CIS 363 MySQL

Chapter 19 Database Triggers

Ch. 19 Triggers

- A trigger is a database object that is associated with a table and that is defined to activate when a particular kind f event occurs for that table.
- A trigger provides a means to execute an SQL statement or set of statements when you insert, update or delete rows in a table.
- Triggers provide the following benefits:
- A trigger can examine row values to be inserted or updated, and it can determine what values were deleted or what they were updated to.
- A trigger can change values before they are inserted into a table or used to updated a table.
- A trigger can help to modify how INSERT, DELETE, or UPDATE work.

Ch. 19 Triggers

Trigger Concepts

- A trigger is an object that belongs to a database. Each trigger within the database must have a different name.
- A trigger is defined to activate when a particular kind of event occurs for a given table. The events for which trigger can be defined are INSERT, DELETE, and UPDATE. A given trigger is defined for only one of these events, but you can define multiple triggers for a table, **one trigger per type of event.**
- Triggers can be defined to activate either **BEFORE** or **after** the event. This means there can be two triggers per event. (There are 6 triggers max. for a table.)
- Triggers may be used to supplement integrity constraints, enforce complex business rules, or to audit changes to the database. Triggers should be used to enforce rules that CANNOT be enforced through referential integrity.

Ch. 19 Triggers

Valid Trigger Types:

BEFORE INSERT row

AFTER INSERT row

BEFORE UPDATE row

AFTER UPDATE row

BEFORE DELETE row

AFTER DELETE row

ORDER OF TRIGGER FIRING:

Before Row

Execute of SQL Statement.

After Row

Ch. 19 Triggers

Trigger Syntax

```
mysql> CREATE  TRIGGER trigger_name
        {BEFORE|AFTER} {INSERT|DELETE|UPDATE}
        ON table_name
        FOR EACH ROW
        BEGIN
        Statements;
        End;
```

- The triggered statement must be a single statement, but if necessary you can use a BEGIN/END compound statement to create a block and include multiple statements within the block. Each statement must be terminated by semicolon character(';') within the block.
- If you are using the mysql client to create such a trigger, you must redefine the statement delimiter.

Ch. 19 Triggers

Example:

```
mysql> CREATE TABLE debit_balance_report (action_dt TIMESTAMP,
    customer_number INT);

/* debit_balance: retrieve customer balance.
        @author: rtimlin
        @original: 06-Mar-2001
        @updated:
      Table:
        @version: 1.0
              ****  Modification History  ****
  Date       User       Description
        ======================================
        3/6/01   rtimlin      Original.
        3/8/01   flast         Added Exception handler.
    */
```

Ch. 19 Triggers

```
mysql> DELIMITER //

mysql> CREATE TRIGGER debit_balance
       BEFORE UPDATE ON customer
       FOR EACH ROW
       BEGIN
       IF (new.balance > new.credit_limit)
      THEN INSERT INTO debit_balance_report (action_dt, customer_number)
       VALUES (SYSDATE(), new.customer_number);
       END IF;
       END; //

mysql> DELIMITER ;
```

Ch. 19 Triggers

Using a trigger to stop an invalid transaction:

- Your business rule on the above may be that we will allow over credit limit amounts to 110% of credit limit, but not higher. In order to stop the SQL statement that caused the trigger in the first place, the trigger must SIGNAL and exception. The SQL 2003 standard defines a procedure called SIGNAL to do just this. Oracle uses a command called RAISE APPLICATION_ERROR.
- Unfortunately the current version of MySQL does not yet support SIGNAL, version 5.2 is set to start this support. It is possible to write our own SIGNAL procedure, but it is sort of a **hack job.**

Ch. 19 Triggers

```
-- This version also doesn't work in MySQL 5.1
mysql> DELIMITER $$

mysql> DROP PROCEDURE IF EXISTS `test`.`my_signal` $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `my_signal`(in_errorText
    VARCHAR(255))
    BEGIN
    SET @sql=CONCAT('UPDATE `', in_errorText, '` SET x=1');
    PREPARE my_signal_stmt FROM @sql;
    EXECUTE my_signal_stmt;
    DEALLOCATE PREPARE my_signal_stmt;
    END $$

mysql> DELIMITER ;
```

