**Практическая работа № 17**

**Создание триггеров для таблицы в MySQL**

Триггер является именованным объектом базы данных, который связан с таблицей, и активизируется, когда специфическое событие происходит для этой таблицы. Это очень здорово, когда Вы должны выполнить проверки значений, которые будут вставлены в таблицу или выполнять вычисления на значениях, включаемых в модификации.

Триггер связан с таблицей и определен, чтобы активизироваться, когда для таблицы выполняется инструкция INSERT, DELETE или UPDATE. Триггер может быть установлен, чтобы активизироваться прежде или после вызова инструкции. Например, Вы можете иметь триггер, срабатывающий перед удалением каждой строки из таблицы, или после каждой модификации уже существующей строки в таблице.

**Синтаксис создания триггеров**

**CREATE TRIGGER trigger\_name trigger\_time trigger\_event**

**ON tbl\_name FOR EACH ROW trigger\_stmt;**

Триггер становится связанным с таблицей с именем **tbl\_name**, которое должно обратиться к постоянной таблице. Вы не можете связывать триггер с view или таблицей TEMPORARY.

**trigger\_time** задает время действия. Это может быть BEFORE или AFTER, чтобы задать, что триггер активизируется прежде или после инструкции, которая активизировала это.

**trigger\_event** указывает вид инструкции, которая активизирует триггер. Здесь **trigger\_event** может быть одним из следующего:

INSERT: всякий раз, когда новая строка вставлена в таблицу. Например, через команды INSERT, LOAD DATA или REPLACE.

UPDATE: всякий раз, когда строка изменяется. Например, через инструкцию UPDATE.

DELETE: всякий раз, когда строка удалена из таблицы. Например, через инструкции DELETE и REPLACE. Однако, инструкции DROP TABLE и TRUNCATE относительно таблицы *НЕ* активизируют триггер, потому что они не используют DELETE!

Важно понять, что **trigger\_event** не представляет литеральный тип инструкции SQL, которая активизирует триггер, поскольку это представляет тип операции таблицы. Например, триггер INSERT активизирован не только инструкцией INSERT, но и LOAD DATA, потому что обе инструкции вставляют строки в таблицу.

Не может быть двух триггеров для данной таблицы, которые имеют те же самые время действия и событие. Например, Вы не можете иметь два триггера BEFORE UPDATE для таблицы. Но Вы можете иметь BEFORE UPDATE и BEFORE INSERT или BEFORE UPDATE и AFTER UPDATE.

**trigger\_stmt** задает инструкцию, которая будет выполнена, когда триггер активизируется. Если Вы хотите выполнять много инструкций, используйте операторную конструкцию BEGIN ... END. Это также дает возможность Вам использовать те же самые инструкции, которые являются допустимыми внутри сохраненных подпрограмм.

Примеры использования триггеров

**Пример 1**

mysql> CREATE TABLE account (acct\_num INT, amount float(10,2));

mysql> CREATE TRIGGER ins\_sum BEFORE INSERT ON account

-> FOR EACH ROW SET @sum = @sum + NEW.amount;

Команда CREATE TRIGGER создает триггер ins\_sum, который связан с таблицей account. Это также включает предложения, которые определяют время активации, событие вызова, и что делать с активированным триггером дальше:

Ключевое слово BEFORE указывает время срабатывания. В этом случае триггер должен активизировать перед каждой строкой, вставленной в таблицу. Другое допустимое ключевое слово здесь: AFTER.

Ключевое слово INSERT указывает событие, которое активизирует триггер. В этом примере триггер срабатывает от инструкции INSERT. Вы можете также создавать триггеры для инструкций DELETE и UPDATE.

Инструкция FOR EACH ROW определяет, что триггер должен сработать один раз для каждой строки, на которую воздействует инструкция в примере. Собственно триггер представляет собой в данном случае простой SET, который накапливает значения, вставленные в столбец amount. Инструкция обращается к столбцу как NEW.amount, что означает "значение столбца amount, которое будет вставлено в новую строку".

Чтобы использовать триггер, установите переменную сумматора в ноль, выполните инструкцию INSERT, а затем посмотрите то, какое значение переменная имеет позже:

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';

Результат работы триггера:

+-----------------------+

| Total amount inserted |

+-----------------------+

| 1852.48 |

+-----------------------+

В этом случае значение @sum после выполнения команды INSERT равно 14.98 + 1937.50 - 100 или 1852.48.

**Пример 2**

Используя конструкцию **BEGIN** ... **END**, Вы можете определять триггер, который выполняет много инструкций.

Вы можете обратиться к столбцам в подчиненной таблице, используя псевдонимы **OLD** и **NEW**.

**OLD.*col\_name*** обращается к столбцу существующей строки прежде, чем она модифицируется или удалится.

**NEW.*col\_name*** обращается к столбцу новой строки, которая будет вставлена, или же к существующей строке после того, как она модифицируется.

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);

Предположим, что Вы вставляете следующие значения в таблицу test1 как показано здесь:

mysql> INSERT INTO test1 VALUES

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

В результате данные в четырех таблицах будут следующие:

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)

**Практическая часть:**

1. Удалить связь между таблицами Продавцы и Заказчики.
2. В PHPMyAdmin отредактировать связи между таблицами Продавцы-Заказы и Заказчики-Заказы. Сделать, чтобы удаление и обновление данных проходило каскадно.
3. При внесении в базу данных нового Заказчика, ему автоматически присваивается Заказ (значение Заказа на Ваше усмотрение), который будет обслуживать Продавец Peel.
4. Создать триггер на удаление продавца, который предварительно перепишет его заказы на продавца «Axelord». Удалить продавца «Peel».
5. Создать триггер на изменение Заказа, который изменит город его продавца на «Москва». Изменить сумму заказа 3009.