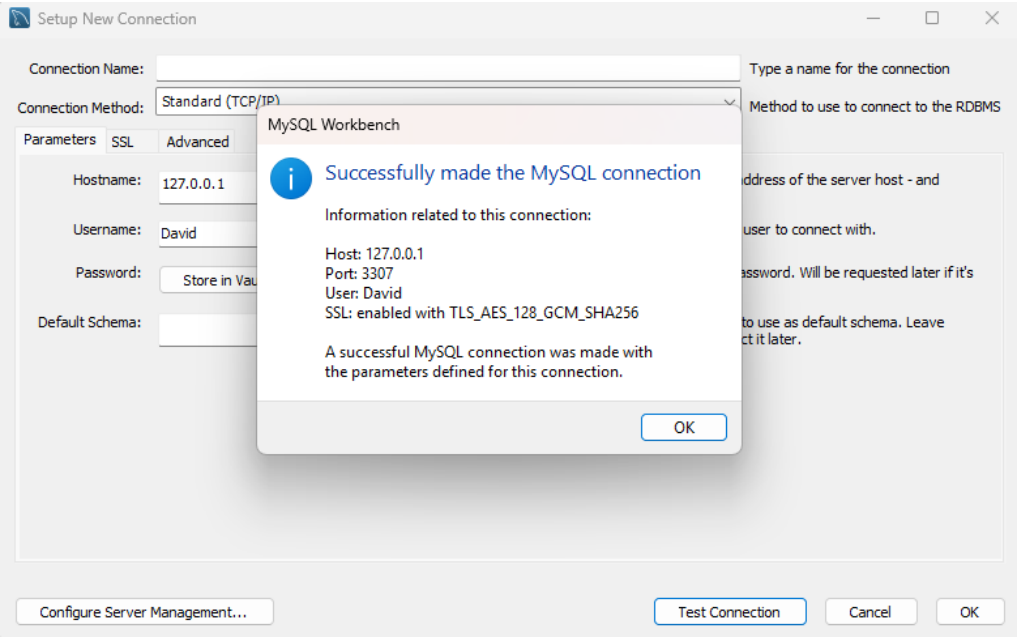


- CRU: Insertar y actualizar, pero sin eliminar.

-- 4. CRU: Insertar y actualizar, pero sin eliminar.

-- Crear usuario con el rol 'CRU'
CREATE USER 'David'@'localhost' IDENTIFIED BY 'cru123';

-- Asignar rol 'CRU', donde puede seleccionar, insertar y actualizar
-- en todas las tablas de la base de datos "hospital"
GRANT SELECT, INSERT, UPDATE ON hospital.* TO 'David'@'localhost';
FLUSH PRIVILEGES;

