26 Oct 2020

Government Engineering College, Thrissur

CS333– Application Software Development Lab

Documentation -

Exp 14 – Creation of Packages
Exp 15 – Creation of database Triggers and Cursors

Date of Submission

26 October 2020

Submitted By

Kowsik Nandagopan D

Roll No 31

TCR18CS031

GECT CSE S5

26 Oct 2020

Experiment 14

AIM

Creation of Packages

Description

Mysql supports stored procedures - the namespace of this is at the database level - there is no explicit support for creating namespaces at a lower level, and the '. ' scope operator seeprates databases and objects. So there is no equivalent entity in **MySQL** to an Oracle **package**

26 Oct 2020

Experiment 15

AIM

Creation of database Triggers and Cursors

Description

Implement Creation of database Triggers and Cursors in MySQL to get valuable insight from the previously created database.

A **trigger** is a named database object that is associated with a table, and that activates when a particular event occurs for the table. Some uses for **triggers** are to perform checks of values to be inserted into a table or to perform calculations on values involved in an update. We can use triggers along with INSERT, UPDATE, and DELETE. We **cannot apply triggers on MySQL views**

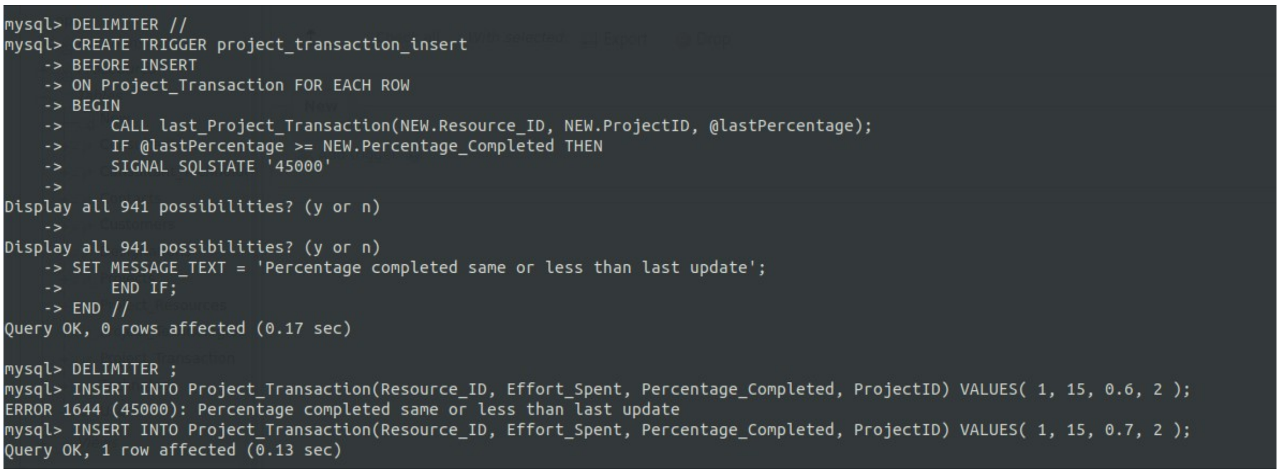
A **cursor** allows you to iterate a set of rows returned by a query and process each row individually. **MySQL cursor** is read-only, non-scrollable and asensitive. You cannot fetch rows in the reversed order. In addition, you cannot skip rows or jump to a specific row in the result set.We have to create cursor(declare) → Set not found handler → Open Cursor → Loop using Fetch → Close Cursor.

Output / Screenshots

1. Trigger

1. Before Insert

- Trigger to check the currently entered percentage completed on project transaction is less than or equal to the previously entered value
- If the trigger is true it will signal the error to the screen and prevents the data to be entered to the database

