Disparadores o triggers de MySQL

Un disparador de SQL es un conjunto de sentencias SQL almacenadas en la base de datos. Un disparador se ejecuta cuando ocurre un determinado evento sobre la tabla con la que está asociado por ejemplo cuando se insertan, actualizan o eliminan filas de una tabla. En MYSQL sólo se definen para las tablas, pero en Oracle las vistas también tienen unos disparadores especiales denominados INSTEAD.

Un disparador no es más que un tipo especial de procedimiento almacenado. La diferencia principal entre un trigger y un procedimiento almacenado es que el disparador se ejecuta automáticamente cuando se produce un determinado evento sobre la tabla mientras que un procedimiento almacenado se debe llamar explícitamente. Por tanto:

- Sirven para ejecutar tareas programadas automáticamente antes o después de realizar un cambio en los datos de las tablas.

- Los disparadores se usan para comprobar la integridad de los datos y controlar restricciones de las tablas que **no se pueden implementar con las Constrains**. Es decir, no controlamos por ejemplo en un disparador para Insert que el valor dado para la clave primaria no esté repetido ya que de ésto ya se encarga la constraint de Primary Key correspondiente que se define a la hora de crear la tabla.

Un disparador puede ser definido para ser invocado antes o después del cambio de datos producido en la tabla mediante las órdenes INSERT, UPDATE o DELETE. MySQL permite definir un máximo de **seis** disparadores para cada tabla.

- BEFORE INSERT
- AFTER INSERT
- BEFORE UPDATE
- AFTER UPDATE
- BEFORE DELETE
- AFTER DELETE

MySQL gestiona los errores ocurridos durante la ejecución de disparadores de esta manera:

- Si lo que falla es un disparador BEFORE, no se ejecuta la operación en el correspondiente registro.
- Un disparador AFTER se ejecuta solamente si el disparador BEFORE (de existir) se ejecutó con éxito.

- Un error durante la ejecución de un disparador BEFORE o AFTER tiene como consecuencia el fallo de toda la sentencia que provocó la ejecución del disparador. Este fallo se traduce en la cancelación o Rollback de todos los cambios realizados por esa sentencia.

NOTA: Tener en cuenta que por ejemplo, la sentencia TRUNCATE elimina los datos de una tabla pero no hace ejecutar ningún disparador.

Los disparadores que se definen para una tabla deben tener un nombre único. La convención que se usa para los nombres de los triggers es poner en primer lugar BEFORE o AFTER, a continuación el nombre de la tabla y por último el tipo de evento INSERT, UPDATE o DELETE. Por ejemplo before_empleados_update.

(**BEFORE** | **AFTER**)_nombre tabla_(**INSERT**| **UPDATE** | **DELETE**)

before_empleados_update.

Los disparadores en Mysql **no pueden**:

- Contener instrucciones como SHOW, LOAD DATA, LOAD TABLE, BACKUP DATABASE, RESTORE o RETURN.
- Usar COMMIT, ROLLBACK, START TRANSACTION, LOCK/UNLOCK TABLES, ALTER, CREATE, DROP, RENAME, EXECUTE… etc.
- Llamadas a procedimientos (Call) o funciones.
- Dentro del cuerpo del disparador no puede ejecutarse la instrucción que provocó su ejecución ya daría lugar a un bucle infinito de llamadas.

```
CREATE TRIGGER trigger_momento trigger_tabla trigger_evento
[BEFORE| AFTER] [DELETE|INSERT|UPDATE]
ON nombre_tabla
FOR EACH ROW
BEGIN
        …                          - - Begin y End para más de una instrucción
END;
```

- Pueden definirse más de un disparador para el mismo evento pero con momentos diferentes.
- FOR EACH ROW significa que para cada una de las filas afectadas por la orden que lo dispara, se ejecutará el código del disparador.

Oracle tiene además disparadores a nivel de orden (FOR EACH STATEMENT) que se ejecutan una única vez antes o después de la orden.

• Los prefijos OLD y NEW se usan únicamente cuando el disparador es a nivel de fila y hacen referencia respectivamente a los valores antiguos y nuevos de cada columna de la tabla. En la orden DELETE sólo se podrán usar los valores OLD, en la orden INSERT los valores NEW de las columnas y en la orden UPDATE tanto NEW como OLD.

```
CREATE TABLE audit_emple(
  id int(11) NOT NULL AUTO_INCREMENT,
  numemp int(11) NOT NULL,
  nombre varchar(50) NOT NULL,
  fecha_cambio datetime DEFAULT NULL,
  accion varchar(50) DEFAULT NULL,
  PRIMARY KEY (id)
);
```

```
DELIMITER $$
CREATE TRIGGER before_empleados_update
BEFORE UPDATE ON empleados
FOR EACH ROW
BEGIN
   INSERT INTO audit_emple                -- Esta  orden INSERT con SET
   SET accion = `update',                 -- es como INSERT con VALUES
         numemp = OLD.numemp,
         nombre = OLD.nombre,
         fecha_cambio= NOW();
END$$
DELIMITER ;
```

Los disparadores están en la carpeta/data_folder/database_name/table_name.trg. Aunque es más fácil verlo en el entorno, podemos ver código usando la siguientes órdenes en el intérprete de SQL:

```
SELECT * FROM Information_Schema.Trigger
WHERE Trigger_schema = 'database_name' AND
    Trigger_name = 'trigger_name';
```

Si quisiéramos ver todos los disparadores de un esquema:

```
SELECT * FROM Information_Schema.Trigger
WHERE Trigger_schema = 'database_name';
```

Para visualizar todos los disparadores de una tabla:

```
SELECT * FROM Information_Schema.Trigger
WHERE Trigger_schema = 'database_name' AND
    Event_object_table = 'table_name';
```

Para borrar disparadores:

```
DROP TRIGGER table_name.trigger_name;

DROP TRIGGER empleados.before_empleados_update;
```

Para modificar un disparador, hay que eliminarlo primero y volver a crearlo con el nuevo código. **No hay ALTER TRIGGER** .

Control de Errores en disparadores

En algunas ocasiones es necesario hacer fallar al disparador, es decir provocar un error intencionado y que la orden que provocó la ejecución del disparador no se lleve a cabo. En Oracle contamos con la orden RAISE_APPLICATION_ERROR (nº de error, mensaje de error), que hace que el disparador falle y se muestre por pantalla el mensaje que creamos apropiado. En MYSQL no existía esta posibilidad, por lo que se recurría a "chapuzas" para conseguirlo.

Para explicarlo, consideramos el siguiente código:

```
CREATE TRIGGER chequea_precio BEFORE INSERT ON libros
FOR EACH ROW
....
        IF new.precio <= 0
                INSERT INTO tabla (id) values (NULL);
        END IF;
END;
....
```

La tabla tiene una columna id no nula, al hacer esta inserción forzamos un error y el disparador falla con lo que no se realiza la orden INSERT pero se genera un error de violación de restricción. Podríamos pensar en mostrar un mensaje con SELECT 'texto del mensaje', pero tanto esta instrucción de salida como las SELECT normales no pueden usarse en los disparadores. Las SELECT.... INTO y los Cursores sí según los manuales, pero en la práctica **no funcionan** los Cursores en disparadores.

Actualmente podemos levantar un error mediante la orden SIGNAL.

```
CREATE TRIGGER chequea_precio BEFORE INSERT ON libros
FOR EACH ROW
....
        IF new.precio <= 0 then
            SIGNAL SQLSTATE '02000'                - - Error no WARNING
            SET MESSAGE_TEXT = 'El precio no puede ser inferior o igual a 0';
        END IF;
END;
....
```