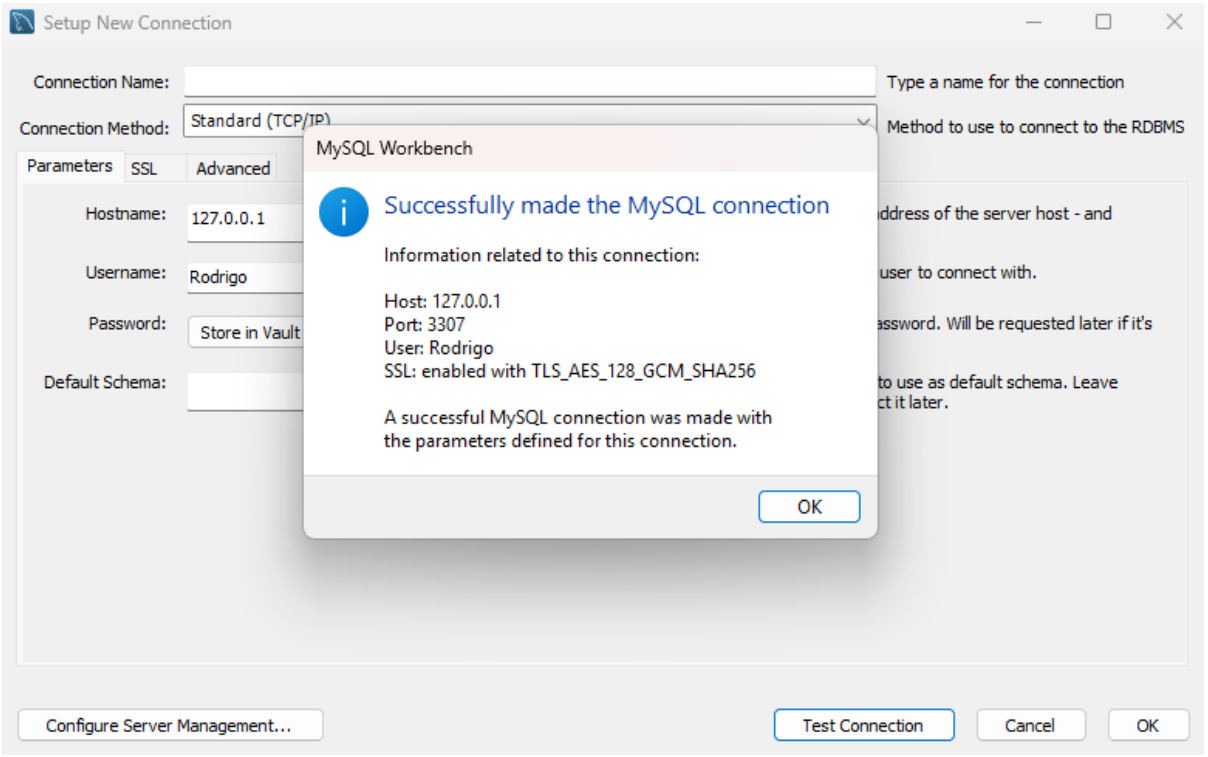


- Solo Lectura: Realizar consultas a las tablas.

-- 5. Solo Lectura: Realizar consultas a las tablas.

-- Crear usuario con el rol 'Solo Lectura'
CREATE USER 'Rodrigo'@'localhost' IDENTIFIED BY 'read123';

-- Asignar rol 'Solo_Lectura', donde puede seleccionar
-- en todas las tablas de la base de datos "hospital"
GRANT SELECT ON hospital.* TO 'Rodrigo'@'localhost';
FLUSH PRIVILEGES;



Verificación de Permisos

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

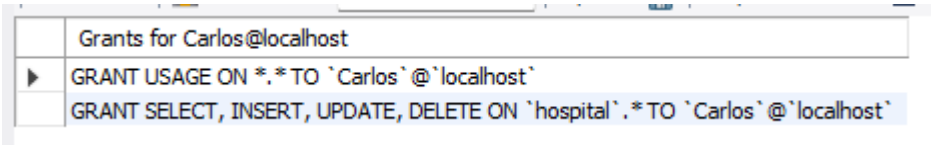
-- Verificar permisos del usuario 'Super Administrador'
SHOW GRANTS FOR 'Alexander'@'localhost';

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

-- Verificar permisos del usuario 'Administrador'
SHOW GRANTS FOR 'Brad'@'localhost';

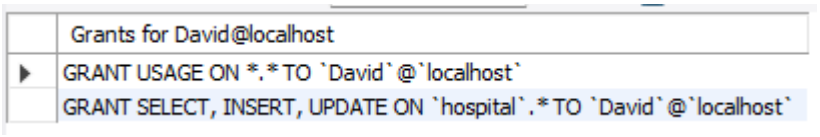
Grants for Brad@localhost
GRANT PROCESS, CREATE USER ON *.* TO `Brad`@`localhost` WITH GRANT OPTION

-- Verificar permisos del usuario 'CRUD'
SHOW GRANTS FOR 'Carlos'@'localhost';