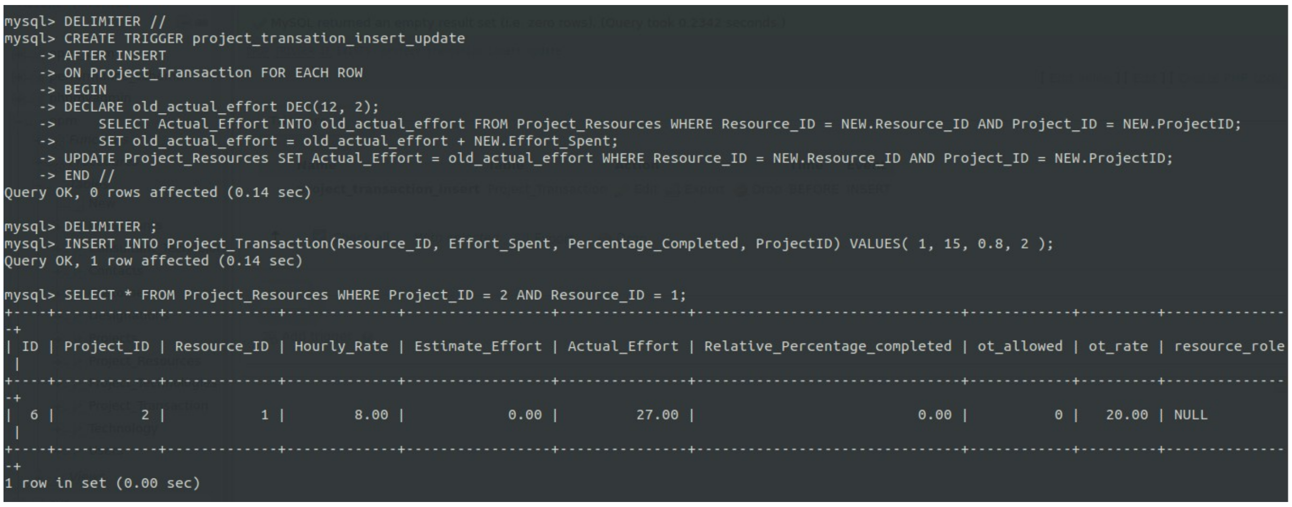


```
mysql> DELIMITER //
mysql> CREATE TRIGGER project_transaction_insert
    -> BEFORE INSERT
    -> ON Project_Transaction FOR EACH ROW
    -> BEGIN
    ->     CALL last_Project_Transaction(NEW.Resource_ID, NEW.ProjectID, @lastPercentage);
    ->     IF @lastPercentage >= NEW.Percentage_Completed THEN
    ->     SIGNAL SQLSTATE '45000'
    ->
Display all 941 possibilities? (y or n)
    ->
Display all 941 possibilities? (y or n)
    -> SET MESSAGE_TEXT = 'Percentage completed same or less than last update';
    ->     END IF;
    -> END //
Query OK, 0 rows affected (0.17 sec)

mysql> DELIMITER ;
mysql> INSERT INTO Project_Transaction(Resource_ID, Effort_Spent, Percentage_Completed, ProjectID) VALUES( 1, 15, 0.6, 2 );
ERROR 1644 (45000): Percentage completed same or less than last update
mysql> INSERT INTO Project_Transaction(Resource_ID, Effort_Spent, Percentage_Completed, ProjectID) VALUES( 1, 15, 0.7, 2 );
Query OK, 1 row affected (0.13 sec)
```

2. After Insert

- Create a trigger to update the project resource table to update actual efforts when we insert the new project transaction



