Bases de Datos I
30/11/2020
Resumen Cap 23.3 30/11
IS-501
José Inestroza

23.3 Usar disparadores

Un disparador es un objeto de base de datos con nombre que está asociado con una tabla y que se activa cuando ocurre un evento en particular para la tabla. Algunos usos de los desencadenadores son para realizar comprobaciones de valores que se insertarán en una tabla o realizar cálculos sobre valores implicados en una actualización.
Un desencadenante se define para activarse cuando una declaración inserta, actualiza o elimina filas en la tabla asociada. Estas operaciones de fila son eventos desencadenantes. Por ejemplo, las filas se pueden insertar mediante INSERT o LOAD DATA declaraciones, y se activa un disparador de inserción para cada fila insertada. Se puede configurar un disparador para que se active antes o después del evento disparador. Por ejemplo, puede activar un disparador antes de cada fila que se inserta en una tabla o después de cada fila que se actualiza.

23.3.1 Sintaxis y ejemplos de disparadores

Para crear un disparador o soltar un disparador, use la instrucción CREATE TRIGGER o DROP TRIGGER, Aquí hay un ejemplo simple que asocia un disparador con una tabla, para activarlo para operaciones INSERT. El disparador actúa como un acumulador, sumando los valores insertados en una de las columnas de la tabla.

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
       FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.01 sec)
```

La instrucción CREATE TRIGGER crea un disparador llamado ins_sum que está asociado con la tabla de cuentas. También incluye cláusulas que especifican el tiempo de acción del disparador, el evento disparador y qué hacer cuando se activa el disparador:

- La palabra clave BEFORE indica el tiempo de acción del disparador. En este caso, el disparador se activa antes de cada fila insertada en la tabla. La otra palabra clave permitida aquí es AFTER.

- La palabra clave INSERT indica el evento desencadenante; es decir, el tipo de operación que activa el gatillo.
  En el ejemplo, las operaciones INSERT provocan la activación del disparador. También puede crear activadores para las operaciones DELETE

y UPDATE.

- La siguiente declaración PARA CADA FILA define el cuerpo del disparador; es decir, la instrucción que se ejecutará cada vez que se active el disparador, lo que ocurre una vez por cada fila afectada por el evento disparador. En el ejemplo, el cuerpo del disparador es un SET simple que acumula en una variable de usuario los valores insertados en la columna de cantidad. La declaración se refiere a la columna como NEW.amount, que significa "el valor de la columna de cantidad que se insertará en la nueva fila".

Para usar el disparador, establezca la variable del acumulador en cero, ejecute una instrucción INSERT y luego vea qué valor tiene la variable después:

```
mysql> SET @sum = 0;
mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-100.00);
mysql> SELECT @sum AS 'Total amount inserted';
+-----------------------+
| Total amount inserted |
+-----------------------+
|               1852.48 |
+-----------------------+
```

En este caso, el valor de @sum después de que se haya ejecutado la instrucción INSERT es 14,98 + 1937,50 - 100 o 1852,48.
Para destruir el disparador, use una instrucción DROP TRIGGER. Debe especificar el nombre del esquema si el desencadenador no está en el esquema predeterminado:

```
mysql> DROP TRIGGER test.ins_sum;
```

Si descarta una tabla, también se descartan los desencadenantes de la tabla. Los nombres de activadores existen en el espacio de nombres del esquema, lo que significa que todos los activadores deben tener nombres únicos dentro de un esquema. Los activadores en diferentes esquemas pueden tener el mismo nombre.

A partir de MySQL 5.7.2, es posible definir varios activadores para una tabla determinada que tengan el mismo evento de activación y tiempo de acción. Por ejemplo, puede tener dos activadores ANTES DE ACTUALIZAR para una tabla. Por defecto, los desencadenantes que tienen el mismo evento desencadenante y tiempo de acción se activan en el orden en que fueron creados. Para afectar el orden de activación, especifique una cláusula después de PARA CADA FILA que indique FOLLOWS o PRECEDES y el nombre de un activador existente que también tiene el mismo evento de activación y tiempo de acción. Con FOLLOWS, el nuevo disparador se activa después del disparador existente. Con PRECEDES, el nuevo disparador se activa antes que el existente. Por ejemplo, la siguiente

definición de activador define otro activador ANTES DE INSERTAR para la cuenta mesa:

```
mysql> CREATE TRIGGER ins_transaction BEFORE INSERT ON account
       FOR EACH ROW PRECEDES ins_sum
       SET
       @deposits = @deposits + IF(NEW.amount>0,NEW.amount,0),
       @withdrawals = @withdrawals + IF(NEW.amount<0,-NEW.amount,0);
Query OK, 0 rows affected (0.01 sec)
```

Este disparador, ins_transaction, es similar a ins_sum pero acumula depósitos y retiros por separado. Tiene una cláusula PRECEDES que hace que se active antes de ins_sum; sin esa cláusula, se activaría después de ins_sum porque se crea después de ins_sum.

Antes de MySQL 5.7.2, no puede haber varios activadores para una tabla determinada que tengan el mismo evento de activación y tiempo de acción. Por ejemplo, no puede tener dos activadores ANTES DE ACTUALIZAR para una tabla. Para trabajar alrededor
Esto, puede definir un disparador que ejecute múltiples sentencias usando la construcción de sentencias compuestas BEGIN ... END después de FOR EACH ROW. (Un ejemplo aparece más adelante en esta sección). Dentro del cuerpo del disparador, las palabras clave OLD y NEW le permiten acceder a las columnas en las filas afectadas por un disparador. OLD y NEW son extensiones de MySQL para disparadores; no distinguen entre mayúsculas y minúsculas. En un disparador INSERT, solo se puede usar NEW.col_name; no hay fila antigua. En un disparador DELETE, solo se puede usar OLD.col_name; no hay nueva fila. En un desencadenador UPDATE, puede usar OLD.col_name para hacer referencia a las columnas de una fila antes de que se actualice y NEW.col_name para hacer referencia a las columnas de la fila después de que se actualice. Una columna denominada OLD es de solo lectura. Puede consultarlo (si tiene el privilegio SELECT), pero no modificarlo. Puede hacer referencia a una columna nombrada con NUEVO si tiene el privilegio SELECT para ello. En un disparador ANTES, también puede cambiar su valor con SET NEW.col_name = value si tiene la ACTUALIZACIÓN privilegio por ello. Esto significa que puede usar un disparador para modificar los valores que se insertarán en una nueva fila o se usarán para actualizar una fila. (Dicha instrucción SET no tiene efecto en un desencadenador AFTER porque el cambio de fila ya habrá ocurrido). En un desencadenador BEFORE, el valor NUEVO para una columna AUTO_INCREMENT es 0, no el número de secuencia que se genera automáticamente cuando la nueva fila realmente se inserta. Al utilizar la construcción BEGIN ... END, puede definir un desencadenante que ejecute varias declaraciones. Dentro del bloque BEGIN, también puede utilizar otra sintaxis permitida dentro de las rutinas almacenadas, como condicionales y bucles. Sin embargo, al igual que para las rutinas almacenadas, si usa el programa mysql para definir un disparador que ejecuta múltiples declaraciones, es necesario redefinir el delimitador de declaraciones mysql para que pueda usar el; delimitador de declaración dentro de la definición del disparador. El siguiente ejemplo ilustra estos puntos. Define un disparador

UPDATE que verifica el nuevo valor que se utilizará para actualizar cada fila y modifica el valor para que esté dentro del rango de 0 a 100. Este debe ser un disparador ANTES porque el valor debe comprobarse antes de que se utilice para actualizar La fila:

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON account
       FOR EACH ROW
       BEGIN
           IF NEW.amount < 0 THEN
               SET NEW.amount = 0;
           ELSEIF NEW.amount > 100 THEN
               SET NEW.amount = 100;
           END IF;
       END;//
mysql> delimiter ;
```

Puede ser más fácil definir un procedimiento almacenado por separado y luego invocarlo desde el desencadenador usando una simple instrucción CALL. Esto también es ventajoso si desea ejecutar el mismo código desde varios disparadores.
Existen limitaciones sobre lo que puede aparecer en las declaraciones que ejecuta un disparador cuando se activa:

1. El desencadenante no puede utilizar la sentencia CALL para invocar procedimientos almacenados que devuelven datos al cliente o que utilizan SQL dinámico. (Los procedimientos almacenados pueden devolver datos al disparador a través de los parámetros OUT o INOUT).

2. El desencadenante no puede utilizar declaraciones que inicien o finalicen una transacción de forma explícita o implícita, como START TRANSACTION, COMMIT o ROLLBACK. (ROLLBACK a SAVEPOINT está permitido porque no finaliza una transacción).

MySQL maneja los errores durante la ejecución del disparador de la siguiente manera:

- Si falla un disparador ANTES, no se realiza la operación en la fila correspondiente.

- Un disparador ANTES se activa al intentar insertar o modificar la fila, independientemente de si el intento tiene éxito posteriormente.

- Un disparador DESPUÉS se ejecuta solo si se dispara ANTES y la operación de fila se ejecuta correctamente.

- Un error durante un activador ANTES o DESPUÉS da como resultado el fallo de toda la declaración que provocó la invocación del activador.

- En el caso de tablas transaccionales, la falla de una declaración debería provocar la reversión de todos los cambios realizados por la declaración. La falla de un disparador hace que la declaración falle, por lo que la falla del disparador también causa una reversión. Para las tablas no transaccionales, dicha reversión no se puede realizar, por lo que, aunque la instrucción falla, cualquier cambio realizado antes del punto del error permanece en vigor.

Los desencadenadores pueden contener referencias directas a tablas por nombre, como el desencadenador 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);
```

Suponga que inserta los siguientes valores en la tabla test1 como se muestra aquí:

```
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
```

Como resultado, las cuatro tablas contienen los siguientes datos:

```
mysql> SELECT * FROM test1;
+------+
| a1   |
+------+
|    1 |
|    3 |
|    1 |
|    7 |
|    1 |
|    8 |
|    4 |
|    4 |
+------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM test2;
+------+
| a2   |
+------+
|    1 |
|    3 |
|    1 |
|    7 |
|    1 |
|    8 |
|    4 |
|    4 |
+------+
8 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM test3;
+----+
| a3 |
+----+
|  2 |
|  5 |
|  6 |
|  9 |
| 10 |
+----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM test4;
+----+------+
| a4 | b4   |
+----+------+
|  1 |    3 |
|  2 |    0 |
|  3 |    1 |
|  4 |    2 |
|  5 |    0 |
|  6 |    0 |
|  7 |    1 |
|  8 |    1 |
|  9 |    0 |
| 10 |    0 |
+----+------+
10 rows in set (0.00 sec)
```