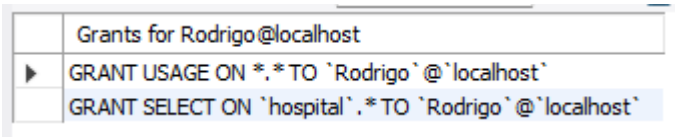
Grants for Carlos@localhost
GRANT USAGE ON *.* TO `Carlos`@`localhost`
GRANT SELECT, INSERT, UPDATE, DELETE ON `hospital`.* TO `Carlos`@`localhost`

-- Verificar permisos del usuario 'CRU'
SHOW GRANTS FOR 'David'@'localhost';



Grants for David@localhost
GRANT USAGE ON *.* TO `David`@`localhost`
GRANT SELECT, INSERT, UPDATE ON `hospital`.* TO `David`@`localhost`

-- Verificar permisos del usuario 'Solo Lectura'
SHOW GRANTS FOR 'Rodrigo'@'localhost';



Grants for Rodrigo@localhost
GRANT USAGE ON *.* TO `Rodrigo`@`localhost`
GRANT SELECT ON `hospital`.* TO `Rodrigo`@`localhost`

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`).

Triggers

Un trigger o desencadenador es un conjunto de instrucciones que se ejecutan de forma automática en una base de datos cuando ocurre un evento específico en una tabla o vista.

Los triggers se utilizan para mantener la integridad de los datos, automatizar tareas, o auditar cambios sin intervención explícita del usuario o la aplicación.

Funcionamiento

El funcionamiento de un trigger se basa en una serie de pasos:

1. **Evento de activación:** Un trigger se activa cuando ocurre un evento de modificación de datos (INSERT, UPDATE o DELETE) en la tabla o vista asociada.
2. **Condición (opcional):** Algunos triggers incluyen condiciones o restricciones adicionales que deben cumplirse para que se ejecute el código del trigger.
3. **Acción:** El trigger ejecuta una acción predeterminada, como la actualización de otra tabla, la validación de datos, o la ejecución de un conjunto de instrucciones SQL.
4. **Automatización:** Una vez configurado, el trigger se ejecuta automáticamente sin necesidad de intervención manual.

Tipos de Trigger

1. **BEFORE** Trigger (Antes de la operación)

- Se ejecuta antes de que la operación de modificación (INSERT, UPDATE o DELETE) se realice sobre la base de datos.
- Generalmente, se utiliza para validar o modificar los datos antes de que se inserten o actualicen en la tabla.
- **Uso común**: Comprobaciones de integridad de datos, validación de valores, y ajustes de datos antes de su inserción.

2. **AFTER** Trigger (Después de la operación)

- Se ejecuta después de que la operación de modificación (INSERT, UPDATE o DELETE) haya sido completada.
- Se utiliza generalmente para acciones que dependan de la operación realizada, como el registro de auditoría o la actualización de otras tablas.

- **Uso común:** Actualización de registros de auditoría, envío de notificaciones, o actualización de tablas relacionadas.

3. **INSTEAD OF** Trigger (En lugar de la operación)

- Este tipo de trigger reemplaza la operación de modificación (INSERT, UPDATE o DELETE) por una acción diferente que tú defines.
- Generalmente se usa en vistas para manejar operaciones de modificación, ya que las vistas no permiten modificaciones directas en algunas situaciones.
- **Uso común:** Modificación de datos en vistas complejas, sustitución de operaciones por otras personalizadas.

    - 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.

1. Automatización de Tareas

Una de las principales ventajas de los triggers es que automáticamente ejecutan acciones en la base de datos sin necesidad de intervención manual cada vez que se produce un evento (como la inserción, actualización o eliminación de registros). Esto reduce la carga de trabajo para los administradores y desarrolladores, ya que muchas tareas se realizan de forma automática.

Beneficios:

- **Reducción de errores humanos**: Al automatizar procesos, se minimiza el riesgo de que los usuarios cometan errores al realizar manualmente estas tareas.
- **Ahorro de tiempo:** Las tareas repetitivas y complejas se hacen de forma automática, liberando a los usuarios de realizar acciones redundantes.

2. Integridad Referencial

Asegura que las relaciones entre las tablas de la base de datos se mantengan consistentes. Los triggers ayudan a garantizar que los datos en las tablas relacionadas se mantengan sincronizados cuando se hacen cambios.

Beneficios:

- **Protege las relaciones entre tablas:** Evita que los datos se corrompan o se vuelvan inconsistentes, manteniendo la base de datos en un estado correcto.
- **Previene errores de eliminación:** Evita que se eliminen registros clave de una tabla si están siendo referenciados por otras.

3. Control de Cambios (Auditoría)

Los triggers también se pueden usar para registrar los cambios que ocurren en la base de datos, lo que es crucial para la auditoría y el seguimiento de modificaciones. Puedes usar un trigger para crear registros automáticos cada vez que se inserte, actualice o elimine un dato.

Beneficios:

- **Transparencia:** Los registros de auditoría permiten ver quién hizo qué cambios en los datos y cuándo lo hizo, lo que es muy útil para seguridad y control.
- **Cumplimiento de normativas:** Algunas regulaciones requieren el seguimiento detallado de los cambios en los datos (como en la industria financiera o de salud). Los triggers ayudan a cumplir con estas normativas.

4. Mantenimiento de la Consistencia de Datos

En bases de datos complejas, donde los datos pueden depender unos de otros (como las tablas de productos, precios, descuentos, etc.), los triggers son fundamentales para mantener la consistencia de los datos cuando se actualizan.

Beneficios:

- **Automatiza la actualización de datos relacionados:** Si los datos de una tabla cambian, los triggers pueden actualizar automáticamente las tablas relacionadas.
- **Evita inconsistencias:** Asegura que todas las dependencias entre tablas se mantengan de forma consistente sin tener que hacer cada actualización de forma manual.

5. Seguridad y Validación de Datos

Los triggers son útiles para validar los datos antes de que se guarden en la base de datos. Pueden verificar si los valores son correctos, si cumplen con ciertas reglas de negocio o si se ajustan a las restricciones definidas (como formatos de fechas, rangos numéricos, etc.).

Beneficios:

- **Prevenir datos incorrectos o inválidos:** Se aseguran de que los datos que se insertan o actualizan cumplan con las reglas establecidas.
- **Refuerza las reglas de negocio:** El trigger puede asegurar que las reglas del negocio se apliquen siempre, incluso si diferentes aplicaciones acceden a la base de datos.

6. Optimización del Rendimiento

Aunque los triggers pueden tener un costo en términos de rendimiento si no se usan correctamente, en muchos casos pueden mejorar el rendimiento de la base de datos al delegar tareas de procesamiento en lugar de hacerlo manualmente desde la aplicación.

Beneficios:

- **Descentralización del procesamiento:** Algunas operaciones pueden delegarse en la base de datos, lo que mejora la eficiencia general.

- **Minimiza el esfuerzo de la aplicación:** La lógica de negocio puede estar centralizada en la base de datos, reduciendo la carga en las aplicaciones que interactúan con ella.

    - 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*.

Ventajas

- Los triggers permiten automatizar tareas que de otra manera tendrían que ser gestionadas manualmente, lo que reduce la intervención humana y el riesgo de errores.
- Los triggers permiten centralizar y hacer cumplir las reglas de negocio a nivel de base de datos. Esto garantiza que cualquier operación que modifique los datos (ya sea realizada por una aplicación o un usuario) se ajuste a las reglas del negocio de manera consistente.
- Los triggers ayudan a mantener la integridad de los datos. Al ejecutar reglas de validación antes o después de una operación (como la inserción o actualización de datos), los triggers previenen la introducción de datos inconsistentes o incorrectos.

