A6. Disparadores.

Índice.

A6. Disparadores.

1. ¿Qué es un disparador?

A6. Disparadores.

1. ¿Qué es un disparador?

Un **disparador**, desencadenador o trigger es un programa almacenado que se dispara (ejecuta) automáticamente como respuesta a algún suceso que ocurra en la base de datos.

En MySQL este tipo de suceso corresponde a alguna instrucción DML sobre alguna tabla (INSERT, UPDATE, DELETE).

El disparador es un mecanismo para asegurar la integridad de los datos y realizar auditorías sobre dichos datos.

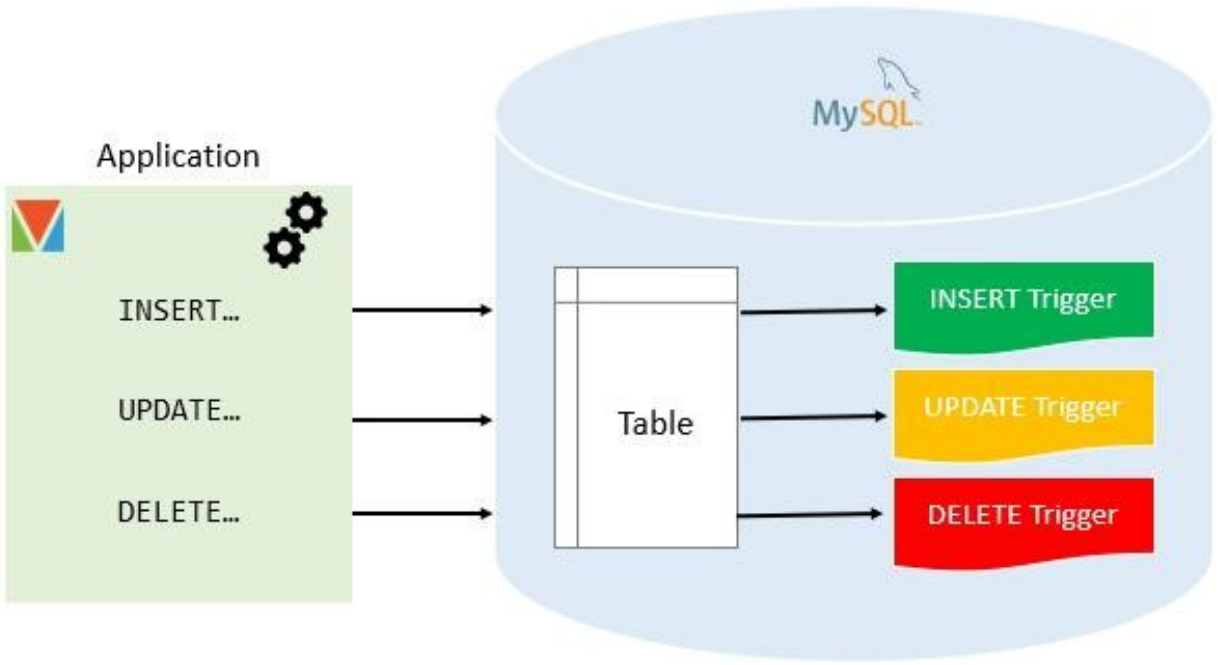
No hay que abusar de la utilización de los disparadores debido a que llevan consigo una sobrecarga del sistema y, por tanto, reducen el rendimiento

No se puede crear un disparador sobre tablas temporales o vistas.

2. Características.

Un **disparador** (o trigger) presenta las siguientes características:

- Se define para una tabla (o vista) específica.
- Se crea para conservar la integridad referencial y la coherencia entre los datos de distintas tablas.
- Si se intenta modificar (agrega, borrar o actualizar) datos de una tabla, se ejecuta (o dispara) el tigger definido de forma automática.
- Se asocia a un evento sobre una tabla (inserción, borrado o actualización).



3. Variables OLD y NEW.

El **disparador** trabaja con dos tipos de variables (OLD y NEW) que son de tipo %ROWTYPE y en las que guarda una copia del registro antes (OLD) y después (NEW) de la acción de la sentencia DML asociada.

Estas dos variables SÓLO son válidas cuando el tigger es a nivel de fila.

ACCION SQL	OLD	NEW
INSERT	No definido; todos los campos toman valor NULL.	Valores que serán insertados cuando se complete la orden.
UPDATE	Valores originales de la fila, antes de la actualización.	Nuevos valores que serán escritos cuando se complete la orden.
DELETE	Valores, antes del borrado de la fila.	No definidos; todos los campos toman el valor NULL.

