**Project No.2**

Yongchi Zhang

As of MySQL 5.7.16, it is possible to define multiple triggers for a given table that have the same trigger event and action time. For example, you can have two BEFORE UPDATE triggers for a table. By default, triggers that have the same trigger event and action time activate in the order they were created. To affect trigger order, specify a clause after FOR EACH ROW that indicates FOLLOWS or PRECEDES and the name of an existing trigger that also has the same trigger event and action time. With FOLLOWS, the new trigger activates after the existing trigger. With PRECEDES, the new trigger activates before the existing trigger.

Thus I don’t need to combine trigger of Q1, Q4 and Q6 as I do in MySQL version 5.6.

Question 1

Establish the trigger:

|  |
| --- |
| delimiter $$  drop trigger if exists movie\_category\_insert $$  create trigger movie\_category\_insert  before insert on made\_money  for each row  begin  if new.Category!='Romantic' and new.Category!='Comedy' and new.Category!='Drama' and new.Category!='Action' then  set new.Category='Action';  end if;  end $$  drop trigger if exists movie\_category\_update $$  create trigger movie\_category\_update  before update on made\_money  for each row  begin  if new.Category!='Romantic' and new.Category!='Comedy' and new.Category!='Drama' and new.Category!='Action' then  set new.Category='Action';  end if;  end $$  delimiter ; |

Question 2

Establish the trigger:

|  |
| --- |
| delimiter $$  **drop** **trigger** **if** **exists** star\_movie $$  **create** **trigger** star\_movie  **before** **insert** **on** appeared\_in  **for** **each** **row**  **begin**  **if** **not** **exists**  **(**  **select** **\***  **from** appeared\_in t1**,** made\_money t2  **where** t1**.**star**=new.**star **and**  t1**.**movie**=**t2**.**movie **and**  **(select** DAY\_OPENED **from** made\_money **where** movie**=new.**movie**)** **>** t2**.**DAY\_OPENED **and**  t2**.**category **in** **(**'Romantic'**,**'Drama'**,**'Comedy'**)**  **)**  **then**  **update** made\_money **as** t3  **set** t3**.**category**=**'Drama'  **where** t3**.**movie**=new.**movie**;**  **end** **if;**  **end** $$  delimiter **;** |

Question 3

Establish the trigger:

|  |
| --- |
| delimiter $$  **drop** **trigger** **if** **exists** star\_married $$  **create** **trigger** star\_married  **before** **insert** **on** married  **for** **each** **row**  **begin**  **declare** dday **date;**  **declare** done **int** **default** -1**;**  **declare** mycursor **cursor** **for**  **(select** t3**.day** **as** divorce\_day  **from** **(**in\_couple t1**,** married t2**)** **left** **outer** **join** divorced t3  **on** t2**.**COUPLE\_NUM**=**t3**.**COUPLE\_NUM  **where** t1**.**star **in** **(select** star **from** in\_couple **where** COUPLE\_NUM**=new.**couple\_num**)** **and**  t1**.**COUPLE\_NUM**=**t2**.**COUPLE\_NUM **);**  **declare** **continue** handler **for** **not** **found** **set** done**=**1**;**  **open** mycursor**;**  checkloop**:loop**  **fetch** mycursor **into** dday**;**  **if** **(**done**=**1**)** **then**  leave checkloop**;**  **end** **if;**  **if** **(**dday **is** **null** **or** **new.day<**dday**)** **then**  SIGNAL **SQLSTATE** '45000'  **set** MESSAGE\_TEXT **=**'Cannot insert data because of the constraint: A star cannot be married to multiple stars simultaneously.'**;**  leave checkloop**;**  **end** **if;**  **end** **loop** checkloop**;**  **end** $$  delimiter **;** |

Question 4

Establish the trigger:

|  |
| --- |
| delimiter $$  **drop** **trigger** **if** **exists** boxoffice\_constraint $$  **create** **trigger** boxoffice\_constraint  **before** **insert** **on** made\_money  **for** **each** **row follows** movie\_category\_insert  **begin**  **if(new.**category**=**'Action' **and** **new.**how\_much**<**10000**)** **then**  SIGNAL **SQLSTATE** '45000'  **set** MESSAGE\_TEXT **=**'Cannot insert data because of the box office constraint.'**;**  elseif**(new.**category**=**'Comedy' **and** **new.**how\_much**>**1000000000**)** **then**  SIGNAL **SQLSTATE** '45000'  **set** MESSAGE\_TEXT **=**'Cannot insert data because of the box office constraint.'**;**  elseif**(new.**how\_much**<**1000 **or** **new.**how\_much**>**3000000000**)** **then**  SIGNAL **SQLSTATE** '45000'  **set** MESSAGE\_TEXT **=**'Cannot insert data because of the box office constraint.'**;**  **end** **if;**  **end** $$  **drop** **trigger** **if** **exists** boxoffice\_constraint\_update $$  **create** **trigger** boxoffice\_constraint\_update  **before** **update** **on** made\_money  **for** **each** **row follows** movie\_category\_update  **begin**  **if(new.**category**=**'Action' **and** **new.**how\_much**<**10000**)** **then**  SIGNAL **SQLSTATE** '45000'  **set** MESSAGE\_TEXT **=**'Cannot insert data because of the box office constraint.'**;**  elseif**(new.**category**=**'Comedy' **and** **new.**how\_much**>**1000000000**)** **then**  SIGNAL **SQLSTATE** '45000'  **set** MESSAGE\_TEXT **=**'Cannot insert data because of the box office constraint.'**;**  elseif**(new.**how\_much**<**1000 **or** **new.**how\_much**>**3000000000**)** **then**  SIGNAL **SQLSTATE** '45000'  **set** MESSAGE\_TEXT **=**'Cannot insert data because of the box office constraint.'**;**  **end** **if;**  **end** $$  delimiter **;** |