Ch. 19 Triggers

-- This version will work in MySQL 5.1

```
mysql> DELIMITER //
mysql> CREATE DEFINER=`root`@`localhost` PROCEDURE `my_signal`(in_errorText VARCHAR(255))
        BEGIN
        update junk set nothing = in_errortext;
        END//
mysql> DELIMITER //
mysql> CREATE TRIGGER debit_balance
        BEFORE UPDATE ON customer
        FOR EACH ROW
        BEGIN
          IF (new.balance > new.credit_limit) THEN INSERT INTO debit_balance_report (action_dt,
          customer_number) VALUES (SYSDATE(), new.customer_number);
    -- If the Amount over the credit limit is greater than 10%, Then Raise an Exception
          IF ((new.balance - new.credit_limit)/new.credit_limit > .1) THEN
          CALL my_signal ('Over Credit Limit Amount is TOO High');
            END IF;
          END IF;
        END;//

mysql> DELIMITER ;
```

Ch. 19 Triggers

- ***All your MySQL functions, procedures and triggers must have an exception handler or you will receive one full letter grade deduction and your work will be returned to be fixed.***
- ***Note the indentation above, failure to indent for readability will result in one full letter grade deduction and your work being returned to you to fix. Also I will not help you with any program that is NOT readable.***
- ***All your MySQL functions, procedures and triggers must have comments similar to the above or you will receive one full letter grade deduction and your work will be returned to be fixed.***

- The RAISE_APPLICATION_ERROR prcoedure takes two input parameters: The error number (which must be between ?20001 and ?20999) and the error message to be displayed.

SQL> UPDATE customer SET balance = 1101 WHERE customer_number = 124;

SQL> select trigger_name, table_name from user_triggers; -- Information on triggers;