4. Disparador vs Procedimiento Almacenado.

La diferencia entre un **disparador** (o trigger) y un Procedimiento Almacenado radica en:

- El disparador NO puede ser invocado directamente SINO que su ejecución es de forma automática.
- El disparador NO recibe parámetros NI devuelve un valor (o valores).
- El disparador SÍ es apropiado para mantener la integridad de los datos, pero NO para ofrecer resultados de consultas.

	Trigger o Disparador	Procedimiento Almacenado
Definición	create trigger	create procedure
Diccionario de datos Con código fuente en	user_triggers	user_source
Invocación	implícita	explícita
Comandos commit, Savepoint y rollback	NO permitidos	SÍ permitidos
Asignación a	Tabla o Vista NO a Base de Datos	Base de Datos NO a tabla NI a Vista
Utilización	comandos INSERT. DELETE, UPDATE	Tareas Administrativas o Dentro de una aplicación

5. Creación de un disparador.

La sintaxis para crear un disparador es la siguiente:

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

Los componentes asociados son:

- Privlegios de ejecución → DEFINER — cuenta de un usuario o del usuario actual
- Nombre → TRIGGER — nombre del trigger
- Temporalidad (trigger_time) → BEFORE, AFTER — antes o después de la operación DML
- Operación (trigger_event) → INSERT, UPDATE, DELETE — operación que se realizará sobre la tabla
- Tabla → ON — define la tabla asociada al trigger
- Secuencia de ejecución → FOR EACH ROW — ejecución del trigger para cada fila de la tabla afectada por la operación DML
- Cuerpo → BEGIN … END — conjunto de instrucciones que forman. NO puede contener un CALL

5. Creación de un disparador.

```
DELIMITER //
CREATE TRIGGER update_vm_products
        AFTER UPDATE
        ON id0ap_virtuemart_products FOR EACH ROW
        BEGIN
             INSERT INTO id0ap_virtuemart_notifications
             VALUES ("", NEW.virtuemart_product_id, "product","update");
        END //
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER ejercicio10_AI
AFTER INSERT ON etapa FOR EACH ROW
BEGIN
    IF (STRCMP(NEW.llegada,NEW.salida=0)) THEN
        SELECT "Salida y llegada en la misma etapa";
    ELSE
    SELECT "Salida y llegada están en etapas distintas"
    END IF;

END; //
DELIMITER;
```

```
DELIMITER //
CREATE TRIGGER ejercicio6_BI
BEFORE INSERT ON ciclista FOR EACH ROW

    DECLARE numCiclistas INT;

    SELECT COUNT(*)
    INTO numCiclistas
    FROM ciclista;

    IF numCiclistas < 100 THEN
        SELECT "Hay menos de 100";
    ELSE
        SELECT "Tenemos ciclistas suplentes";
    END IF;
END; //
DELIMITER;
```

6. Eliminación de disparadores.

La sintaxis para borrar un disparador es la siguiente:

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name
```

Puede ponerse al principio o al final del código, o bien solitario, o bien antes de su creación.

```
drop trigger tr_validar_cliente

DROP TRIGGER IF EXISTS update_vm_products ;
DELIMITER //
CREATE TRIGGER update_vm_products
        AFTER UPDATE
        ON id0ap_virtuemart_products FOR EACH ROW
        BEGIN
              INSERT INTO id0ap_virtuemart_notifications
              VALUES ("", NEW.virtuemart_product_id, "product","update");
        END //
DELIMITER ;
```

7. Utilización de disparadores.

La utilización de los disparadores resulta útil en cuestiones asociadas a:

- Copias o réplicas de datos → mantener sincronizada la copia de seguridad con los datos.
- Mantenimiento del stock de productos.
- Auditoría de las operaciones sobre las tablas.
- Validación de los datos → además de los constrainsts (restricciones) que aparecen la creación de las tablas, se utiliza los disparadores para validar dichos datos que se van a almacenar y mantener su consistencia.

Un aspecto muy importante a tener en cuenta es la combinación entre una operación DML y el disparador asociado: si el disparador falla, también lo hará la operación DML que intenta modificarla tabla de la base de datos. Esto puede ser útil para evitar entradas indeseadas en las tablas de las bases de datos; de esta forma, si el disparador detecta un dato incorrecto, se podría provocar una situación de error que anularía y abortase la operación DML.