**Триггеры** - это особые процедуры которые срабатывают при изменении данных в таблицы операторами INSERT, UPDATE и DELETE.

Триггеры могут срабатывать как до BEFORE так и после AFTER изменения таблицы. Получается, что триггер может быть в 6 состояниях.

|  |  |
| --- | --- |
| 1  2  3 | INSERT (BEFORE | AFTER)  UPDATE (BEFORE | AFTER)  DELETE (BEFORE | AFTER) |

**Синтаксис**

CREATE

[DEFINER = {*user* | CURRENT\_USER}]

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.

Когда триггер активизирован, предложение DEFINER определяет привилегии, которые применяются, как описано ниже в этом разделе.

***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 иREPLACE. Однако, инструкции DROP TABLE и TRUNCATE относительно таблицы *НЕ* активизируют триггер, потому что они не используют DELETE!

**Добавление триггера**

Для того чтобы понять работу триггера необходим пример.

Задание

1. Создать таблицу user, включающую поля id, fam, name, otch, pass
2. Необходимо при добавлении записи в табл user, пароль преобразовывать в хеш md5(), также имя и отчество преобразовывать в инициалы.

Решение

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | mysql> DELIMITER //  mysql> CREATE TRIGGER `test\_user\_pass` BEFORE INSERT ON `test`.`user`      -> FOR EACH ROW      -> BEGIN      ->  SET NEW.name = LEFT(NEW.name,1);      ->  SET NEW.otch = LEFT(NEW.otch,1);      ->  SET NEW.pass = md5(NEW.pass);      -> END//  Query OK, 0 rows affected (0.09 sec)    mysql> DELIMITER ; |

Теперь вставляем туда запись

|  |  |
| --- | --- |
| 1  2 | mysql> INSERT  INTO `user` SET `fam`='Нагайченко', `name`='Максим',  `otch` = 'Валерьевич', `pass` = 'password', `login` = 'maxnag';  Query OK, 1 row affected (0.00 sec) |

Что теперь в таблице

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | mysql> SELECT \* FROM `user`;  +----+-------------+------+------+----------------------------------+--------+  | id | fam         | name | otch | pass                             | login  |  +----+-------------+------+------+----------------------------------+--------+  |  1 | Нагайченко  | M    | В    | 5f4dcc3b5aa765d61d8327deb882cf99 | maxnag |  +----+-------------+------+------+----------------------------------+--------+    1 row in set (0.00 sec) |

Как видно всего несколькими строчками можно убрать целые методы, которые мы использовали при регистрации нового пользователя. Теперь еще надо создать триггер на UPDATE таблицы, с таким же телом, чтобы пользователь не смог записать полное имя, отчество и пароль не в МД5();

**Создание триггера на BEFORE UPDATE**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | mysql> DELIMITER //  mysql> CREATE TRIGGER `test\_user\_pass2` BEFORE UPDATE ON `test`.`user`      -> FOR EACH ROW      -> BEGIN      ->  SET NEW.name = LEFT(NEW.name,1);      ->  SET NEW.otch = LEFT(NEW.otch,1);      ->  SET NEW.pass = md5(NEW.pass);      -> END//  Query OK, 0 rows affected (0.09 sec)    mysql> DELIMITER ; |

**Обновление записи**

|  |  |
| --- | --- |
| 1  2 | mysql> INSERT  INTO `user` SET `fam`='Нагайченко', `name`='Максим',  `otch` = 'Валерьевич', `pass` = 'password', `login` = 'maxnag';  Query OK, 1 row affected (0.00 sec) |

Итог

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | mysql> SELECT \* FROM `user`;  +----+-------------+------+------+----------------------------------+--------+  | id | fam         | name | otch | pass                             | login  |  +----+-------------+------+------+----------------------------------+--------+  |  1 | Иванов      | И    | И    | e242f36f4f95f12966da8fa2efd59992 | ivan   |  +----+-------------+------+------+----------------------------------+--------+    1 row in set (0.00 sec) |