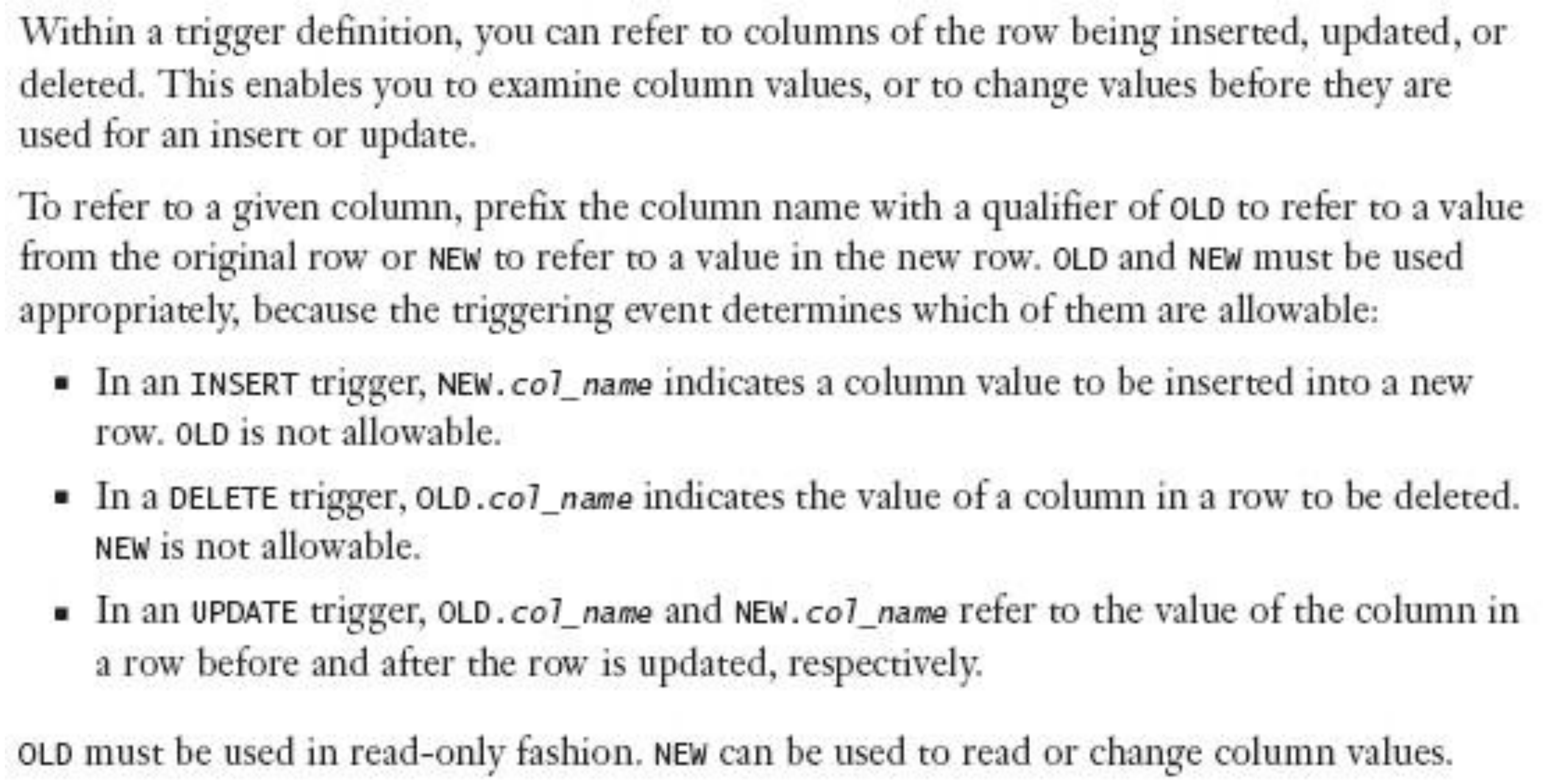
Ch. 19 Triggers

An example for OLD and New Column Values (p.310)

```
mysql> delimiter //

mysql> CREATE TRIGGER Capital_bu
            BEFORE UPDATE
            ON Capital
            FOR EACH ROW
            BEGIN
              SET @country = OLD.Country;
              SET @captial_old = OLD.Capital;
              SET @capital_new = NEW.Capital;
            End;//

mysql> delimiter ;
```

Ch. 19 Triggers

Referring to OLD and NEW column Values



Within a trigger definition, you can refer to columns of the row being inserted, updated, or deleted. This enables you to examine column values, or to change values before they are used for an insert or update.

To refer to a given column, prefix the column name with a qualifier of `OLD` to refer to a value from the original row or `NEW` to refer to a value in the new row. `OLD` and `NEW` must be used appropriately, because the triggering event determines which of them are allowable:

- In an `INSERT` trigger, `NEW.`*`col_name`* indicates a column value to be inserted into a new row. `OLD` is not allowable.
- In a `DELETE` trigger, `OLD.`*`col_name`* indicates the value of a column in a row to be deleted. `NEW` is not allowable.
- In an `UPDATE` trigger, `OLD.`*`col_name`* and `NEW.`*`col_name`* refer to the value of the column in a row before and after the row is updated, respectively.

`OLD` must be used in read-only fashion. `NEW` can be used to read or change column values.

Ch. 19 Triggers

Restrictions on Triggers

The current trigger implementation in MySQL has become limitations:

- You can NOT use the CALL statement
- You can NOT begin or end transactions
- You can NOT create a trigger for a TEMPORARY table or a view.
- Trigger creation is subject to the same restrictions placed on stored routine creation.

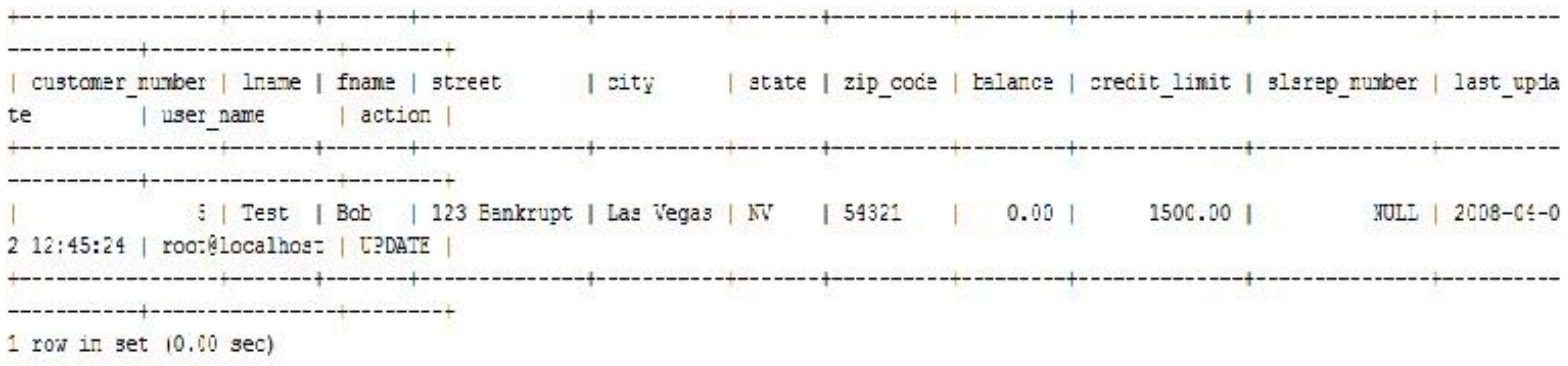
Ch. 19 Triggers

Using a TRIGGER to log changes

```
mysql> DELIMITER $$

mysql> CREATE TRIGGER tg_update_customer
        AFTER UPDATE ON customer
        FOR EACH ROW
        BEGIN
         INSERT INTO customer_history (customer_number, lname, fname, street, city, state,
    zip_code, credit_limit, balance, user_name, action, stamp) VALUES
    (OLD.customer_number, OLD.lname, OLD.fname, OLD.street, OLD.city, OLD.state,
    OLD.zip_code, OLD.credit_limit, OLD.balance, CURRENT_USER, 'UPDATE',
    SYSDATE());
        END; $$

mysql> DELIMITER ;
```

Ch. 19 Triggers

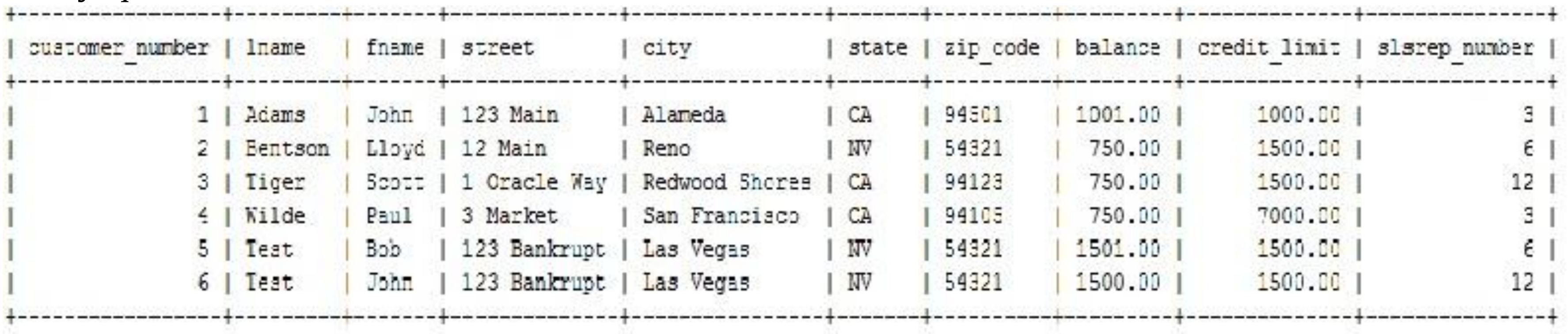
```
mysql> select * from customer_history;
Empty set (0.00 sec)
mysql> update customer set balance = 1750 where customer_number=5;
ERROR 1336 (0A000): Dynamic SQL is not allowed in stored function or trigger
mysql> select * from customer_history;
Empty set (0.00 sec)
mysql> update customer set balance = 1501 where customer_number=5;
Query OK, 1 row affected (0.05 sec) Rows matched: 1 Changed: 1 Warnings: 0
mysql> select * from customer_history;
```



```
+-----------------+-------+-------+--------------+------------+-------+----------+---------+--------------+--------------+----------
-----------+------------------+--------+
| customer_number | lname | fname | street       | city       | state | zip_code | balance | credit_limit | slsrep_number | last_upda
te          | user_name        | action |
+-----------------+-------+-------+--------------+------------+-------+----------+---------+--------------+--------------+----------
-----------+------------------+--------+
|               5 | Test  | Bob   | 123 Bankrupt | Las Vegas | NV    | 54321    |    0.00 |      1500.00 |          NULL | 2008-04-0
2 12:45:24 | root@localhost | UPDATE |
+-----------------+-------+-------+--------------+------------+-------+----------+---------+--------------+--------------+----------
-----------+------------------+--------+
1 row in set (0.00 sec)
```