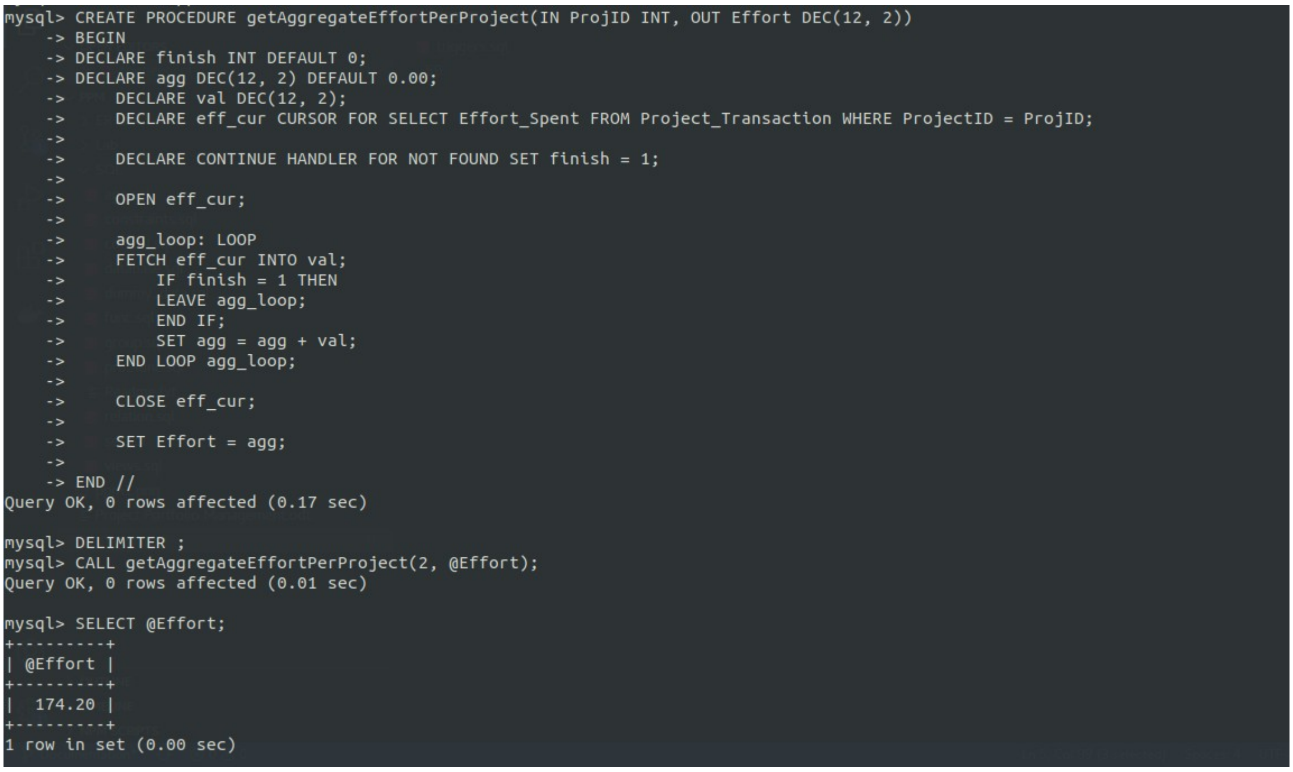
```
mysql> DELIMITER //
mysql> CREATE TRIGGER project_transation_insert_update
    -> AFTER INSERT
    -> ON Project_Transaction FOR EACH ROW
    -> BEGIN
    -> DECLARE old_actual_effort DEC(12, 2);
    ->     SELECT Actual_Effort INTO old_actual_effort FROM Project_Resources WHERE Resource_ID = NEW.Resource_ID AND Project_ID = NEW.ProjectID;
    ->     SET old_actual_effort = old_actual_effort + NEW.Effort_Spent;
    -> UPDATE Project_Resources SET Actual_Effort = old_actual_effort WHERE Resource_ID = NEW.Resource_ID AND Project_ID = NEW.ProjectID;
    -> END //
Query OK, 0 rows affected (0.14 sec)

mysql> DELIMITER ;
mysql> INSERT INTO Project_Transaction(Resource_ID, Effort_Spent, Percentage_Completed, ProjectID) VALUES( 1, 15, 0.8, 2 );
Query OK, 1 row affected (0.14 sec)

mysql> SELECT * FROM Project_Resources WHERE Project_ID = 2 AND Resource_ID = 1;
+----+------------+-------------+-------------+-----------------+---------------+-------------------------------+------------+---------+--------------
-+
| ID | Project_ID | Resource_ID | Hourly_Rate | Estimate_Effort | Actual_Effort | Relative_Percentage_completed | ot_allowed | ot_rate | resource_role
 |
+----+------------+-------------+-------------+-----------------+---------------+-------------------------------+------------+---------+--------------
-+
|  6 |          2 |           1 |        8.00 |            0.00 |         27.00 |                          0.00 |          0 |   20.00 | NULL
 |
+----+------------+-------------+-------------+-----------------+---------------+-------------------------------+------------+---------+--------------
-+
1 row in set (0.00 sec)
```

2. Cursor

Compute the whole effort spent per project from project transaction



```
mysql> CREATE PROCEDURE getAggregateEffortPerProject(IN ProjID INT, OUT Effort DEC(12, 2))
    -> BEGIN
    -> DECLARE finish INT DEFAULT 0;
    -> DECLARE agg DEC(12, 2) DEFAULT 0.00;
    ->     DECLARE val DEC(12, 2);
    ->     DECLARE eff_cur CURSOR FOR SELECT Effort_Spent FROM Project_Transaction WHERE ProjectID = ProjID;
    ->
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET finish = 1;
    ->
    ->     OPEN eff_cur;
    ->
    ->     agg_loop: LOOP
    ->     FETCH eff_cur INTO val;
    ->         IF finish = 1 THEN
    ->         LEAVE agg_loop;
    ->         END IF;
    ->         SET agg = agg + val;
    ->     END LOOP agg_loop;
    ->
    ->     CLOSE eff_cur;
    ->
    ->     SET Effort = agg;
    ->
    -> END //
Query OK, 0 rows affected (0.17 sec)

mysql> DELIMITER ;
mysql> CALL getAggregateEffortPerProject(2, @Effort);
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT @Effort;
+---------+
| @Effort |
+---------+
|  174.20 |
+---------+
1 row in set (0.00 sec)
```