Eventos en MySQL. Triggers.

MySQL. Introducción a los Before Insert Trigger

Los disparadores MySQL BEFORE INSERT se disparan automáticamente antes de que ocurra un evento de inserción en la tabla.
A continuación se ilustra la sintaxis básica de la creación de un trigger MySQL BEFORE INSERT:

```
CREATE TRIGGER trigger_name
    BEFORE INSERT
    ON table_name FOR EACH ROW
trigger_body;
```

En la sintaxis vemos que:
Primero, se especifica el nombre del disparador que desea crear en la cláusula CREATE TRIGGER.
En segundo lugar, el uso de la cláusula BEFORE INSERT para especificar el tiempo para invocar el trigger.
En tercer lugar, se especifica el nombre de la tabla con la que está asociado el trigger después de la palabra clave ON.
Finalmente, vien el cuerpo del trigger que contiene una o más instrucciones SQL que se ejecutan cuando se invoca el trigger.
Si tiene varias declaraciones en el trigger_body, debe usar el bloque BEGIN END y cambiar el delimitador predeterminado:

```
DELIMITER $$

CREATE TRIGGER trigger_name
    BEFORE INSERT
    ON table_name FOR EACH ROW
BEGIN
    -- statements
END$$

DELIMITER ;
```

Tenga en cuenta que en un disparador BEFORE INSERT, puede acceder y cambiar los valores NEW (nuevos). Sin embargo, no puede acceder a los valores OLD porque, obviamente, los valores OLD (viejos) no existen.

Ejemplo de disparador MySQL BEFORE INSERT

Crearemos un disparador BEFORE INSERT para mantener una tabla de resumen de otra tabla.
Primero, cree una nueva tabla llamada WorkCenters:

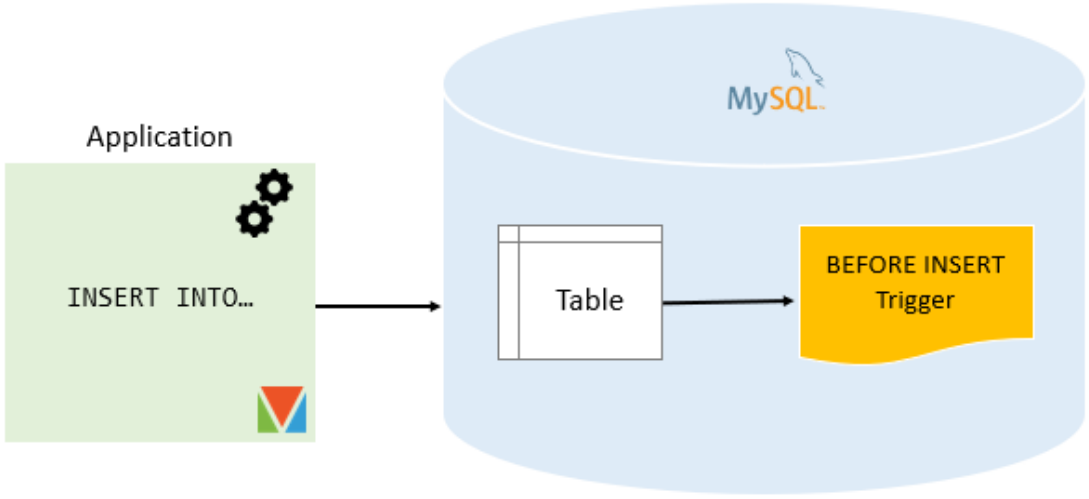

Figura 1: MySQL BEFORE INSERT Triggers

```
DROP TABLE IF EXISTS WorkCenters;

CREATE TABLE WorkCenters (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    capacity INT NOT NULL
);
```

En segundo lugar, cree otra tabla llamada WorkCenterStats que almacene el resumen de la capacidad de los centros de trabajo:

```
DROP TABLE IF EXISTS WorkCenterStats;

CREATE TABLE WorkCenterStats(
    totalCapacity INT NOT NULL
);
```

El siguiente trigger actualiza la capacidad total en la tabla WorkCenterStats antes de insertar un nuevo centro de trabajo en la tabla WorkCenter:

```
DELIMITER $$

CREATE TRIGGER before_workcenters_insert
BEFORE INSERT
ON WorkCenters FOR EACH ROW
BEGIN
    DECLARE rowcount INT;

    SELECT COUNT(*)
    INTO rowcount
    FROM WorkCenterStats;
```

```
    IF rowcount > 0 THEN
        UPDATE WorkCenterStats
        SET totalCapacity = totalCapacity + new.capacity;
    ELSE
        INSERT INTO WorkCenterStats(totalCapacity)
        VALUES(new.capacity);
    END IF;

END $$

DELIMITER ;
```

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

```
CREATE TRIGGER before_workcenters_insert
```

En segundo lugar, el evento disparador es:

```
BEFORE INSERT
```

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

```
ON WorkCenters FOR EACH ROW
```

Finalmente, dentro del cuerpo del trigger, verificamos si hay alguna fila en la tabla WorkCenterStats.
Si la tabla WorkCenterStats tiene una fila, el trigger agrega la capacidad a la columna totalCapacidad. De lo contrario, inserta una nueva fila en la tabla WorkCenterStats.

Prueba del disparador MySQL BEFORE INSERT

Primero, inserte una nueva fila en la tabla WorkCenter:

```
INSERT INTO WorkCenters(name, capacity)
VALUES('Mold Machine',100);
```

En segundo lugar, consulte los datos de la tabla WorkCenterStats:

```
SELECT * FROM WorkCenterStats;
```

Vemos que se ha invocado el desencadenador insertó una nueva fila en la tabla WorkCenterStats.
Tercero, inserte un nuevo centro de trabajo:

```
INSERT INTO WorkCenters(name, capacity)
VALUES('Packing',200);
```

Finalmente, consulte los datos de WorkCenterStats:

```
SELECT * FROM WorkCenterStats;
```

El disparador ha actualizado la capacidad total de 100 a 200 como se esperaba.
Tenga en cuenta que para mantener correctamente la tabla de resumen WorkCenterStats, también debe crear disparadores para gestionar la actualización y eliminar eventos en la tabla WorkCenters.
En este tutorial, ha aprendido cómo crear un trigger MySQL ANTES DE INSERTAR para mantener una tabla resumen de otra tabla.