DBMSL 31147

Assignment 7

Date of Completion : 4/9/20 Date of Submission : 4/10/20

Title : PL/SQL Stored Procedure and Stored Function.

Problem Statement :

Write a stored Procedure namely proc_Grade for the categorization of customer. If purchases by customer in year is <=20000 and >= 10000 then customer will be placed in platinum category. If purchase by customer is between 9999 and 5000 category is gold if purchase between 4999 and 2000 category is Silver

Write a PL/SQL block for using procedure created with above requirement. Customer (cust_id, name, total_purchase)

Category (cust_id, name, class).

Learning Objectives :

1) Understand the concept of procedure in MYSQL.
2) Understand IN, OUT, INOUT parameters for Procedures.
3) Understand the functions implementation

Learning Outcomes :

1) Implement and learn about procedure with and without parameters.
2) Learn the use of function in MYSQL.

S/W & H/W requirements :

- Windows / Ubuntu.
- MYSQL

Theory :

What is stored procedure?

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again. So if you have query that you write over and over again, save it as stored procedure, and then call it to execute it

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value that is passed.

Page No.
Date

Advantages of procedure:
i) Reduces network traffic.
ii) Centralize business logic in the database
iii) Make database more secure

Disadvantages of procedure:-
i) Resources usages.
ii) Troubleshooting
iii) Maintenances.

We can pass parameters to procedures in three ways:
1) IN parameters
2) OUT parameters
3) IN OUT parameters.

A procedure may or may not return a value syntax to create a stored procedure:

```
DROP PROCEDURE IF EXISTS proc_name;
DELIMITER //
CREATE PROCEDURE proc_name(
        param1 type1,
        param2 type2)
BEGIN
    :
    SQL statements and Queries
    :
END //
DELIMITER ;
```

Procedure can be invoked by.

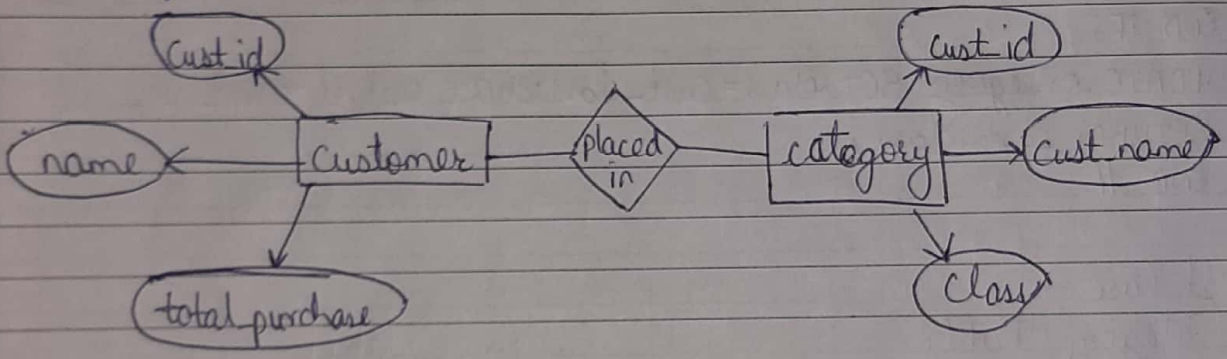
```
CALL proc_name(param1, param2);
```

Functions:
A function is a named PL/SQL block which is similar to a procedure. The major difference is that a function always return a value, but a stored procedure may or may not return value.

Syntax:

```
CREATE FUNCTION func_name(parameter).
Returns .data_type [characteristic]
BEGIN
    :
    MY SQL statements
    :
    RETURN expression;
END.
```

ER Diagram:



Source Code

Parameterized procedure

```
DELIMITER //
CREATE PROCEDURE proc_grade (IN cust_no INT, OUT class_assigned VARCHAR(100))
 BEGIN
DECLARE total_price INT;
SELECT total_purchase into total_price FROM customer WHERE cust_id = cust_no;
IF total_price BETWEEN 10000 AND 20000 THEN
SET class_assigned = 'Platinum';
ELSEIF total_price BETWEEN 5000 AND 9999 THEN
 SET class_assigned = "Gold";
ELSEIF total_price BETWEEN 2000 AND 4999 THEN
 SET class_assigned = "silver";
 END IF
UPDATE category SET class_assigned Where cust_id = cust_no;
END //
```

Page No.
Date

Function:

```
CREATE FUNCTION func_grade (cust_no INT) RETURNS VARCHAR(100)
BEGIN
DECLARE cust_class VARCHAR(100);
DECLARE total_price INT;
SELECT total_purchase INTO total_price FROM customer WHERE cust_id = cust_no;
IF total_price BETWEEN 10000 AND 20000 THEN
 SET cust_class = "Platinum";
ELSEIF total_price BETWEEN 5000 AND 9999 THEN
 SET cust_class = "Gold";
ELSEIF total_price BETWEEN 2000 AND 4999 THEN
 SET cust_class = "Silver";
END IF;
UPDATE category SET class = cust_class WHERE cust_id = cust_no;
RETURN cust_class;
END //
```

Test Case:

Category Table

Cust_id	Cust_name	class
1	Will	NULL
5	Harry	NULL

Customer Table

Cust_id	cust_name	total_purchase
1	Will	2000
5	Harry	15000

1) `CALL proc_grade(1, @cat);`
`SELECT @cat`

@cat
Silver

2) `CALL proc_grade(5, @cat);`
`SELECT @cat`

@cat
Platinum

After category Upgrade.

cust_id	cust_name	class
1	Will	Silver
5	Harry	Platinum

Conclusion:

We have understood and successfully implemented stored procedure and stored function in PL/SQL.

MySQL 8.0 Command Line Client

```
ysql> SELECT * FROM category;
+---------+-----------+---------+
| cust_id | cust_name | class   |
+---------+-----------+---------+
|       1 | Will      | NULL    |
|       2 | John      | NULL    |
|       3 | Smith     | NULL    |
|       4 | James     | NULL    |
|       5 | Harry     | NULL    |
|       6 | Donald    | NULL    |
|       7 | New       | NULL    |
+---------+-----------+---------+
 rows in set (0.00 sec)

ysql> SELECT * FROM customer;
+---------+---------+----------------+
| cust_id | name    | total_purchase |
+---------+---------+----------------+
|       1 | Will    |           2000 |
|       2 | John    |           5000 |
|       3 | Smith   |           4999 |
|       4 | James   |           9989 |
|       5 | Harry   |          15000 |
|       6 | Donald  |          10000 |
|       7 | New     |           5200 |
+---------+---------+----------------+
 rows in set (0.01 sec)

ysql> CALL proc_grade_parametized2(1,@cat);
uery OK, 1 row affected (0.01 sec)

ysql> SELECT @cat;
+--------+
| @cat   |
+--------+
| Silver |
+--------+
 row in set (0.00 sec)

ysql> SELECT * FROM category;
+---------+-----------+---------+
| cust_id | cust_name | class   |
+---------+-----------+---------+
|       1 | Will      | Silver  |
|       2 | John      | NULL    |
|       3 | Smith     | NULL    |
|       4 | James     | NULL    |
|       5 | Harry     | NULL    |
|       6 | Donald    | NULL    |
|       7 | New       | NULL    |
```