Ключевые слова OLD и NEW дают возможность Вам обратиться к столбцам в строках, на которые воздействует триггер OLD и NEW не чувствительны к регистру. В триггере INSERT может использоваться только NEW.*col\_name*: не имеется никакой старой строки. В триггере DELETE не ожидается никакой новой строки, так что может использоваться исключительно OLD.*col\_name*. В триггере UPDATE Вы можете использовать OLD.*col\_name*, чтобы обратиться к столбцам строки прежде, чем они изменятся, иNEW.*col\_name* , чтобы обратиться к ним уже после внесения изменений.

**Удаление триггера**

Для удаления триггера используется, как обычно оператор DROP, пример

|  |  |
| --- | --- |
| 1 | DROP TRIGGER [IF EXISTS] [schema\_name.]trigger\_name |

Где

schema\_name — название БД,

trigger\_name — название триггера

**Список созданных триггеров**

Показать триггер можно с помощью команды

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SHOW TRIGGERS [{FROM | IN} db\_name]      [LIKE 'pattern' | WHERE expr]    Это полная часть команды, в основном пользуются командами:    SHOW TRIGGERS [FROM db\_name] [LIKE 'pattern']  или  SHOW TRIGGERS |

Это полная часть команды, в основном пользуются командами:

|  |  |
| --- | --- |
| 1  2  3 | SHOW TRIGGERS [FROM db\_name] [LIKE 'pattern']  или  SHOW TRIGGERS |

Вот что мы увидем, когда выполним команду

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18 | mysql> SHOW TRIGGERS\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*               Trigger: test\_user\_pass                 Event: INSERT                 Table: user             Statement: BEGIN                            SET NEW.name = LEFT(NEW.name,1);                            SET NEW.otch = LEFT(NEW.otch,1);                            SET NEW.pass = md5(NEW.pass);                        END                Timing: BEFORE               Created: NULL              sql\_mode: STRICT\_TRANS\_TABLES,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION               Definer: root@localhost  character\_set\_client: utf8  collation\_connection: utf8\_general\_ci    Database Collation: utf8\_unicode\_ci  1 row in set (0.01 sec) |

Рассмотрим ещё один пример работы триггера:

Имеется простой пример, который связывает триггер с таблицей для инструкций INSERT. Он действует как сумматор, чтобы суммировать значения, вставленные в один из столбцов таблицы.

Следующие инструкции создают таблицу и триггер для нее:

mysql> CREATE TABLE account (acct\_num INT, amount DECIMAL(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 инструкцийDELETE 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';

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

| Total amount inserted |

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

| 1852.48 |

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

В этом случае значение @sum после выполнения команды INSERT равно 14.98 + 1937.50 - 100 или1852.48.

Используя конструкцию BEGIN ... END, Вы можете определять триггер, который выполняет много инструкций. Внутри блока BEGIN Вы также можете использовать другой синтаксис, который позволяется внутри сохраненных подпрограмм, типа условных выражений и циклов. Однако, точно как для сохраненных подпрограмм, если Вы используете программу **mysql**, чтобы определить триггер, необходимо переопределить операторный разделитель **mysql** так, чтобы Вы могли использовать ; как операторный разделитель внутри описания триггера. Следующий пример иллюстрирует эти моменты. Это определяет триггер UPDATE, который проверяет новое значение, которое нужно использовать для модифицирования каждой строки, и изменяет значение, чтобы оставаться внутри диапазона от 0 до 100. Это должно быть триггером типа BEFORE, потому что значение должно быть проверено прежде, чем оно используется, чтобы модифицировать строку:

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 ;

Может быть проще определить сохраненную процедуру отдельно и затем вызвать ее из триггера простой инструкцией CALL. Это также выгодно, если Вы хотите вызывать ту же самую подпрограмму изнутри нескольких триггеров.

Запросы, созданные по примерам и результат их выполнения отправить на электронную почту **olga.bogdanova@vvsu.ru**