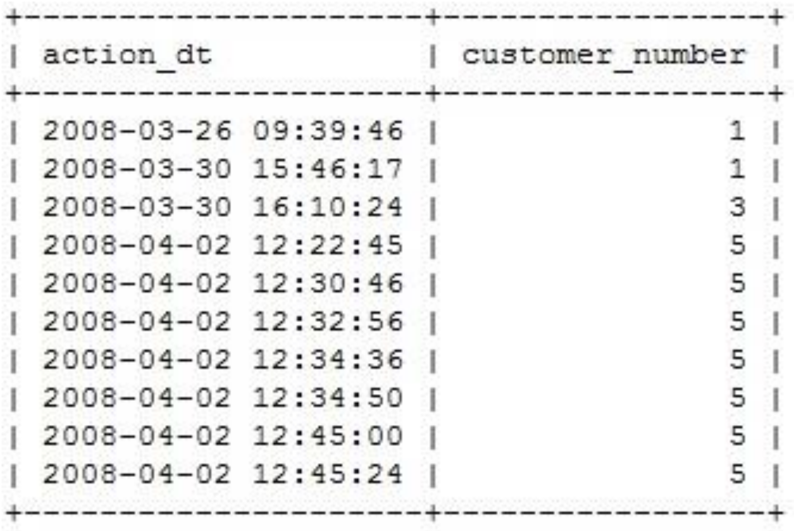
Ch. 19 Triggers

mysql> select * from customer;



```
+-----------------+--------+-------+--------------+---------------+-------+----------+---------+--------------+--------------+
| customer_number | lname  | fname | street       | city          | state | zip_code | balance | credit_limit | slsrep_number |
+-----------------+--------+-------+--------------+---------------+-------+----------+---------+--------------+--------------+
|               1 | Adams  | John  | 123 Main     | Alameda       | CA    | 94501    | 1001.00 |      1000.00 |            3 |
|               2 | Bentson| Lloyd | 12 Main      | Reno          | NV    | 54321    |  750.00 |      1500.00 |            6 |
|               3 | Tiger  | Scott | 1 Oracle Way | Redwood Shores| CA    | 94123    |  750.00 |      1500.00 |           12 |
|               4 | Wilde  | Paul  | 3 Market     | San Francisco | CA    | 94105    |  750.00 |      7000.00 |            3 |
|               5 | Test   | Bob   | 123 Bankrupt | Las Vegas     | NV    | 54321    | 1501.00 |      1500.00 |            6 |
|               6 | Test   | John  | 123 Bankrupt | Las Vegas     | NV    | 54321    | 1500.00 |      1500.00 |           12 |
+-----------------+--------+-------+--------------+---------------+-------+----------+---------+--------------+--------------+
```

mysql> select * from debit_balance_report;



```
+---------------------+-----------------+
| action_dt           | customer_number |
+---------------------+-----------------+
| 2008-03-26 09:39:46 |               1 |
| 2008-03-30 15:46:17 |               1 |
| 2008-03-30 16:10:24 |               3 |
| 2008-04-02 12:22:45 |               5 |
| 2008-04-02 12:30:46 |               5 |
| 2008-04-02 12:32:56 |               5 |
| 2008-04-02 12:34:36 |               5 |
| 2008-04-02 12:34:50 |               5 |
| 2008-04-02 12:45:00 |               5 |
| 2008-04-02 12:45:24 |               5 |
+---------------------+-----------------+
```

Ch. 19 Triggers

- **What about modifying the current table? Say I want to store the username in the current table as well.**

```
mysql> ALTER TABLE customer ADD (user_name VARCHAR(50));
mysql> CREATE TRIGGER tg_update_customer
        AFTER UPDATE ON customer
        FOR EACH ROW
        BEGIN
          INSERT INTO customer_history (customer_number, lname, fname, street,
    city, state, zip_code, credit_limit, balance, user_name, action) VALUES
    (OLD.customer_number, OLD.lname, OLD.fname, OLD.street, OLD.city,
    OLD.state, OLD.zip_code, OLD.credit_limit, OLD.balance, CURRENT_USER,
    'UPDATE');
        SET NEW.user_name = CURRENT_USER;
        END; $$
ERROR 1362 (HY000): Updating of NEW row is not allowed in after trigger
```

