## 6.7 Disparadores (Triggers)

Los disparadores son una clase especial de procedimiento almacenado que se que se lanza automáticamente.

En MySql Server este objeto existe desde la versión 5.0.

Un disparador está siempre asociado a una tabla y a una instrucción de modificación de datos dentro de esa tabla. Se lanzará automáticamente justo antes o después (dependiendo del tipo de disparador) de la instrucción para la que se ha creado.

Un disparador no podrá estar asociado a una vista ni a una tabla temporal.

Las instrucciones para las que se puede preparar un disparador son aquellas que modifican datos, por tanto serán UPDATE, INSERT o DELETE. El disparador quedará asociado a una de estas instrucciones sobre una tabla concreta.

También tenemos que indicar si queremos que se lance el disparador antes (BEFORE) o después (AFTER) que la instrucción a la que se asocia.

Podremos tener hasta 6 disparadores asociados a cada tabla, pero todos deben ser diferentes. Por ejemplo, podemos tener un disparador “alter insert” y otro “before insert”, pero no podemos tener dos disparadores “before insert” sobre la misma tabla (El último machacará al anterior que tuviéramos del mismo tipo).

Son una herramienta eficaz que se utiliza para asegurar que se cumplen las restricciones de atributos de las que hablábamos en la obtención del modelo físico (representación del diagrama E/R en modelo relacional) en el apartado 1 de la unidad de repaso (Unidad 1).

A veces, este tipo de procedimiento almacenado es muy útil, por ejemplo, podríamos crear un disparador que, cuando queden un número mínimo de unidades de los productos que vende nuestra empresa, automáticamente genere un pedido de dicho producto.

Sintaxis

```
CREATE TRIGGER nombre_trigger {BEFORE | AFTER} {INSERT | DELETE | UPDATE}
ON nombre_tabla
FOR EACH ROW
    cuerpo_del_trigger;
```

nombre_trigger.- Es el nombre con el que se almacena el disparador.

nombre_tabla.- Tabla con la que está asociada el disparador.

BEFORE | AFTER.- Especifica el momento en el que se activará.

{INSERT | DELETE | UPDATE}.- Indica la instrucción que activará el trigger. Se puede utilizar el mismo desencadenador para más de una de estas instrucciones.

cuerpo_del_trigger.- Sentencia/s que se disparará/n. Si es más de una sentencia encerrarán en un bloque de sentencias (entre BEGIN y END).

**Los Alias OLD y NEW de MySQL**

Para trabajar con triggers, es fundamental conocer la existencia de dos objetos internos con los que trabaja MySQL (OLD y NEW), estós tienen sus equivalentes en otras herramientas de GBD.

Cuando ordenamos la ejecución de instrucciones de modificación de datos (INSERT, DELETE y/o UPDATE), internamente MySQL trabaja con dos objetos llamados OLD y NEW, estós objetos son un "ALIAS" de la tabla que estamos modificando (por tanto, tendrá la misma estructura, esto es, las mismas columnas) y contendrá solo la fila que hemos solicitado modificar.

Así, para la tabla centros (codcentro, nomcentro, dircentro)

a) Para UPDATE, si hemos pedido cambiar la dirección del centro 20:

OLD.dircentro ==> contendrá el antiguo valor de la dirección del centro 20.

NEW.dircentro ==> contendrá la nueva dirección del centro 20.

b) Para INSERT, si hemos pedido insertar un nuevo centro:

OLD.dircentro ==> No contendrá ningún valor, ya que se trata de un centro que antes de la operación no existía.

NEW.dircentro ==> contendrá la nueva dirección del centro que estamos insertando.

c) Para DELETE, si hemos pedido eliminar el centro 20:

OLD.dircentro ==> contendrá la dirección del centro 20, el que pretendemos eliminar.

NEW.dircentro ==> No contendrá ningún valor, ya que, después de la ejecución, el centro 20 no existirá.

**NOTA IMPORTANTE.-** Estos valores solo son accesibles desde los disparadores o triggers, nunca desde cualquier otro bloque de código que no esté dentro de un trigger.

Internamente, una modificación (UPDATE) implica un borrado (DELETE) y una inserción (INSERT).

**Permisos sobre los disparadores**

Desde la versión 5.0.2 de MySql Server, la creación de un disparador solo se puede hacerse con permisos de SUPER (administradores).

Para su ejecución hay que tener en cuenta que:

SET NEW.nombre_columna = valor ==> implica privilegio UPDATE sobre la columna nombre_columna .

SET nombre_var = NEW.nombre_columna ==> implica privilegio SELECT sobre la columna.

**Otras instruccioner relacionadas con triggers**

1. SHOW TRIGGERS [ {FROM | IN} nombre_bd]

    [LIKE 'patron' | WHERE predicado]

Los triggers que existen en una base de datos quedan registrados en la tabla INFORMATION_SCHEMA.TRIGGERS.

2. DROP TRIGGER [IF EXISTS] [nombre_bd.]nombre_trigger

**Ejemplo** (Sobre la tabla “departamentos” de BD empresa, comprobar que el presupuesto de un departamento no pueda decrementarse nunca)

```
DELIMITE $$
CREATE TRIGGER mitrigger BEFORE UPDATE ON departamentos
FOR EACH ROW
BEGIN
DECLARE numero int
if new.presude <old.presude then
begin
        set new.presude = old.presude;
        -- Ver Uso de SIGNAL.
        SIGNAL SQLSTATE '01000' SET MESSAGE_TEXT = 'El presupuesto no se modificará';
end;
end if;
END $$
DELIMITER ;
```

**Uso de la sentencia SIGNAL**

Esta sentencia nos permite provocar errores o warnings en MySQL y, además, definir el mensaje que se mostrará asociado a ese error/warning.

En otras herramientas, como MS SQL Server, podríamos mostrar el mensaje que indica que el presupuesto no va a ser modificado simplemente mediante una sentencia SELECT:

```
-- select 'El presupuesto no se modificará'
```

Sin embargo, en MySQL Server, los triggers no permiten mostrar valores, por lo que tenemos que utilizar la sentencia SIGNAL. Esta sentencia permite provocar errores o warnings.

Si ponemos un código para SQLSTATE que empiece por '01' solo provocaremos un warning, es decir, un aviso y podremos mostrar un mensaje asociado a este aviso.

Por el contrario si SQLSTATE fuese por ejemplo '55019' o '45000', provocaríamos un error que abortaría la operación que lanzó el trigger, además de mostrar el mensaje correspondiente.

Para más información consultar el siguiente enlace:

http://dev.mysql.com/doc/refman/5.5/en/signal.html

**¿Qué no puedo hacer dentro de un disparador MySQL?**

Dentro del código de los disparadores MySQL no podremos:

- Utilizar la sentencia SELECT para mostrar resultados (si para asignarlos)
- Llamar a procedimientos almacenados
- Utilizar transacciones.