Stored Functions

Introduction

A **stored function** is a special kind of stored program that returns a single value. You use stored functions to encapsulate common formulas that are reusable among SQL statements or stored programs.

Diferent from a stored procedure, you can use a stored function in SQL statements wherever an expression is used. This helps improve the readability and maintainability of the procedural code.

Contd...

First, you specify the name of the stored function after CREATE FUNCTION clause.

Second, you list all parameters of the stored function inside the parentheses. By default, all parameters are the IN parameters. You cannot specify IN , OUT or INOUT modifers to the parameters.

Third, you must specify the data type of the return value in the RETURNS statement. It can be any valid MySQL data types.

Functions

```
mysql> delimiter //
mysql> create function mysquare(val int)
    -> returns int
    -> begin
    -> declare result int;
    -> set result=0;
    -> set result=val*val;
    -> return result;
    -> end //
Query OK, 0 rows affected (0.09 sec)

mysql> delimiter ;
mysql> select mysquare(3);
+--------------+
| mysquare(3) |
+--------------+
|            9 |
+--------------+
1 row in set (0.04 sec)
```

Diferences between stored functions and stored procedures

1.A FUNCTION **always returns** a value using the return statement. **-** Practical scenarios, when expecting a value to be returned which in turn helps for computation in rest of code

PROCEDURE **may return** one or more values **through parameters** or may not return any at all.

IN,OUT,INOUT parameters are different types. IN will be the input to the procedure. OUT will be the output from the procedure and this helps to get the output from the procedure. INOUT usually a same parameter behaves as input as well as output.

2.Functions are normally used for **computations** where as **procedures** are normally used for **executing business logic**.

3. A Function returns 1 value only. Procedure can return multiple values.

Contd...

4.Stored procedure always returns an integer value of zero by default. Whereas function return types could be scalar or table or table values.- This is because Functions mainly meant for computation

5.A function can be called directly by SQL statement like select func_name while procedures cannot.

6.Stored procedure has the security and reduces the network trafc and also we can call stored procedure in any no. of applications at a time.

7.A Function can be used in the SQL Queries while a procedure cannot be used in SQL queries .that cause a major diference b/w function and procedures.

How to find factorial with mysql recursive stored procedure with example?

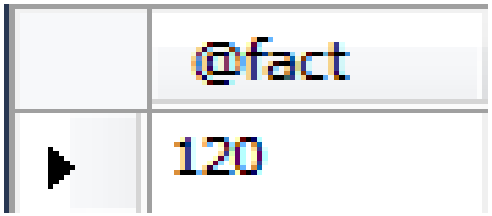
```
DROP PROCEDURE IF EXISTS find_fact;
DROP PROCEDURE IF EXISTS factorial;
 DELIMITER $$
CREATE PROCEDURE find_fact(IN n INT)
BEGIN
 CALL factorial(n,@fact);
SELECT @fact;
END$$

DELIMITER $$
CREATE PROCEDURE factorial(IN n INT,OUT fact INT)
BEGIN
  IF n = 1 THEN
SET fact := 1;
 ELSE
CALL factorial(n-1,fact);
SET fact := n * fact;
 END IF;
END$$
```

Contd...

Calling Procedure & Output

```
CALL find_fact(5);
```



	@fact
▶	120

```
DROP PROCEDURE IF EXISTS `multipleCursorsAtOne`;
DELIMITER $$
CREATE PROCEDURE `multipleCursorsAtOne`()
BEGIN
    DROP TABLE IF EXISTS userNames;
    CREATE TEMPORARY TABLE userNames
    (userName varchar(200) NOT NULL);

    BEGIN
        DECLARE done BOOLEAN DEFAULT false;
        DECLARE p_first_name VARCHAR(200);
        DECLARE cursor_a CURSOR FOR SELECT user_name FROM user_info LIMIT 1,3;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

        OPEN cursor_a;

            cursor_a_loop: LOOP
                FETCH cursor_a INTO p_first_name;
                IF done THEN
                        LEAVE cursor_a_loop;
                END IF;
                -- cursor loop statements

                IF p_first_name IS NOT NULL AND p_first_name <> "" THEN
                   INSERT INTO userNames(userName) VALUES(p_first_name);
                END IF;
            END LOOP;

        CLOSE cursor_a;
    END;
```

Multiple cursors in mysql stored procedure with example

```
    BEGIN
        DECLARE done BOOLEAN DEFAULT false;
        DECLARE p_first_name VARCHAR(200);
        DECLARE cursor_a CURSOR FOR SELECT user_name FROM user_info LIMIT 4,3;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

        OPEN cursor_a;

            cursor_a_loop: LOOP
                FETCH cursor_a INTO p_first_name;
                IF done THEN
                        LEAVE cursor_a_loop;
                END IF;
                -- cursor loop statements

                IF p_first_name IS NOT NULL AND p_first_name <> "" THEN
                   INSERT INTO userNames(userName) VALUES(p_first_name);
                END IF;
            END LOOP;

        CLOSE cursor_a;
    END;

    SELECT * FROM userNames;
END
$$
```

OUTPUT:

	userName
▶	babu
	Vidhyaa
	Lakshanya
	Gopal
	VVS
	Apple