Name : AKASH SIKARWAR Roll No. : 17ETCS002017

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

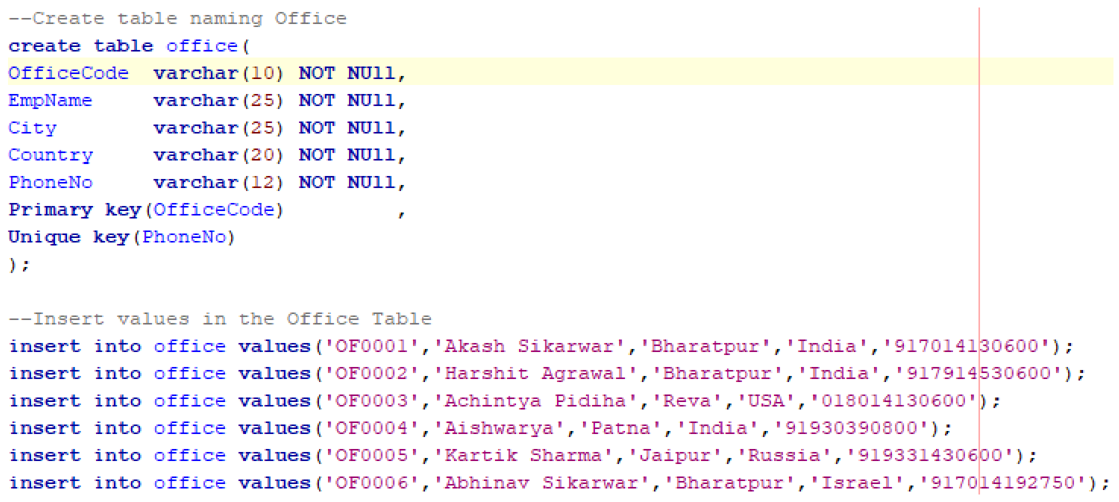
   a. Design and implement a procedure which accepts one INOUT parameter (count) and one IN parameter (inc). Inside the stored procedure, increase the counter (count) by the value of the inc parameter.

   b. Create a table OFFICES in OFFICEDB with attributes such as OfficeCode, Name, City, Country, and Phone.

      i. Write a 'GetPhoneNo' procedure in MySQL to take the name of a person as an input and display the phone number of that person.

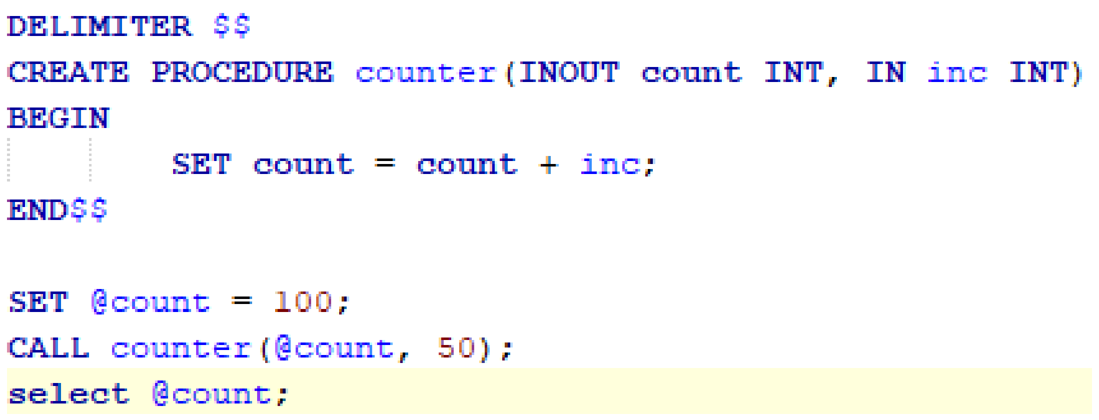
ii. Write a 'GetOffice' procedure in MySQL to take the name of a country as an input and display the number of offices located in a particular Country.

5. Presentation of Results



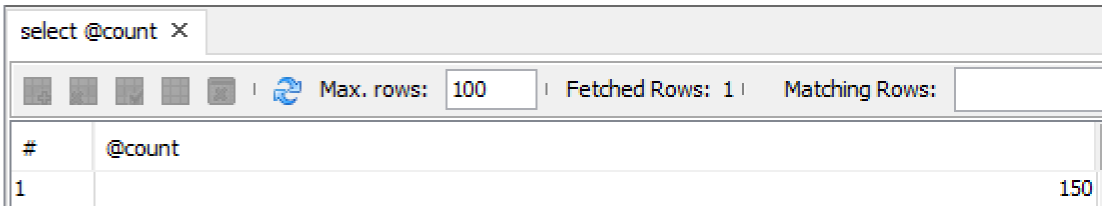
```
--Create table naming Office
create table office(
OfficeCode  varchar(10) NOT NUll,
EmpName     varchar(25) NOT NUll,
City        varchar(25) NOT NUll,
Country     varchar(20) NOT NUll,
PhoneNo     varchar(12) NOT NUll,
Primary key(OfficeCode)          ,
Unique key(PhoneNo)
);

--Insert values in the Office Table
insert into office values('OF0001','Akash Sikarwar','Bharatpur','India','917014130600');
insert into office values('OF0002','Harshit Agrawal','Bharatpur','India','917914530600');
insert into office values('OF0003','Achintya Pidiha','Reva','USA','018014130600');
insert into office values('OF0004','Aishwarya','Patna','India','91930390800');
insert into office values('OF0005','Kartik Sharma','Jaipur','Russia','919331430600');
insert into office values('OF0006','Abhinav Sikarwar','Bharatpur','Israel','917014192750');
```

Figure 1 Create table and Insert values



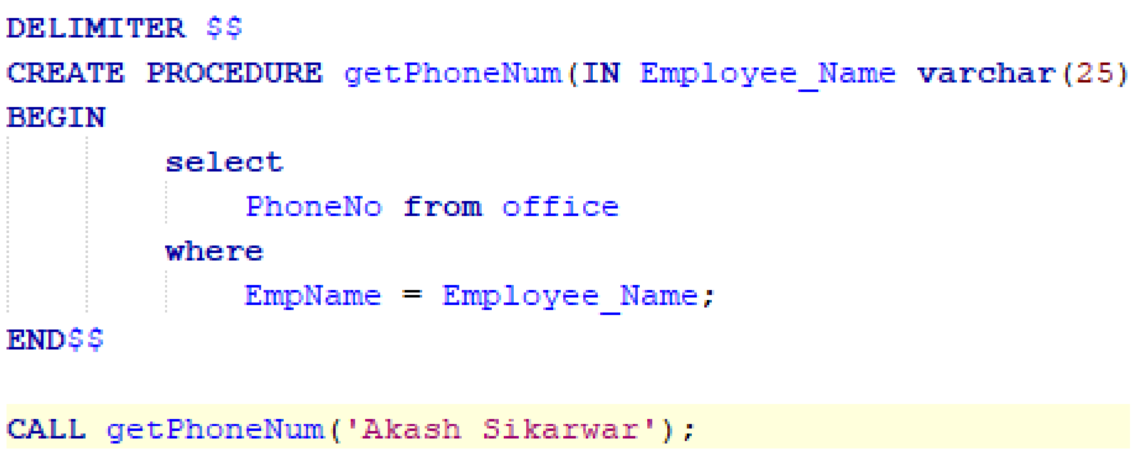
```
DELIMITER $$
CREATE PROCEDURE counter(INOUT count INT, IN inc INT)
BEGIN
        SET count = count + inc;
END$$

SET @count = 100;
CALL counter(@count, 50);
select @count;
```

Figure 2 Procedure for counter and Calling the Procedure



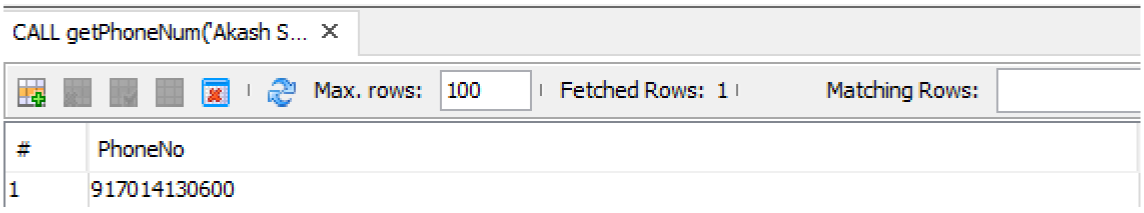
```
select @count ×
Max. rows: 100 | Fetched Rows: 1 | Matching Rows:
#   @count
1                150
```

Figure 3 Result of the counter procedure call



```
DELIMITER $$
CREATE PROCEDURE getPhoneNum(IN Employee_Name varchar(25))
BEGIN
        select
            PhoneNo from office
        where
            EmpName = Employee_Name;
END$$

CALL getPhoneNum('Akash Sikarwar');
```

