MySQL
Practical 7 Stored Procedure & Function

Compound Statement for stored procedure

1. BEGIN ... END Compound Statement

which can appear within stored programs (stored procedures and functions, triggers.

```
BEGIN
  Statement list
END
```

2. DECLARE Statement

The DECLARE statement is used to define various items local to a program:
Local variable,conditions .

3. DECLARE LOCAL variable.

This statement declares local variables within stored programs.

DECLARE *var_name* [, *var_name*] ... *type* [DEFAULT *value*]

e.g declare no int

4. Assign values to variable

Set is used to assign the value to variable.

```
SET variable_name = value;
```

```
e.g  DECLARE total INT DEFAULT 0;
       SET total = 10;
```

Another way to assign value to the variable **which is fetch from select statement**.

e.g

```
DECLARE productCount INT DEFAULT 0;

SELECT COUNT(*)
INTO productCount
FROM products;
```

5. IF Statement

```
IF search_condition THEN
        statement_list;
   [ELSEIF search_condition THEN
         statement_list] ...;
   [ELSE
```

```
          statement_list ;]
END IF;
```

6. WHILE Statement

```
WHILE search_condition DO
   statement_list
END WHILE;
```

STORED PROCEDURE

Syntax :

```
CREATE PROCEDURE procedure_name(parameter_list)
BEGIN
  statements;
END
```

- To execute a stored procedure, you use the CALL statement:
  Mysql>CALL stored_procedure_name(argument_list);

- To drop the procedure

Mysql> drop procedure procedure_name;

- **To Update procedure**
  1. First drop the procedure
  2. create again the procedure

E.G : simple procedure to display hello message.

Step 1 : change delimiter to $$

```
     mysql> delimiter $$
```

step 2 : create procedure

```
       mysql> create procedure disp()
 begin
  select "hello";
end $$
```

step 3 : change delimiter to ;

```
mysql>delimiter;
```

call the procedure

```
mysql> call disp();

+-------+
| hello |
+-------+
| hello |
+-------+
```

e.g **declare a variable and assign value and display value**

```
mysql>delimiter $$    // change delimiter to $$
mysql> create procedure vartest()
        begin
           declare n int(2);
          set n =10;
         select n;
          end $$
Query OK, 0 rows affected, 1 warning (0.13 sec)

mysql> delimiter ;     / / change delimiter back to ;
mysql> call vartest();
+------+
| n    |
+------+
|   10 |
+------+
```

e.g procedure containing if loop

```
mysql > delimiter //

mysql > create procedure iftest()
           begin
         declare n int(2);
          set n = 10;
            if n>10 then
                    select concat(n,' is greater');
           else
            select concat(n,' is smaller') ;
         end if;
         end //

mysql> delimiter ;                                  -> set delimiter semicolon
```

call the procedure

```
mysql>   call iftest();
+---------+
| smaller |
+---------+
| smaller |
```

e.g procedure of while loop

step 1 change delimiter to $$

```
 mysql> delimiter $$
```

step 2 : create procedure

```
create procedure whiletest()
begin
  declare n int(2) ;
  set n = 10;
  while n > 0 do
      select n;
     set n = n -1;
  end while;
end $$
```

step 3 : change delimiter to ;

mysql> delimiter ;

step 4 : call the procedure

mysql> call whiletest();

e.g procedure using the query

```
mysql > delimiter $$

mysql>  create procedure selecttest()
begin
  declare n int(2) ;
  set n = 20;
  select *from employee where age > n;
end $$

mysql> delimiter ;
```

mysql> call selecttest();

```
+--------------+------+--------+-------------+------------+----------+------------+
| name         | age  | city   | designation | department | salary   | joindate   |
+--------------+------+--------+-------------+------------+----------+------------+
| rohan patel  |   26 | NULL   | salesman    | sales      |  9000.00 | NULL       |
| virat        |   32 | mumbai | admin       | admin      | 10000.00 | NULL       |
| sameer       |   32 | mumbai | accountant  | admin      | 12000.00 | 2011-09-27 |
| hares        |   24 | NULL   | salesman    | sales      | 11000.00 | NULL       |
```

- **Parameters in stored procedure.**

Parameter syntax :

[IN | OUT | INOUT] parameter_name datatype[(length)]

IN Parameter

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.

