Name: K Srikanth Roll Number: 17ETCS002124

Laboratory 8

Title of the Laboratory Exercise: Stored Procedure in MySQL

1. Introduction and Purpose of Experiment

   A stored Procedure is a procedure stored in a database which can be called by the database engine and connected programming languages. A stored procedure is invoked using the CALL statement. A procedure has a name, a parameter list, and SQL statement(s). The parameters make the stored procedure more flexible and useful. In MySQL, a parameter has one of three modes: IN, OUT, or INOUT. By doing this lab, students will be able to implement MySQL stored procedure.

2. Aim and Objectives

   Aim

   - To implement MySQL stored procedure

   Objectives

   At the end of this lab, the student will be able to

   - Design SQL procedures for the given problem statement
   - Implement the procedures in MySQL

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Create the table and its attributes with essential properties
   iii. Design the procedures in MySQL
   iv. Implement the procedures in MySQL
   v. Test the implemented procedures
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

4. Questions

   a. Design and implement a procedure which accepts one INOUT parameter (count) and one IN parameter (inc). Inside the stored procedure, increase the counter (count) by the value of the inc parameter.
   b. Create a table OFFICES in OFFICEDB with attributes such as OfficeCode, OfficeName, OfficePhone, City, and Country.
      i. Write a ‘GetPhoneNo’ procedure in MySQL to take the name of an office and display the phone number of that office.

ii. Write a 'GetOffice' procedure in MySQL to take the name of a particular country and display the number of offices located in that Country.

5. Presentation of Results

Creating a Table

Query

```
-- Creating a Table "Offices"

create table offices(OfficeCode int AUTO_INCREMENT,
OfficeName varchar(20),
OfficePhone BIGINT,
City VARCHAR(20),
Country VARCHAR(20),
Primary Key (OfficeCode));

```

Figure 1 MySQL Queries to create "offices" table.

Result

```
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| OfficeCode  | int         | NO   | PRI | NULL    | auto_increment |
| OfficeName  | varchar(20) | YES  |     | NULL    |                |
| OfficePhone | bigint      | YES  |     | NULL    |                |
| City        | varchar(20) | YES  |     | NULL    |                |
| Country     | varchar(20) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Figure 2 MySQL Metadata for "offices" table.

Inserting Data

Query

```
-- Inserting Data into Table "Offices"

insert into offices(OfficeName,OfficePhone,City,Country) values ("Bangalore",123131312,"Bangalore","India");
insert into offices(OfficeName,OfficePhone,City,Country) values ("Apple",123131312312,"Bangalore","India");
insert into offices(OfficeName,OfficePhone,City,Country) values ("Tesla",123131312,"Bangalore","India");
insert into offices(OfficeName,OfficePhone,City,Country) values ("Samsung",123131312,"Bangalore","India");

select * from offices;

```

Figure 3 MySQL Queries to insert data into "offices" table.

Result

```
mysql> select * from offices;
+------------+------------+--------------+-----------+---------+
| OfficeCode | OfficeName | OfficePhone  | City      | Country |
+------------+------------+--------------+-----------+---------+
|          1 | Bangalore  |    123131312 | Bangalore | India   |
|          2 | Apple      | 123131312312 | Bangalore | India   |
|          3 | Tesla      |    123131312 | Bangalore | India   |
|          4 | Samsung    |    123131312 | Bangalore | India   |
+------------+------------+--------------+-----------+---------+
4 rows in set (0.00 sec)
```

Figure 4 MySQL Data inside "office" table.

Question A

Query

```
-- Question A

delimiter @@

CREATE PROCEDURE counting(INOUT a INT,IN inc INT)
begin

    set a = a + inc;

end @@

set @count=0;
call counting(@count,200);
call counting(@count,300);
select @count;

```

Figure 5 MySQL Queries for given problem statement using procedure.(Question 1)

Result

```
mysql> select @count;
+---------+
| @count |
+---------+
|    500 |
+---------+
1 row in set (0.00 sec)
```

Figure 6 MySQL result for the given problem statement using procedure.(Question 1)

Question B (i)

Query

```
--- Question B

delimiter @@
CREATE PROCEDURE getPhNumber(IN Name VARCHAR(20))
begin
    select OfficePhone from offices where OfficeName = Name;
end @@

call getPhNumber("Apple");

```

Figure 7 Figure 5 MySQL Queries for given problem statement using procedure.(Question 2)

Result

```
mysql> call getPhNumber("Apple");
+--------------+
| OfficePhone  |
+--------------+
| 123131312312 |
+--------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Figure 8 MySQL result for the given problem statement using procedure.(Question 2)

Question B (ii)

Query

```
48  -- Question C
49
50  delimiter @@
51  CREATE PROCEDURE getTheCityCount(IN OfficeCountry VARCHAR(20))
52  begin
53      select count(*) from offices where Country = OfficeCountry ;
54  end @@
55
56  call getTheCityCount("India");
57
```

Figure 9 Figure 5 MySQL Queries for given problem statement using procedure.(Question 3)

Result

```
mysql> call getTheCityCount("India");
+----------+
| count(*) |
+----------+
|        4 |
+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Figure 10 MySQL result for the given problem statement using procedure.(Question 3)

6. Conclusions

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 an SQL query that you write over and over again, save it as a stored procedure, and then just 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(s) that is passed.

These Procedures can be used with parameters also as we saw in Question B.

7. Comments

1. Limitations of Experiments

Sometimes using these procedures can be complicated as they not be that easy to debug.

2. Limitations of Results

Using procedure can help reduce writing queries again and again

3. Learning happened

Learned using procedures in MySQL and executing the queries.

Name: K Srikanth Roll Number: 17ETCS002124