Question 5

Establish the trigger:

|  |
| --- |
| delimiter $$  **drop** **trigger** **if** **exists** divorce\_constraint\_insert $$  **create** **trigger** divorce\_constraint\_insert  **before** **insert** **on** divorced  **for** **each** **row**  **begin**  **if** **(new.day** **<** **(** **select** t1**.day**  **from** married t1  **where** t1**.**couple\_num **=** **new.**couple\_num **)** **)**  **then**  **set** **new.day** **=** **(** **select** t2**.day**  **from** married t2  **where** t2**.**couple\_num **=** **new.**couple\_num **);**  **end** **if;**  **end** $$  **drop** **trigger** **if** **exists** divorce\_constraint\_update $$  **create** **trigger** divorce\_constraint\_update  **before** **update** **on** divorced  **for** **each** **row**  **begin**  **if** **(new.day** **<** **(** **select** t1**.day**  **from** married t1  **where** t1**.**couple\_num **=** **new.**couple\_num **)** **)**  **then**  **set** **new.day** **=** **(** **select** t2**.day**  **from** married t2  **where** t2**.**couple\_num **=** **new.**couple\_num **);**  **end** **if;**  **end** $$  delimiter **;** |

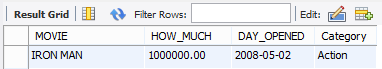
Question 6

|  |
| --- |
| delimiter $$  **drop** **trigger** **if** **exists** write\_to\_log\_data $$  **create** **trigger** write\_to\_log\_data  **before** **insert** **on** made\_money  **for** **each** **row follows** boxoffice\_constraint  **begin**  **insert** **into** log\_data **values(new.**movie**,new.**category**);**  **end** $$ |

**Results**

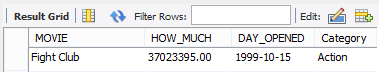
a) Insert a new movie, with values (“IRON MAN”, 1000000, 2008-05-02, “ACTON”) in MADE\_MONEY table.

|  |
| --- |
| **insert** **into** made\_money **values(**'IRON MAN'**,** 1000000**,** 20080502**,** 'ACTON'**);** |



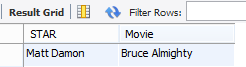
b) Update the CATEGORY of the movie “Fight Club” to “Horror” in MADE\_MONEY table.

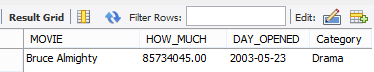
|  |
| --- |
| **update** made\_money **set** Category**=**'Horror' **where** movie**=**'Fight Club'**;** |



c) Insert a new tuple in APPEARED\_IN table, with values (“Matt Damon”, “Bruce Almighty”).

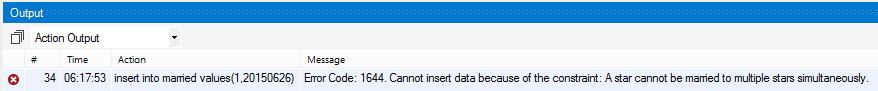
|  |
| --- |
| **insert** **into** appeared\_in **values(**'Matt Damon'**,** 'Bruce Almighty'**);** |





d) Insert a new tuple in MARRIED, with values (1, 2015-06-26).

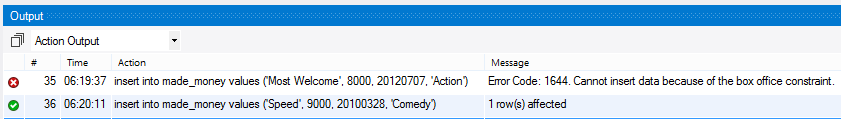
|  |
| --- |
| **insert** **into** married **values(**1**,**20150626**);** |



It returns the error code 1644.

e) Insert two new tuples in MADE\_MONEY, having values (“Most Welcome”, 8000, 2012-07-07, “Action”) and (“Speed”, 9000, 2010-03-28, “Comedy”).

|  |
| --- |
| **insert** **into** made\_money **values** **(**'Most Welcome'**,** 8000**,** 20120707**,** 'Action'**);**  **insert** **into** made\_money **values** **(**'Speed'**,** 9000**,** 20100328**,** 'Comedy'**);** |

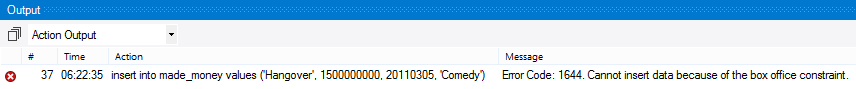


For the first insertion, it returns the error code 1644.

For the second insertion, it is successful.

f) Insert a new tuple in MADE\_MONEY, having values (“Hangover”, 1500000000, 2011-03-05, “Comedy”).

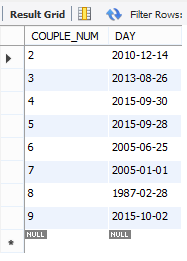
|  |
| --- |
| **insert** **into** made\_money **values** **(**'Hangover'**,** 1500000000**,** 20110305**,** 'Comedy'**);** |



It returns the error code 1644.

g) Insert a new tuple in DIVORCED, with values (6, 2004-01-01).

|  |
| --- |
| **insert** **into** divorced **values** **(**6**,** 20040101**);** |



After performing the above statements, the LOG\_DATA Table is:

