MODIFICACIÓN DE UNA BBDD
RESTRICCIONES Y DISPARADORES

Diseño y Programación de una BBDD

ÍNDICE

RESTRICCIONES Y DISPARADORES

1. **Explicar para qué se utilizan las cláusulas ON UPDATE y ON DELETE al momento de crear una relación entre dos tablas. Explica y pon un ejemplo de cada uno de los valores que puede tomar cada una de estas restricciones (RESTRICT, CASCADA, SET NULL) indicando qué sucedería si la restricción se configura de una manera determinada.**

Estas cláusulas se utilizan para definir el comportamiento de una relación entre tablas cuando se actualiza o elimina un registro en la tabla principal. Los valores que pueden tomar son:

RESTRICT: Impide la actualización o eliminación si hay registros relacionados en la tabla secundaria.

CASCADE: Actualiza o elimina automáticamente los registros relacionados en la tabla secundaria.

SET NULL: Establece los valores de las columnas relacionadas en la tabla secundaria como NULL si se actualiza o elimina el registro en la tabla principal.

Ejemplo:

Supongamos que tenemos una tabla de "EMPLEADOS" con una clave primaria "ID_EMPLEADO" y otra tabla "TAREAS" con una clave foránea "ID_EMPLEADO" que se relaciona con "EMPLEADOS", y configuramos la relación de esta manera:

```
ALTER TABLE TAREAS

ADD FOREIGN KEY (ID_EMPLEADO) REFERENCES EMPLEADOS(ID_EMPLEADO)

ON DELETE SET NULL

ON UPDATE CASCADE;
```

Si eliminamos un empleado en la tabla "EMPLEADOS", la columna "ID_EMPLEADO" en la tabla "TAREAS" se establecerá como NULL, lo que significa que las tareas relacionadas con ese empleado quedarán sin asignar a nadie.

Si actualizamos el ID de un empleado en la tabla "EMPLEADOS", todas las tareas relacionadas en la tabla "TAREAS" se actualizarán automáticamente con el nuevo ID del empleado.

2. **Explicar para qué se utiliza la restricción CHECK y de qué manera puede ayudar al control de la información de la Base de Datos. Para qué se utiliza la cláusula opcional ENFORCED/NOT ENFORCED? ¿Cuál es la sintaxis para crear este tipo de restricción? Poner un ejemplo con las tablas de la Base de Datos y explicar cómo funcionaría.**

La restricción **CHECK** se utiliza para imponer condiciones en los valores que se pueden insertar en una columna. Puede ayudar a controlar la integridad de los datos en la base de datos.

La cláusula opcional **ENFORCED** o **NOT ENFORCED** indica si la restricción debe aplicarse o no. Si se establece en **NOT ENFORCED**, la restricción se registra, pero no se aplica.

Sintaxis para crear una restricción CHECK:

```
CREATE TABLE Nombre_Tabla (

Nombre_Columna Tipo_Dato CHECK (condición)

);
```

Por ejemplo, con la tabla **“Actividad”** de nuestra BBDD, un ejemplo sería con la columna **“duración_sesion_minutos”** esta abarca actividades que duran entre **30 y 55 minutos**. Podríamos crear la siguiente **restricción CHECK**:

```
CREATE TABLE Actividad (

ID_Actividad INT PRIMARY KEY,

Nombre VARCHAR(50),

Duracion INT CHECK (Duracion >= 30 AND Duracion <= 55) ENFORCED

);
```

En este caso, la **restricción CHECK** se **asegura de que la duración de una actividad esté entre 30 y 55 minutos**. Si intentas insertar una actividad con una duración fuera de ese rango, se generaría un error y la inserción se detendrá.

Esto garantiza que los datos en la tabla **"Actividad"** cumplan con las restricciones especificadas y ayudan a mantener la integridad de la información en la base de datos del gimnasio.

3. **¿De qué manera pueden ayudar a optimizar las operaciones en una Base de Datos las columnas autogeneradas? Da un ejemplo utilizando la Base de datos del Gimnasio como referencia.**

Las columnas autogeneradas pueden optimizar las operaciones al reducir la necesidad de proporcionar manualmente valores en inserciones.

Por ejemplo, en la base de datos del **"Gimnasio",** si tenemos la tabla de "socio" autogenerada con su respectiva columna **"id_socio",** no es necesario proporcionar un valor al insertar un nuevo miembro, ya que la base de datos va a generar automáticamente un ID único.

4. **Definir lo que es un disparador y explicar en qué situaciones resultaría útil su implementación en la Base de Datos del Gimnasio.**

Un disparador es un procedimiento almacenado que se ejecuta automáticamente en respuesta a un evento específico en la base de datos.

Resulta útil para **automatizar tareas, validar datos o mantener la integridad de la base de datos** en situaciones específicas en la Base de Datos del Gimnasio.