Ch. 19 Triggers

- You can only change the NEW image of the record in a BEFORE trigger NOT and AFTER trigger. The reason being AFTER executes after the SQL statement completes and then it is too late to change the NEW image.

```
mysql> CREATE TRIGGER tg_update_customer
        BEFORE UPDATE ON customer
        FOR EACH ROW -> BEGIN
        INSERT INTO customer_history (customer_number, lname, fname, street, city,
    state, zip_code, credit_limit, balance, user_name, action) VALUES
    (OLD.customer_number, OLD.lname, OLD.fname, OLD.street, OLD.city, OLD.state,
    OLD.zip_code, OLD.credit_limit, OLD.balance, CURRENT_USER, 'UPDATE');
        SET NEW.user_name = CURRENT_USER;
        END; $$

ERROR 1235 (42000): This version of MySQL doesn't yet support 'multiple triggers with the same action time and event for one table'

mysql> DELIMITER ;
```

Ch. 19 Triggers

- ***OOP's, we already have a BEFORE UPDATE TRIGGER ON Customer table. We will need to merge the two.***

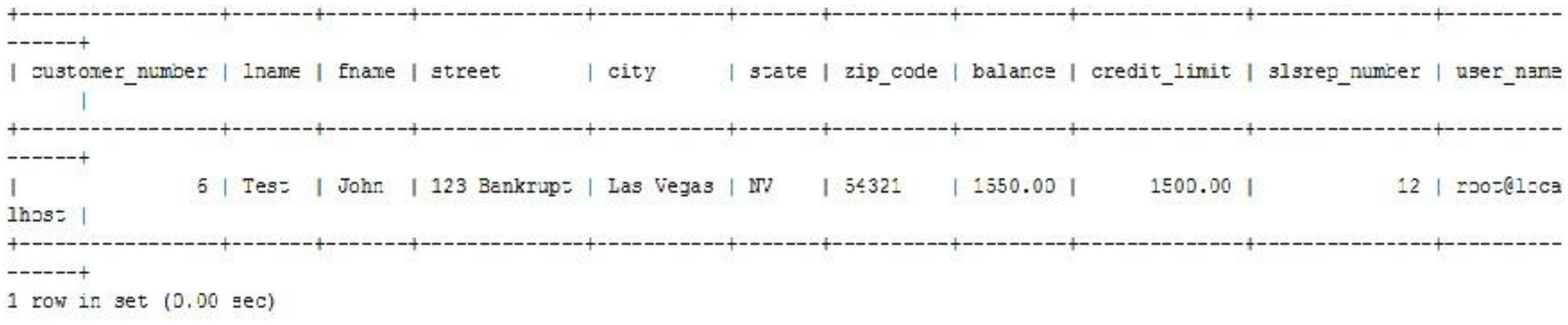
```
mysql> DROP TRIGGER tg_update_customer;
mysql> DROP TRIGGER debit_balance;
mysql> DELIMITER $$
mysql> CREATE TRIGGER tg_update_customer
            BEFORE UPDATE ON customer
            FOR EACH ROW
            BEGIN
            IF (new.balance > new.credit_limit) THEN INSERT INTO debit_balance_report (action_dt, customer_number)
     VALUES (SYSDATE(), new.customer_number);
 -- If the Amount over the credit limit is greater than 10%, Then Raise an Exception
            IF ((new.balance - new.credit_limit)/new.credit_limit > .1) THEN
            CALL my_signal ('Over Credit Limit Amount is TOO High');
                        END IF;
                END IF;
     INSERT INTO customer_history (customer_number, lname, fname, street, city, state, zip_code, credit_limit, balance,
     user_name, action, stamp) VALUES (OLD.customer_number, OLD.lname, OLD.fname, OLD.street, OLD.city, OLD.state,
     OLD.zip_code, OLD.credit_limit, OLD.balance, CURRENT_USER, 'UPDATE', SYSDATE());
            SET NEW.user_name = CURRENT_USER;
            SET NEW.updated = SYSDATE();
            END; $$

mysql> DELIMITER ;
```

Ch. 19 Triggers

mysql> UPDATE customer SET balance = 1550 WHERE customer_number=6;

