Eventos en MySQL. Triggers After Delete.

Introducción a los disparadores AFTER DELETE de MySQL

Los disparadores de MySQL AFTER DELETE se invocan automáticamente después de que se produce un evento de eliminación en la tabla.
Aquí está la sintaxis básica para crear un disparador MySQL AFTER DELETE:

```
CREATE TRIGGER trigger_name
    AFTER DELETE
    ON table_name FOR EACH ROW
trigger_body;
```

Las sintaxis nos muestra:
Primero, el nombre del trigger que desea crear en la cláusula CREATE TRIGGER.
En segundo lugar, la cláusula AFTER DELETE para especificar el momento en que se invoca el disparador.
En tercer lugar, el nombre de la tabla, con la que está asociado el trigger, después de la palabra clave ON.
Finalmente, el cuerpo del disparador que contiene una o más instrucciones que se ejecutan cuando se invoca el disparador.
Si el cuerpo del disparador tiene varias instrucciones,se deben usar BEGIN END para delimitar el bloque, y cambiar el delimitador predeterminado de ";.ª "$$"(por ejemplo), como se muestra a continuación:

```
DELIMITER $$

CREATE TRIGGER trigger_name
    AFTER DELETE
    ON table_name FOR EACH ROW
BEGIN
    -- statements
END$$

DELIMITER ;
```

En un trigger AFTER DELETE, se puede acceder a las filas OLD, pero no se pueden modificar. Hay que tener en cuenta que no existen filas NEW en este tipo de trigger.

Ejemplo de trigger AFTER DELETE en MySQL

Veamos el siguiente ejemplo de trigger AFTER DELETE

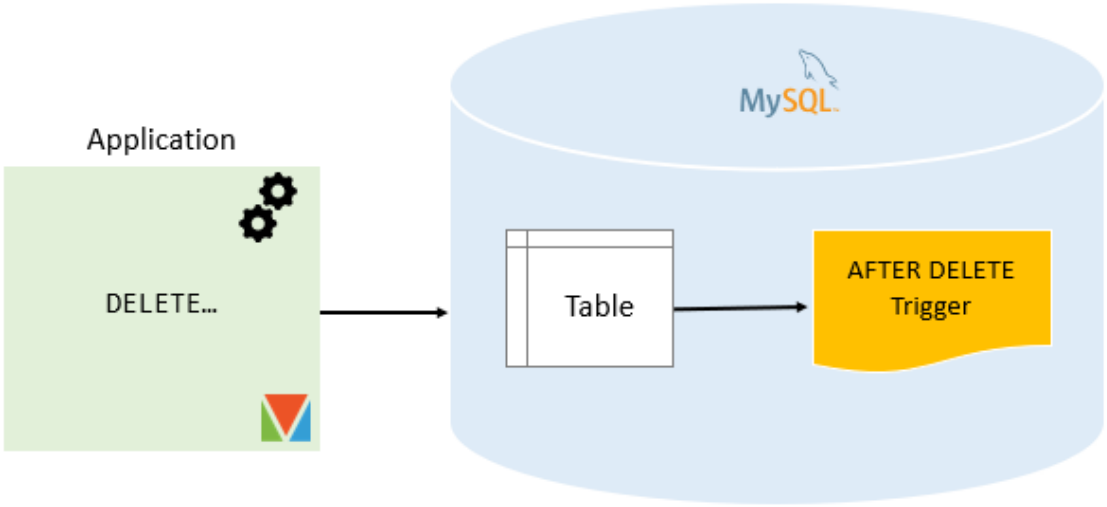

Figura 1: MySQL AFTER DELETE Trigger

Preparar la tabla con la que vamos a trabajar

Primero, cree una nueva tabla llamada **Salaries** (Salarios):

```
DROP TABLE IF EXISTS Salaries;

CREATE TABLE Salaries (
    employeeNumber INT PRIMARY KEY,
    salary DECIMAL(10,2) NOT NULL DEFAULT 0
);
```

En segundo lugar insertamos varias filas en la nueva tabla de salarios:

```
INSERT INTO Salaries(employeeNumber,salary)
VALUES
    (1002,5000),
    (1056,7000),
    (1076,8000);
```

Tercero, cree otra tabla llamada **SalaryBudgets** que almacene la suma total de los salarios de la tabla **Salaries**:

```
DROP TABLE IF EXISTS SalaryBudgets;

CREATE TABLE SalaryBudgets(
    total DECIMAL(15,2) NOT NULL
);
```

Cuarto, uso la función SUM() para obtener el salario total de la tabla **Salaries** e insertarlo en la tabla **SalaryBudgets**:

```
INSERT INTO SalaryBudgets(total)
SELECT SUM(salary)
FROM Salaries;
```

Finalmente, consulta los datos de la tabla **SalaryBudgets**:

```
SELECT * FROM SalaryBudgets;
```

Creación del trigger AFTER DELETE de nuestro ejemplo:

El siguiente disparador AFTER DELETE actualiza el salario total en la tabla **SalaryBudgets** después de que se elimina una fila de la tabla **Salaries** (Salarios):

```
CREATE TRIGGER after_salaries_delete
AFTER DELETE
ON Salaries FOR EACH ROW
UPDATE SalaryBudgets
SET total = total - old.salary;
```

En este disparador:
Primero, el nombre del trigger es **after_salaries_delete** especificado en la cláusula CREATE TRIGGER:

```
CREATE TRIGGER after_salaries_delete
```

En segundo lugar, el evento desencadenante es:

```
AFTER DELETE
```

En tercer lugar, la tabla con la que está asociado el trigger es la tabla **Salaries**:

```
ON Salaries FOR EACH ROW
```

Finalmente, dentro del cuerpo del trigger, restamos el salario eliminado del salario total.

Probar el disparador MySQL AFTER DELETE

Primero, elimine una fila de la tabla Salarios:

```
DELETE FROM Salaries
WHERE employeeNumber = 1002;
```

En segundo lugar, consulte el salario total de la tabla SalaryBudgets:

```
SELECT * FROM SalaryBudgets;
```

Como puedes ver en el resultado, el **total** se reduce en una cantidad que se corresponde con el **salario eliminado**.
Tercero, eliminar todas las filas de la tabla **Salaries**:

```
DELETE FROM Salaries;
```

Finalmente, consulte el total de la tabla **SalaryBudgets**:

```
SELECT * FROM SalaryBudgets;
```

El disparador actualizó el total a cero.