Eventos en MySQL. Triggers After Insert.

Introducción a los disparadores MySQL AFTER INSERT

Los disparador de MySQL AFTER INSERT se invocan automáticamente después de que ocurre un evento de inserción en la tabla.
A continuación se muestra la sintaxis básica de la creación de un trigger MySQL AFTER INSERT:

```
CREATE TRIGGER trigger_name
    AFTER INSERT
    ON table_name FOR EACH ROW
        trigger_body
```

En la sintaxis podemos ver que:
Primero, se especifica el nombre del disparador que desea crear después de las palabras clave CREATE TRIGGER.
En segundo lugar, la cláusula AFTER INSERT para especificar el tiempo en el que invocar el disparador.
En tercer lugar, el nombre de la tabla en la que desea crear el trigger, después de la palabra clave ON.
Finalmente, tenemos el cuerpo del disparador que consiste en una o más declaraciones que se ejecutan cuando se invoca el disparador.
En caso de que el cuerpo del disparador tenga varias declaraciones, debe usar el bloque BEGIN END y cambiar el delimitador predeterminado:

```
DELIMITER $$

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

DELIMITER ;
```

En un disparador AFTER INSERT, puede acceder a los valores NUEVOS pero no puede cambiarlos. Además, no puede acceder a los valores OLD porque no hay OLD en los triggeres INSERT.

Ejemplo de disparador, MySQL, AFTER INSERT

Considere el siguiente ejemplo de disparador AFTER INSERT.

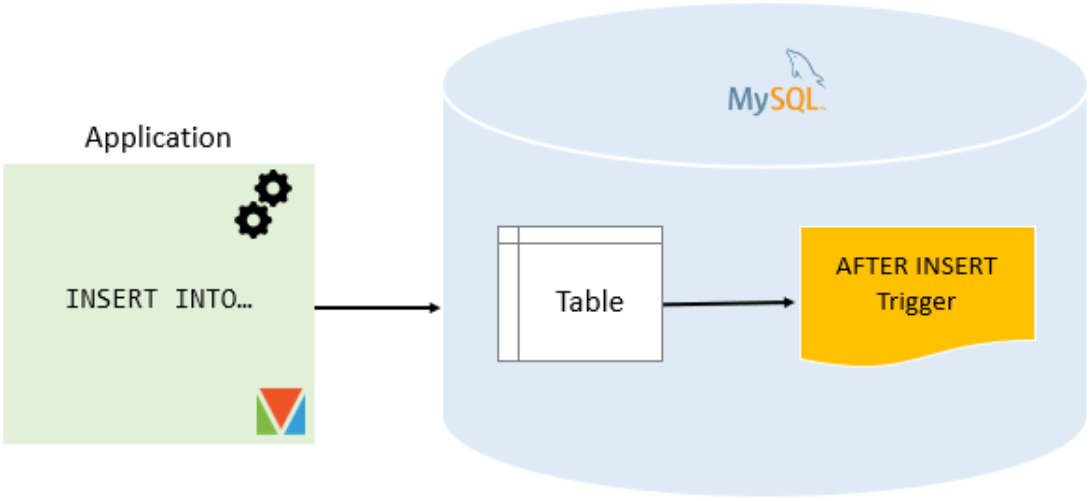

Figura 1: MySQL AFTER INSERT Triggers

Preparar una tabla de ejemplo

Primero, cree una nueva tabla llamada miembros:

```
DROP TABLE IF EXISTS members;

CREATE TABLE members (
    id INT AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255),
    birthDate DATE,
    PRIMARY KEY (id)
);
```

Segundo, cree otra tabla llamada recordatorios que almacene mensajes recordatorios para los miembros.

```
DROP TABLE IF EXISTS reminders;

CREATE TABLE reminders (
    id INT AUTO_INCREMENT,
    memberId INT,
    message VARCHAR(255) NOT NULL,
    PRIMARY KEY (id , memberId)
);
```

Ejemplo de creación de un Trigger AFTER INSERT

La siguiente declaración crea un disparador AFTER INSERT que inserta un recordatorio (reminder) en la tabla reminders (recordatorios) si la fecha de nacimiento del miembro es NULL.

```
DELIMITER $$
```

```
CREATE TRIGGER after_members_insert
AFTER INSERT
ON members FOR EACH ROW
BEGIN
    IF NEW.birthDate IS NULL THEN
        INSERT INTO reminders(memberId, message)
        VALUES(new.id,CONCAT('Hi ', NEW.name, ', please update your date of birth.'));
    END IF;
END$$

DELIMITER ;
```

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

```
CREATE TRIGGER after_members_insert
```

En segundo lugar, el evento disparador es:

```
AFTER INSERT
```

En tercer lugar, la tabla con la que está asociado el disparador es la tabla members:

```
ON members FOR EACH ROW
```

Finalmente, dentro del cuerpo del disparador, inserte una nueva fila en la tabla **reminders** si la fecha de nacimiento del miembro es NULL.

Prueba del trigger MySQL AFTER INSERT

Primero, inserte dos filas en la tabla members:

```
INSERT INTO members(name, email, birthDate)
VALUES
    ('John Doe', 'john.doe@example.com', NULL),
    ('Jane Doe', 'jane.doe@example.com','2000-01-01');
```

En segundo lugar, consulta los datos de la tabla members:

```
SELECT * FROM members;
```

Tercero, consultar datos de la tabla *reminders*:

```
SELECT * FROM reminders;
```

Hemos insertado dos filas en la tabla members. Sin embargo, solo la primera fila tiene un valor de fecha de nacimiento NULL, por lo tanto, el disparador solo ha insertado una fila en la tabla reminders.