Query OK, 1 row affected (0.03 sec) Rows matched: 1 Changed: 1 Warnings: 0

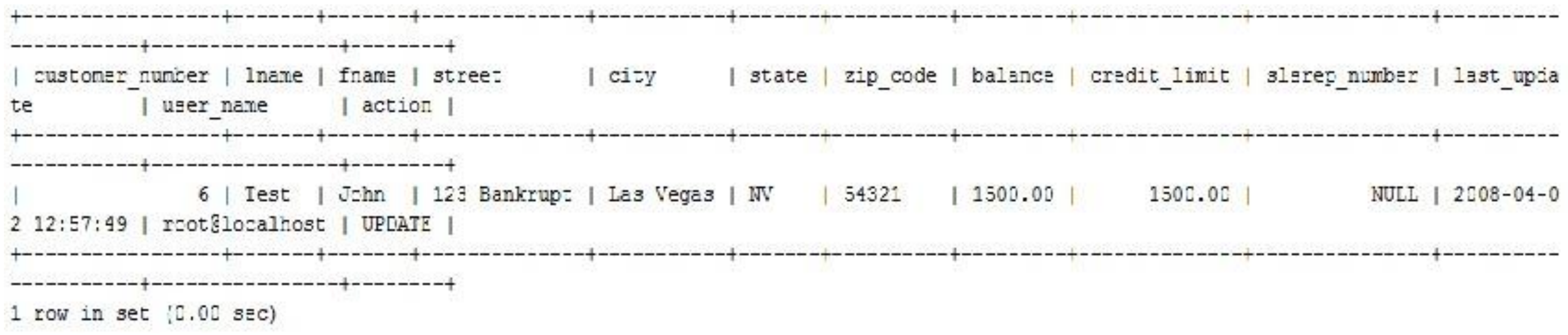
mysql> SELECT * FROM customer WHERE customer_number = 6;



```
+-----------------+-------+-------+--------------+-----------+-------+----------+---------+--------------+--------------+----------
------+
| customer_number | lname | fname | street       | city      | state | zip_code | balance | credit_limit | slsrep_number | user_name
      |
+-----------------+-------+-------+--------------+-----------+-------+----------+---------+--------------+--------------+----------
------+
|               6 | Test  | John  | 123 Bankrupt | Las Vegas | NV    | 54321    | 1550.00 |      1500.00 |           12 | root@loca
lhost |
+-----------------+-------+-------+--------------+-----------+-------+----------+---------+--------------+--------------+----------
------+
1 row in set (0.00 sec)
```

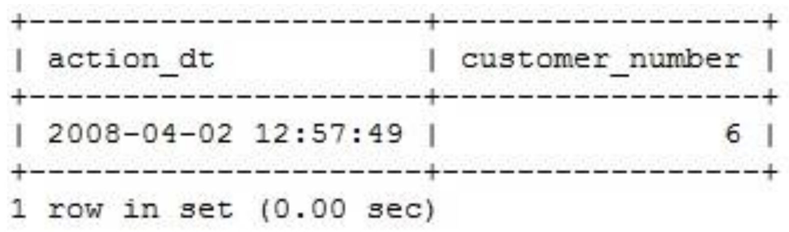
Ch. 19 Triggers

mysql> SELECT * FROM customer_history WHERE customer_number = 6;



```
+-----------------+-------+-------+--------------+-----------+-------+----------+---------+--------------+--------------+----------
-----------+----------------+--------+
| customer_number | lname | fname | street       | city      | state | zip_code | balance | credit_limit | slsrep_number | last_upda
te          | user_name      | action |
+-----------------+-------+-------+--------------+-----------+-------+----------+---------+--------------+--------------+----------
-----------+----------------+--------+
|               6 | Test  | John  | 123 Bankrupt | Las Vegas | NV    | 54321    | 1500.00 |      1500.00 |          NULL | 2008-04-0
2 12:57:49 | root@localhost | UPDATE |
+-----------------+-------+-------+--------------+-----------+-------+----------+---------+--------------+--------------+----------
-----------+----------------+--------+
1 row in set (0.00 sec)
```

mysql> SELECT * FROM debit_balance_report WHERE customer_number = 6;



```
+---------------------+-----------------+
| action_dt           | customer_number |
+---------------------+-----------------+
| 2008-04-02 12:57:49 |               6 |
+---------------------+-----------------+
1 row in set (0.00 sec)
```

Ch. 19 Triggers

Destroying a Trigger

- To destroy a trigger, use the DROP TRIGGER statement.

DROP TRIGGER W$\underbrace{\text{ORLD.}}_{\text{Database}}\underbrace{\text{Capital_bi;}}_{\text{Trigger}}$

If your default database is “World”, you can also write syntax like:

DROP TRIGGER Capital.Bi;

- Drop trigger destroys a trigger explicitly. Triggers also are destroyed implicitly under some circumstances. When you drop a table that has triggers associated with it, MySQL drops the triggers as well. When you drop a database, doing so causes tables in the databases to be dropped, and thus also drops any triggers for those tables.