Desventajas

- A medida que se agregan más triggers a la base de datos, el sistema se vuelve más complejo y más difícil de mantener. Los triggers pueden interactuar entre sí de maneras inesperadas, lo que puede hacer que el comportamiento de la base de datos sea difícil de rastrear y depurar.
- Si no se gestionan correctamente, los triggers pueden tener efectos secundarios no deseados, como el ciclo infinito de ejecuciones. Esto puede ocurrir si un trigger

realiza una operación que activa a otro trigger, lo que a su vez activa al primero, y así sucesivamente.

- El uso de triggers puede tener implicaciones de seguridad si no se gestionan adecuadamente, especialmente cuando se manejan datos sensibles. Los triggers pueden ejecutar acciones en nombre de un usuario, lo que podría dar lugar a escaladas de privilegios o a la ejecución de código no autorizado si el acceso no está bien restringido.

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.

- **Notificaciones Automáticas (Alertas y Notificaciones)**

En algunos casos, los triggers se pueden utilizar para enviar notificaciones automáticas a los usuarios o sistemas externos cuando ocurren ciertos eventos en la base de datos. Esto es útil para alertar a los administradores, clientes o sistemas integrados cuando se producen cambios importantes.

- **Implementación de Reglas de Negocio Automáticas**

Los triggers son ideales para automáticamente aplicar reglas de negocio que deben cumplirse siempre que se manipulen datos. Esto asegura que los requisitos del negocio se mantengan consistentes, sin necesidad de que los desarrolladores o usuarios realicen estas comprobaciones manualmente.

- **Auditoría y Control de Cambios**

Los triggers son muy útiles para realizar un seguimiento de los cambios en la base de datos, registrando automáticamente cada vez que un dato es insertado, actualizado o eliminado. Esta auditoría es fundamental para conocer quién hizo qué cambios y cuándo, especialmente en entornos donde la seguridad y el cumplimiento son importantes.

    - 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.

1. Sistema Bancario y Financiero: Auditoría de transacciones

Las instituciones financieras (bancos, aseguradoras, etc.) dependen de triggers para auditar transacciones y controlar la integridad de los datos, debido a la necesidad de garantizar la seguridad y la transparencia.

Caso de uso: Auditoría y control de transacciones financieras

Los bancos utilizan triggers para auditar transacciones de depósitos, retiros y transferencias en tiempo real. Por ejemplo, cada vez que un cliente realiza una transacción, un trigger puede registrar la actividad en una tabla de auditoría.

2. Actualización automática de inventarios

Las plataformas de comercio electrónico como Amazon, eBay o Walmart utilizan triggers para gestionar de manera eficiente el inventario, especialmente cuando se realizan compras o devoluciones de productos.

Caso de uso: Gestión de inventario y actualizaciones automáticas

Cada vez que un cliente compra un producto o se realiza una devolución, un trigger puede actualizar automáticamente el stock disponible en la base de datos, asegurando que los niveles de inventario se mantengan precisos en tiempo real.

3. Mantenimiento de la integridad referencial

Las empresas de manufactura y distribución como Ford, General Electric o Nestlé utilizan triggers para mantener la integridad referencial entre las tablas de productos, órdenes y proveedores.

Caso de uso: Integridad referencial y control de existencias

En sistemas de inventarios complejos, como los de empresas manufactureras, los triggers aseguran que no se eliminen registros importantes (como productos o proveedores) si están siendo referenciados por otras tablas.

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:

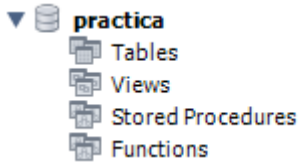
Imagina que tienes una empresa que desea llevar un control detallado sobre los cambios realizados en los registros de sus empleados. Para ello, se necesita crear una tabla de auditoría que registre cualquier acción (inserción, actualización o eliminación) que se realice en la tabla de **Empleados**.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

```
-- crear la base de datos "practica"
CREATE DATABASE practica;
-- seleccionar la base de datos "practica"
use practica;
```




```
-- crear la tabla "empleados"
```

```
CREATE TABLE empleados (
   EmpID INT PRIMARY KEY,
   Nombre VARCHAR(50),
   Departamento VARCHAR(50),
   Salary DECIMAL(10, 2)
);
```

