Connexions module: m18780

# DISPARADORES EN BASES DE DATOS RELACIONALES\*

# Miguel-Angel Sicilia

This work is produced by The Connexions Project and licensed under the Creative Commons Attribution License  $^{\dagger}$ 

### Abstract

Se introduce el concepto de disparador (trigger) en bases de datos relacionales, y la sintaxis SQL para crearlos.

# 1 Disparadores en SQL

En ocasiones es necesario mantener restricciones en la base de datos que no pueden expresarse directamente con las sentencias de creación de tablas como CREATE TABLE. Por ejemplo, en una aplicación bancaria, si un cliente se queda sin saldo para un pago en una cuenta (es decir, entra en "números rojos"), se deberá crear automáticamente un crédito personal para el descubierto. Si en la base de datos teníamos una tabla CUENTAS con la información de las cuentas bancarias (incluyendo un atributo saldo), entonces deberíamos "observar" la tabla, y cuando se ejecutase una sentencia UPDATE sobre la misma, habría que comprobar si se cumple la condición saldo<0, en cuyo caso, habría que crear una tupla en la tabla PRESTAMOS. Esta idea de observar cambios es la que se implementa en el concepto de disparador (trigger).

Los disparadores (**triggers**) son funciones definidas por el diseñador de la base de datos que se ejecutan cuando se producen ciertas operaciones en las bases de datos. Más concretamente, son programas a los que se les da un nombre de objeto en la base de datos, se asocia con una tabla determinada, y se activa cuando ocurre un cierto tipo de evento en esa tabla, como una inserción, actualización o borrado. Si un evento ocurre, el administrador de disparadores dentro del gestor de bases de datos llama a la función adecuada para procesar el evento, por lo que cuando ejecutamos un INSERT, DELETE o UPDATE, es posible que se desencadene la ejecución de disparadores, cosa que hay que tener muy en cuenta al desarrollar y evaluar aplicaciones.

# 1.1 Un primer ejemplo

Pensemos en una aplicación bancaria en la que los CLIENTES tienen CUENTAS y PRESTAMOS. El siguiente diagrama muestra un esquema simplificado para esta base de datos.

<sup>\*</sup>Version 1.1: Dec 8, 2008 11:49 am US/Central

<sup>†</sup>http://creativecommons.org/licenses/by/2.0/

Connexions module: m18780 2

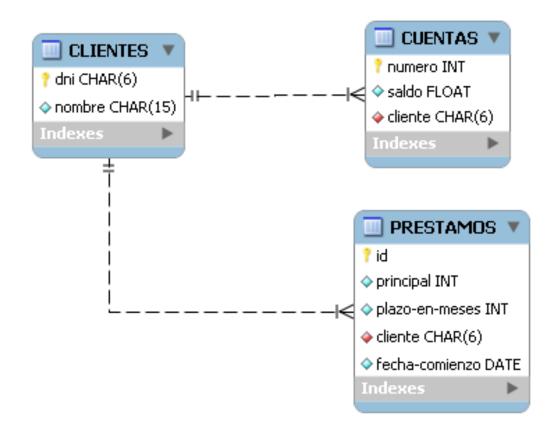


Figure 1

Inicialmente, el banco tiene la política de no permitir que un cliente quede en números rojos. Podríamos pensar en crear un disparador para esa tarea.

# 1.2 Un primer ejemplo, mejor pensado

### 1.3 Sintaxis

La sintaxis para la creación de disparadores en SQL es la siguiente:

CREATE TRIGGER nombre

momento evento

ON nombre\_tabla FOR EACH [ROW|SENTENCE] sentencia

El nombre del disparador se usará si se desea eliminar el disparador, concretamente, se usa como argumento del comando DROP TRIGGER. El resto de la sintaxis se describe a continuación:

- La palabra siguiente (momento) determina si la función debe ser llamada antes (BEFORE) o después (AFTER) del evento.
- El siguiente elemento del comando determina qué eventos dispararán la ejecución (INSERT, DELETE, UPDATE). Es posible especificar múltiples eventos utilizado el operador OR.

Connexions module: m18780 3

• El nombre de la relación (nombre\_tabla) determinará la tabla afectada por el evento. La instrucción FOR EACH determina si el disparador se ejecutará para cada fila afectada o bien antes (o después) de que la secuencia se haya completado.

• La sentencia es la definición del código que se invocará.

NOTA: En algunos gestores como MySQL 5.1 solo se soportan disparadores que se ejecutan por cada tupla afectada (FOR EACH ROW), pero otros permiten la sintaxis FOR EACH STATEMENT que se ejecutan por sentencia.

Es importante consultar la documentación del gestor de bases de datos para saber exáctamente con qué sentencias se activan los disparadores. Por ejemplo, DROP TABLE en MySQL no utiliza DELETE, por lo que no ejecuta los disparadores asociados a DELETE. Sin embargo,

Una característica importante de los disparadores es que no se puede tener más de uno para una determinada combinación de tabla, evento y momento. Es decir, no se pueden definir dos disparadores sobre la tabla T siendo ambos AFTER INSERT. Esto hace que si hay diferentes acciones a realizar para esa combinación, habrá que programar el disparador para que ejecute condicionalmente unas acciones u otras.

## 1.4 Ejecución encadenada

Si una disparador ejecuta sentencias SQL, entonces estas funciones pueden disparar otros disparadores (o dispararse a sí mismas). A esto se le llama ejecución de disparadores en cascada.

No hay ninguna limitación explicita en cuanto al número de niveles de cascada o encadenamiento. Si un disparador se ejecuta por un INSERT en una tabla T e inserta una nueva tupla en la misma tabla, se invocará de nuevo al mismo disparador (por el nuevo INSERT ejecutado en el código del disparador).

### 1.5 Funciones y procedimientos almacenados