TEMA 7
TRIGGERS O DISPARADORES

BASE DE DATOS

Contenidos de la Unidad:

1. ¿Qué es un Trigger o Disparador?
2. Trigger
3. Triggers de tablas
4. Valores OLD y NEW
5. Auditorias
6. Consideraciones

1. ¿Qué es un Trigger o Disparador?

TEMA 7: TRIGGER O DISPARADORES

- Son un tipo especial de rutina que se ejecuta cuando en una tabla ocurre un evento del tipo INSERT, UPDATE o DELETE, es decir escrituras. En inglés se les conoce como trigger.
- No puede haber dos disparadores distintos asociados al mismo momento y sentencia.

2. Trigger

TEMA 7: TRIGGERS O DISPARADORES

¿Para que se utilizan?

- Implementar restricciones complejas de seguridad o integridad
- Posibilitar la realización de operaciones de manipulación sobre vistas
- Prevenir transacciones erróneas
- Implementar reglas de negocio complejas
- Generar automáticamente valores derivados
- Auditar las actualizaciones
- Enviar alertas

Ejemplo:

```
DELIMITER $$
DROP TRIGGER IF EXISTS comprobacion_saldo;$$
CREATE TRIGGER comprobacion_saldo BEFORE UPDATE ON movimientos FOR EACH ROW
BEGIN
  IF NEW.cantidad < 0 THEN
      SET NEW.cantidad = 0;
  ELSEIF NEW.cantidad > 100 THEN
      SET NEW.cantidad = 100;
  END IF;
END;$$
DELIMITER ;
```

3. Triggers de tablas

TEMA 7: TRIGGERS O DISPARADORES

Valores old y new

- Hacen referencia a los valores anterior y posterior a una actualización a nivel de fila, respectivamente.

- Sintaxis old.nombreColumna new.nombreColumna

- Debe tenerse en cuenta el evento de disparo:

  Si es DELETE => se utilizará old.nombreColumna (new será NULL)

  Si es INSERT => se utilizará new.nombreColumna (old será NULL)

  Si es UPDATE => se podrán utilizar tanto new como old

- Hay algunas combinaciones de OLD y NEW con BEFORE o AFTER que no son posibles. Hay que utilizar el sentido común. Si hay algún error, suele ser el Error 1362: Updating of NEW row is not allowed in after trigger.

4. Valores OLD y NEW

TEMA 7: TRIGGERS O DISPARADORES

- Cuando la sentencia que hace “saltar” al disparador es un **DELETE**, se podrán utilizar dentro del disparador variables del tipo **OLD**.xxxxxxxx.
- Cuando la sentencia que hace “saltar” al disparador es un **INSERT**, se podrán utilizar dentro del disparador variables del tipo **NEW**.xxxxxxxx.
- Por último si la operación que ha disparado el trigger es un **UPDATE**, se pueden utilizar tanto **NEW**.xxxxxxx como **OLD**.xxxxxxxx

¿Cuándo se disparan?

- Los disparadores se "disparan" antes (**BEFORE**) o después (**AFTER**) de la acción que los hace saltar. Puede haber varios disparadores "enganchados" a una misma tabla, pero no pueden saltar justo en el mismo momento.
- El nivel de disparo puede ser a nivel de orden o a nivel de fila (FOR EACH ROW):
  - A **nivel de orden**: el trigger se activará una sola vez para cada orden, independientemente del número de filas afectadas por ella. Es la opción por defecto si no se especifica en el trigger.
  - A **nivel de fila**: el trigger se activará una vez para cada fila afectada por la orden.

¿Cómo saber que triggers hay?