5. **¿Cuál es la sintaxis para la creación de un disparador/trigger?**

```
CREATE TRIGGER nombre_trigger

{BEFORE | AFTER} {INSERT | UPDATE | DELETE}

ON nombre_tabla

FOR EACH ROW

BEGIN

    -- Cuerpo del disparador

END;
```

6. **Antes de proceder a crear un disparador se utiliza la cláusula DELIMITER. ¿Para qué sirve esta cláusula y cómo se utiliza?**

La cláusula **DELIMITER** se utiliza para cambiar el delimitador de comandos SQL en un entorno de creación de disparadores. Permite escribir el cuerpo del disparador con comandos SQL que pueden incluir punto y coma (;) sin que esto se interprete como el final del comando.

Cómo se utiliza:

Antes de definir el disparador, estableces un **delimitador personalizado,** como **"//". Esto se hace con la cláusula DELIMITER.**

```
DELIMITER //
```

Luego, defines **el disparador.** El delimitador personalizado se utiliza al final del disparador para indicar su final.

```
CREATE TRIGGER nombre_trigger

-- Cuerpo del disparador

//
```

Finalmente, restauras el delimitador original, que es el punto y coma (;), utilizando la cláusula DELIMITER sin argumentos.

```
DELIMITER ;
```

Al cambiar el delimitador a **"//", SQL entenderá que el disparador no finaliza hasta que se encuentre "//" al final**. Esto permite incluir múltiples sentencias SQL dentro del cuerpo del disparador sin que el punto y coma dentro del cuerpo sea interpretado como el final del disparador en sí.

7. **¿Cuántos tipos de disparadores hay? (clasificación de los *triggers* por sus desencadenantes) ¿Cuáles son? Da un ejemplo de cada uno de ellos.**

Los tipos de disparadores se clasifican por sus desencadenantes:

- **BEFORE INSERT:** Se ejecuta antes de insertar un nuevo registro.

Ejemplo: Supongamos que tienes una tabla "Clientes" y deseas asignar automáticamente un número de cliente al insertar un nuevo cliente. Puedes usar un disparador BEFORE INSERT para lograrlo.

```
CREATE TRIGGER trig_before_insert_cliente

BEFORE INSERT ON Clientes
```

```
FOR EACH ROW

BEGIN

SET NEW.numero_cliente = CONCAT('C', LPAD(NEW.ID_Cliente, 4, '0'));

END;
```

- **AFTER INSERT:** Se ejecuta después de insertar un nuevo registro.

Ejemplo: Después de insertar un nuevo registro en la tabla "Pedidos", un disparador AFTER INSERT podría enviar una notificación por correo electrónico al cliente para confirmar el pedido.

- **BEFORE UPDATE:** Se ejecuta antes de actualizar un registro.

Ejemplo: Supongamos que tienes una tabla "Inventario" y deseas evitar que el stock baje de cero. Puedes usar un disparador BEFORE UPDATE para verificar y evitar la actualización si el stock se reduce a un valor negativo.

```
CREATE TRIGGER trig_before_update_inventario

BEFORE UPDATE ON Inventario

FOR EACH ROW

BEGIN

IF NEW.stock < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'El stock no puede ser negativo';

END IF;

END;
```

- **AFTER UPDATE:** Se ejecuta después de actualizar un registro.

Ejemplo: Después de actualizar el estado de un pedido en la tabla "Pedidos", un disparador AFTER UPDATE podría registrar la fecha y hora de la última actualización en una tabla de auditoría.

- **BEFORE DELETE:** Se ejecuta antes de eliminar un registro.

Ejemplo: Antes de eliminar un registro en la tabla "Usuarios", un disparador BEFORE DELETE podría verificar si el usuario tiene dependencias en otras tablas y evitar la eliminación si es necesario.

- **AFTER DELETE:** Se ejecuta después de eliminar un registro.

Ejemplo: Después de eliminar un cliente de la tabla "Clientes", un disparador AFTER DELETE podría realizar una copia de seguridad de los datos del cliente eliminado en una tabla de historial.

8. ¿Para qué y en qué tipo de disparador se utilizan los prefijos OLD y NEW?

Los prefijos **OLD** y **NEW** se utilizan en los disparadores para acceder a los valores anteriores (OLD) y nuevos (NEW) de las filas afectadas por la operación (por ejemplo, inserción, actualización o eliminación). Se utilizan en disparadores de actualización y eliminación para comparar o modificar los valores antes y después de la operación.

Disparadores de Actualización (UPDATE): Los prefijos OLD y NEW se utilizan en los disparadores que se activan antes o después de una operación de actualización en una fila de una tabla.

Disparadores de Eliminación (DELETE): También se utilizan en los disparadores que se activan antes o después de una operación de eliminación en una fila de una tabla.

En estos disparadores, OLD se refiere a los valores anteriores de las filas antes de la operación (antes de la actualización o eliminación), y NEW se refiere a los nuevos valores que se aplicarán después de la operación. Puedes utilizar estos valores para comparar el estado anterior y posterior de los datos y tomar decisiones basadas en esas diferencias. Por ejemplo, puedes registrar cambios en un historial o evitar la eliminación de un registro si ciertas condiciones no se cumplen.

No se utilizan en los disparadores de inserción (INSERT), ya que no hay valores "antiguos" en una fila que aún no existe antes de la inserción.