Figure 4 Procedure Call to get Phone Number

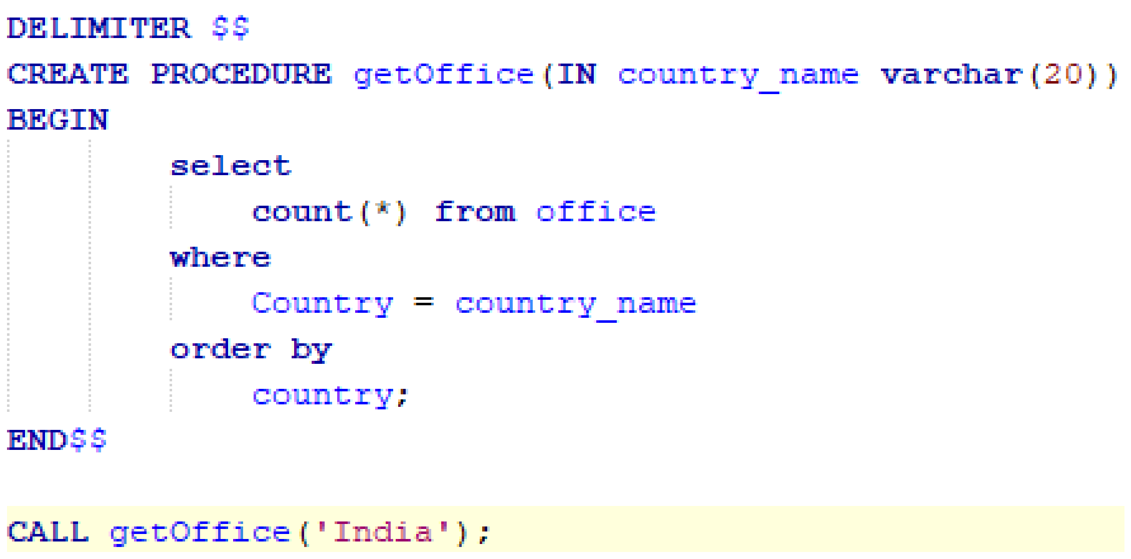


CALL getPhoneNum('Akash S... ×

Max. rows: 100 | Fetched Rows: 1 | Matching Rows:

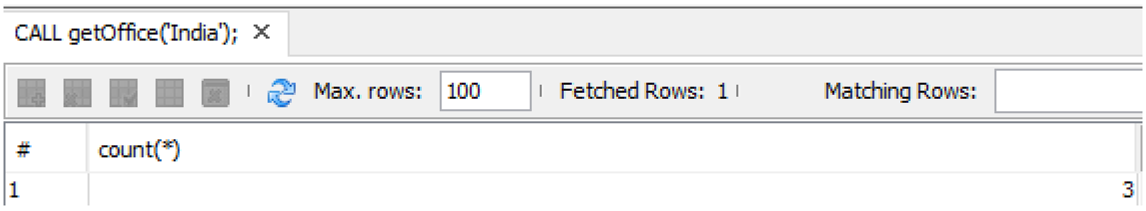
#	PhoneNo
1	917014130600

Figure 5 Result of the getPhoneNo procedure call



```
DELIMITER $$
CREATE PROCEDURE getOffice(IN country_name varchar(20))
BEGIN
        select
            count(*) from office
        where
            Country = country_name
        order by
            country;
END$$

CALL getOffice('India');
```

Figure 6 Procedure call to count number of offices in a particular country



CALL getOffice('India'); ×

Max. rows: 100 | Fetched Rows: 1 | Matching Rows:

#	count(*)
1	3

Figure 7 Result of getOffice procedure call with Argument "India"

6. Conclusions

In MySQL, a parameter has one of three modes: IN, OUT, or INOUT.

- IN is the default mode. When you define an IN parameter in a stored procedure, the calling program has to pass an argument to the stored procedure. The value of an IN parameter is protected. It means that even the value of the IN parameter is changed inside the stored procedure, its original value is retained after the stored procedure ends. In other words, the stored procedure only works on the copy of the IN parameter.
- The value of an OUT parameter can be changed inside the stored procedure and its new value is passed back to the calling program. Notice that the stored procedure cannot access the initial value of the OUT parameter when it starts.
- An INOUT parameter is a combination of IN and OUT parameters. It means that the calling program may pass the argument, and the stored procedure can modify the INOUT parameter, and pass the new value back to the calling program.

7. Comments

1. Limitations of Experiments

- It's difficult to debug stored procedures. Unfortunately, MySQL does not provide any facilities to debug stored procedures like other enterprise database products such as Oracle and SQL Server.
- Developing and maintaining stored procedures often requires a specialized skill set that not all application developers possess. This may lead to problems in both application development and maintenance.

2. Recommendations

Stored procedures help reduce the network traffic between applications and MySQL Server. Because instead of sending multiple lengthy SQL statements, applications have to send only the name and parameters of stored procedures.