Técnico en

DESARROLLO DE SOFTWARE

Bases de Datos II

Bases de Datos II

Unidad V

Desencadenadores (trigger)

1. Desencadenadores (Triggers)

A partir de MySQL 5.0.2 se incorporó el soporte básico para disparadores (triggers). Un disparador es un objeto con nombre dentro de una base de datos el cual se asocia con una tabla y se activa cuando ocurre en ésta un evento en particular. Por ejemplo, las siguientes sentencias crean una tabla y un disparador para sentencias INSERT dentro de la tabla. El disparador suma los valores insertados en una de las columnas de la tabla. El disparador suma los valores insertados en una de las columnas de la tabla:

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
```

Sintaxis

CREATE TRIGGER **nombre_disparador momento_disparador evento_disparador** ON **nombre_tabla** FOR EACH ROW **sentencia_disparador**

Un disparador es un objeto con nombre en una base de datos que se asocia con una tabla, y se activa cuando ocurre un evento en particular para esa tabla. El disparador queda asociado a la tabla **nombre_tabla**. Esta debe ser una tabla permanente, no puede

ser una tabla **TEMPORAL** ni una vista. **momento_disparador** es el momento en que el disparador entra en acción. Puede ser BEFORE (antes) o AFTER (después), para indicar que el disparador se ejecute antes o después que la sentencia que lo activa. **evento_disparador** indica la clase de sentencia que activa al disparador. Puede ser INSERT, UPDATE, o DELETE.

Por ejemplo, un disparador BEFORE para sentencias INSERT podría utilizarse para validar los valores a insertar. No puede haber dos disparadores en una misma tabla que correspondan al mismo momento y sentencia. Por ejemplo, no se pueden tener dos disparadores BEFORE UPDATE. Pero sí es posible tener los disparadores BEFORE UPDATE y BEFORE INSERT o BEFORE UPDATE y AFTER UPDATE. **sentencia_disparador** es la sentencia que se ejecuta cuando se activa el disparador. Si se desean ejecutar múltiples sentencias, deben colocarse entre BEGIN ... END, el constructor de sentencias compuestas. Esto además posibilita emplear las mismas sentencias permitidas en rutinas almacenadas.

Nota: Antes de MySQL 5.0.10, los disparadores no podían contener referencias directas a tablas por su nombre. A partir de MySQL 5.0.10, se pueden escribir disparadores como el llamado testref, que se muestra en este ejemplo:

```
CREATE TABLE test1(a1 INT);

CREATE TABLE test2(a2 INT);

CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY KEY);

CREATE TABLE test4(
```

```
 a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

 b4 INT DEFAULT 0

);

DELIMITER $$

CREATE TRIGGER testref BEFORE INSERT ON test1

 FOR EACH ROW BEGIN

 INSERT INTO test2 SET a2 = NEW.a1;

 DELETE FROM test3 WHERE a3 = NEW.a1;

 UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;

 END

$$

DELIMITER ;
```

INSERT INTO test3 (a3) VALUES (NULL), (NULL), (NULL), (NULL), (NULL),(NULL), (NULL), (NULL), (NULL), (NULL);

INSERT INTO test4 (a4) VALUES (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);

Si en la tabla test1 se insertan los siguientes valores:

```
mysql> INSERT INTO test1 VALUES

-> (1), (3), (1), (7), (1), (8), (4), (4);

Query OK, 8 rows affected (0.01 sec)

Records: 8 Duplicates: 0 Warnings: 0
```

2. Sintaxis para eliminar triggers

DROP TRIGGER [nombre_esquema.]nombre_disparador

Elimina un disparador. El nombre de esquema es opcional. Si el esquema se omite, el disparador se elimina en el esquema actual.

Ejecución	Evento	Toma de datos	Observaciones
BEFORE	INSERT	New.nombre_campo: campo nuevo Ejemplo dentro del trigger yo puedo hacer referencia al nuevo código así new.codigo	
ALTER	INSERT	New.nombredelcampo: campo nuevo	No se puede trabajar sobre la misma tabla
BEFORE	UPDATE	New.nombredelcampo: campo nuevo old.nombredelcampo: campo viejo. Ejemplo en el trigger se podría preguntar If(old.costo>new.costo) Se refiere a que compare el que está actualmente y el Nuevo por el que va a ser remplazado	

AFTER	UPDATE	New.nombredelcampo: campo nuevo old.nombredelcampo: campo viejo.	No se puede trabajar sobre la misma tabla
BEFORE	DELETE	old.nombredelcampo: campo viejo	
AFTER	DELETE	old.nombredelcampo: campo viejo	