OUT 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.

IN-OUT Parameter:
n 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.

IN parameter example
e.g Create a procedure which takes two parameters for department and age. And display records from employee table for department and age is greater than given age in parameter.
IN parameter will pass the value to the procedure
Step 1 : delimiter $$

Step 2 :
```
  mysql> create procedure display(IN dept varchar(30), IN a int(2) )
    begin
          select *from employee where department=dept and age > a ;
     end $$
```

Step 3 : delimiter ;

Step 4 : call procedure.
Mysql > call display('production',25);

-OUT Prarameter
e.g procedure for demonstrate OUT parameter

create procedure which pass the department name as arguments calculate total of the salary of given procedure. Total of salary is stored in OUT parameter.

Step 1 : delimiter $$

Step 2 : -
```
         create procedure outtest(IN dept varchar(10) , OUT total int)
  begin
    select sum(salary) into total from employee where department=dept;
  end $$
```

step 3 : delimiter ;

step 4 : **call outtest('production',@total);**

step 5 : **select @total;**

Display the List of procedure in the database

Syntax : show procedure status where db = ‘databasename’;

e.g mysql>show procedure status where db=’testdb’;

Stored Functions

A stored function is a special kind stored program that returns a single value. They are reusable among SQL statements or stored programs.

Different from a stored procedure, you can use a stored function in SQL statements wherever an expression is used.

Syntax for creating Function

```
CREATE FUNCTION function_name(
    param1 datatype, param2 datatype,…

      )
RETURNS datatype;
BEGIN
 -- statements

  RETURN expression;


END $$
```

By default, all parameters are the IN parameters. **You cannot specify IN , OUT or INOUT modifiers** to parameter like stored procedure in Function

write the code in the body of the stored function in the BEGIN END block

Call the function

Function can be called by select statement or within a stored procedure also.

e.g Create function which work like pow function. Pass 2 arguments no1 and no2 and returns the multiplication of no1 and no2

Create function

Step -1 change delimiter.

Mysql> delimiter $$

Step -2 define the function

```
create function powfunct(b int,p int)
returns int(4)
```

```
          DETERMINISTIC
begin
               declare temp int;
               set temp = b*p;
            return (temp);
       end $$
```

Step 4 : change delimiter

Mysql> delimiter ;

Step 3 : call the function

Mysql> select powfunct(3,4) ;

```
  powfunct(3,4)
          12
```

e.g 2 create function which calculate the bonus of the employee based on salary

step 1 : mysql> delimiter $$

```
step 2  : mysql> create function calbonus( sal int)
 returns float(10,2)
DETERMINISTIC
begin

  declare bonus float(10,2);

  if sal > 15000 then
     set bonus = sal * 0.10 ;
  else
     set bonus = sal * 0.15;
  end if;

  return (bonus);
  end $$
```

step 3 : mysql> delimiter ;

step 4 mysql> select name, salary , calbonus(salary) from employee;

Exercise for procedure

1 Create procedure called proc1 which declare one integer variable and one varchar variable and display both the variables.

2. Create procedure called proc2 in which declare the variable counter = and execute while loop   until counter > 0 .

3. create procedure called proc3, which pass the argurment N. and procedure make total of first N number. E.g N =5 then sum = (1+2+3+4+5) = 15 use while loop.

4. create a procedure called proc4 which pass the student id in parameter and find average of marks of given student id from stud_sub table. E.g call proc2(1)

 5. Create procedure called proc5 in which pass the number and display whether number is odd or even. [hint if mod(n,2) = 0 then]

6. create procedure called proc6 which pass the orderid as parameter and find the total quantity order form sales_order_detail, total of quantity order should be stored in OUT parameter.

Exercise for procedure

1. Create a function func1 which takes the number as parameter and return the value “odd” or “even” .

2. Create a function func2() which take the age attribute of employee table, if age is <=25 status will be “young”, if age between 26 to 32 status ”middle” if age > 32 status will be “old”. Function returns the status. Write a select query which display the name ,age and status of every employee.

3. Create a function fun3() which takes a orderno as input and returns the name(description) of the product . hint(use product_master and sales_order_detail) Use necessary select query to display function output.