- Para saber si una base de datos tiene triggers la forma más sencilla es:

  mysql> **SHOW TRIGGERS FROM ebanca;**

- Otra posibilidad es consultarlo en information_schema, p.ej. En la base de datos "ebanca":

  ```
  SELECT trigger_name, event_manipulation,  event_object_table
  FROM information_schema.triggers  WHERE trigger_schema LIKE
  '%ebanca%';
  ```

- Para ver el código ejecutado mediante el trigger, p. ej. para el trigger llamado "n_rojos":

  ```
  SELECT action_statement FROM  information_schema.triggers
  where trigger_name LIKE  '%n_rojos%';
  ```

Ejemplo:

```
DELIMITER $$
DROP TRIGGER IF EXISTS comprobacion_saldo;$$
CREATE TRIGGER comprobacion_saldo BEFORE UPDATE ON movimientos FOR EACH ROW
BEGIN
  IF NEW.cantidad < 0 THEN
      SET NEW.cantidad = 0;
  ELSEIF NEW.cantidad > 100 THEN
      SET NEW.cantidad = 100;
  END IF;
END;$$
DELIMITER ;
```

Control de valores de entrada

Una sentencia que provocará que se dispare es:

- mysql> UPDATE movimientos SET cantidad = -30 WHERE id_movimiento=37;

Para comprobar si ha funcionado (en cantidad escribe 0 y no -30):

- mysql> SELECT * FROM movimientos WHERE id_movimiento=37;

Ejemplo 2:

```
DELIMITER $$

DROP TRIGGER IF EXISTS actualizar_cuenta;$$


CREATE TRIGGER actualizar_cuenta BEFORE INSERT ON movimientos
FOR EACH ROW

BEGIN
UPDATE cuentas SET saldo = saldo + NEW.cantidad  WHERE cod_cuenta
= NEW.cod_cuenta;

END;$$

DELIMITER ;
```

Para comprobarlo:

```
mysql> select c.* from cuentas c, clientes cl where cl.dni like '11111111' and
c.cod_cliente=cl.codigo_cliente;
```

Ahora lo hacemos saltar:

```
mysql> insert into movimientos(dni, fechahora,cantidad,cod_cuenta)
values ('11111111','2022-04-26',-1000,1);
```

Ahora volvemos a mirar el saldo del cliente con dni '11111111'

```
mysql> select c.* from cuentas c, clientes cl where cl.dni like '11111111' and
c.cod_cliente=cl.codigo_cliente;
```

5. Auditorias

TEMA 7: TRIGGERS O DISPARADORES

Ir apuntando todos los movimientos que se hacen en otra tabla:

```
CREATE TABLE auditoria_movimientos (
        id_mov int not null auto_increment,
        cod_cuenta_ant varchar(100),
        fecha_ant datetime,
        cantidad_ant int,
        cod_cuenta_n varchar(100),
        fecha_n datetime,
        cantidad_n int,
        usuario varchar(40),
        fecha_mod datetime,
        primary key(id_mov)
) ENGINE = InnoDB;
```

```
DELIMITER $$
CREATE TRIGGER trigger_auditoria_movimientos AFTER UPDATE ON
    movimientos
FOR EACH ROW
BEGIN
  INSERT INTO auditoria_movimientos(cod_cuenta_ant, fecha_ant,
    cantidad_ant, cod_cuenta_n, fecha_n, cantidad_n, usuario, fecha_mod)
   VALUES (OLD.cod_cuenta,OLD.fechahora, OLD.cantidad, NEW.cod_cuenta,
    NEW.fechahora, NEW.cantidad, CURRENT_USER(),NOW());

END;$$
DELIMITER ;
```

Para que salte (antes hay que borrar otro trigger de la BD que salta en el mismo momento, buscadlo con el comando SHOW TRIGGERS..):

mysql> **UPDATE movimientos SET cantidad = 888 where idmov=38;**

Para comprobar que ha funcionado:

mysql> **select * from auditoria_movimientos;**

6. Consideraciones

TEMA 7: TRIGGERS O DISPARADORES

Consideraciones de utilización

- Se necesita tener privilegio de TRIGGER y sobre los objetos referenciados
- No se puede asociar un trigger a una tabla del esquema de SYS
- Si un trigger falla, la operación DML asociada no se realizará
- No conviene abusar del uso de triggers por razones de ineficiencia