Table: empleados

Columns:

EmpID	int PK
Nombre	varchar(50)
Departamento	varchar(50)
Salario	decimal(10,2)

	EmpID	Nombre	Departamento	Salario
*	NULL	NULL	NULL	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)

```
-- crear la tabla "auditoria"
CREATE TABLE auditoria (
   AudID INT PRIMARY KEY AUTO_INCREMENT ,
   Accion VARCHAR(10),
   EmpID INT,
   Nombre VARCHAR(50),
   Departamento VARCHAR(50),
   Salario DECIMAL(10, 2),
   Fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Table: auditoria

Columns:

AudID	int AI PK
Accion	varchar(10)
EmpID	int
Nombre	varchar(50)
Departamento	varchar(50)
Salario	decimal(10,2)
Fecha	timestamp

	AudID	Accion	EmpID	Nombre	Departamento	Salario	Fecha
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

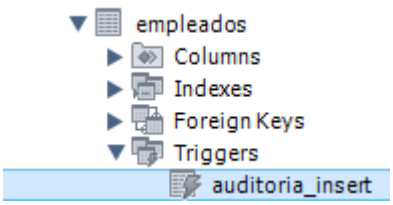
3. **Crear el trigger**:
    - El trigger debe activarse **después** de realizar cualquier operación (INSERT, UPDATE o DELETE) sobre la tabla de empleados. El trigger debe insertar un nuevo registro en la tabla de auditoría cada vez que se realice una de estas operaciones.

```
-- Trigger para INSERT
DELIMITER //
CREATE TRIGGER auditoria_insert
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 ;
```

Trigger: auditoria_insert

Definition:

Event	INSERT
Timing	AFTER

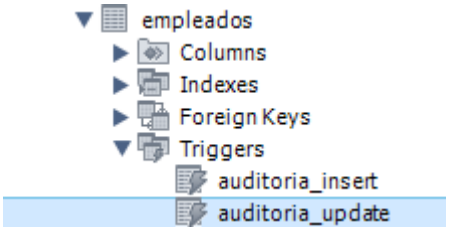


```
-- Trigger para UPDATE
DELIMITER //
CREATE TRIGGER auditoria_update
AFTER UPDATE ON empleados
FOR EACH ROW
BEGIN
   INSERT INTO auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
   VALUES ('UPDATE', NEW.EmpID, NEW.Nombre, NEW.Departamento, NEW.Salario,
NOW());
END//
DELIMITER ;
```

Trigger: auditoria_update

Definition:

```
Event    UPDATE
Timing   AFTER
```

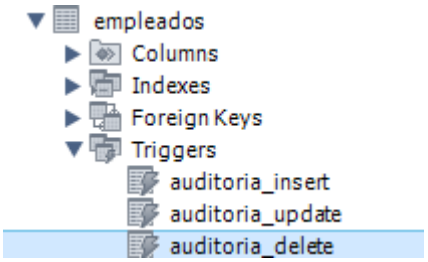



```
-- Trigger para DELETE
DELIMITER //
CREATE TRIGGER auditoria_delete
AFTER DELETE ON empleados
FOR EACH ROW
BEGIN
   INSERT INTO auditoria (Accion, EmpID, Nombre, Departamento, Salario, Fecha)
   VALUES ('DELETE', OLD.EmpID, OLD.Nombre, OLD.Departamento, OLD.Salario,
NOW());
END//
DELIMITER ;
```

Trigger: auditoria_delete

Definition:

```
Event    DELETE
Timing   AFTER
```



-- Insertar registro en la tabla "empleados"
INSERT INTO empleados (EmpID, Nombre, Departamento, Salario)
VALUES
(1, 'Fernando', 'Ventas', 4500.00),
(2, 'Marco', 'Mantenimiento', 8000.00),
(3, 'Santiago', 'Contabilidad', 2000.00);

EmpID	Nombre	Departamento	Salario
1	Fernando	Ventas	4500.00
2	Marco	Mantenimiento	8000.00
3	Santiago	Contabilidad	2000.00
NULL	NULL	NULL	NULL

-- Visualizar registro en la tabla "auditoria"
SELECT * FROM auditoria;

AudID	Accion	EmpID	Nombre	Departamento	Salario	Fecha
1	INSERT	1	Fernando	Ventas	4500.00	2024-12-30 20:41:09
2	INSERT	2	Marco	Mantenimiento	8000.00	2024-12-30 20:41:09
3	INSERT	3	Santiago	Contabilidad	2000.00	2024-12-30 20:41:09
NULL	NULL	NULL	NULL	NULL	NULL	NULL

-- Actualizar registro en la tabla "empleados"
UPDATE empleados
SET Salario = 6445.00
WHERE EmpID = 3;

EmpID	Nombre	Departamento	Salario
1	Fernando	Ventas	4500.00
2	Marco	Mantenimiento	8000.00
3	Santiago	Contabilidad	6445.00
NULL	NULL	NULL	NULL

-- Visualizar registro en la tabla "auditoria"
SELECT * FROM auditoria;

AudID	Accion	EmpID	Nombre	Departamento	Salario	Fecha
1	INSERT	1	Fernando	Ventas	4500.00	2024-12-30 20:41:09
2	INSERT	2	Marco	Mantenimiento	8000.00	2024-12-30 20:41:09
3	INSERT	3	Santiago	Contabilidad	2000.00	2024-12-30 20:41:09
4	UPDATE	3	Santiago	Contabilidad	6445.00	2024-12-30 20:42:54
NULL	NULL	NULL	NULL	NULL	NULL	NULL

-- Eliminar registro en la "tabla empleados"
DELETE FROM empleados
WHERE EmpID = 2;

EmpID	Nombre	Departamento	Salario
1	Fernando	Ventas	4500.00
3	Santiago	Contabilidad	6445.00
NULL	NULL	NULL	NULL

-- Visualizar registro en la tabla "auditoria"
SELECT * FROM auditoria;

AudID	Accion	EmpID	Nombre	Departamento	Salario	Fecha
1	INSERT	1	Fernando	Ventas	4500.00	2024-12-30 20:41:09
2	INSERT	2	Marco	Mantenimiento	8000.00	2024-12-30 20:41:09
3	INSERT	3	Santiago	Contabilidad	2000.00	2024-12-30 20:41:09
4	UPDATE	3	Santiago	Contabilidad	6445.00	2024-12-30 20:42:54
5	DELETE	2	Marco	Mantenimiento	8000.00	2024-12-30 20:44:36
NULL	NULL	NULL	NULL	NULL	NULL	NULL

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.

Enlace Git Hub:

https://github.com/CristianTambaco/Deber_ROLES---TRIGGER.git

Formato de Entrega:

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

Bibliografía

[1] «¿Qué es un trigger en una base de datos?», Ayuda Ley Protección Datos. Accedido: 30 de diciembre de 2024. [En línea]. Disponible en: https://ayudaleyprotecciondatos.es/bases-de-datos/trigger/

[2] «SQL triggers: disparadores y uso», CertiDevs. Accedido: 30 de diciembre de 2024. [En línea]. Disponible en: https://certidevs.com/tutorial-sql-triggers

[3] javifer2, «Trigger. ¿Qué es? ¿Cómo se hace? ¿Y cómo funciona? AFTER – FOR INSERT», Datos y demás. Accedido: 31 de diciembre de 2024. [En línea]. Disponible en: https://javifer2.wordpress.com/2020/04/12/trigger-que-es-como-se-hace-y-como-funciona-after-for-insert/

[4] «Los Triggers en sql server | Sutori». Accedido: 31 de diciembre de 2024. [En línea]. Disponible en: https://www.sutori.com/es/historia/los-triggers-en-sql-server--e7QBQRL8ApxaVY59M1prC5sZ