What is a 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 an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- Thus, Stored procedures are pre-compiled object which are compiled for the first time and its compiled format is saved which executes (compiled code) whenever it is called.
- 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.

Stored Procedure Syntax - MySQL

```
DELIMITER //
CREATE PROCEDURE procedure_name(param1,param2,...)
BEGIN
    Write Query
END //
DELIMITER ;
```

Execute a Stored Procedure

```
CALL procedure_name();
```

Creating a Stored Procedure in MYSQL

- The following SQL statement creates a stored procedure named “getEmloyee" that selects all records from the “Employee" table:

```
DELIMITER //
CREATE PROCEDURE getEmployee()
 BEGIN
    SELECT * FROM Employee ORDER BY ename;
 END //
 DELIMITER;
```

#Calling Procedure

```
CALL getEmployee();
```

The following SQL statement creates a stored procedure named "getEmployee" that selects name and age of customers whose age is greater than 20 from the "Employee" table:

```
DELIMITER //
CREATE PROCEDURE SelectAllCustomers()
BEGIN
 SELECT name,age FROM customer WHERE age>20;
END //
DELIMITER ;
```

To Execute:

```
CALL SelectAllCustomers();
```

Know it: *Only change query for insert, update and delete operations, other contents remain same.*

#DROP Stored Procedure

```
DROP PROCEDURE procedure_name;
```

Working with Parameters In MySQL, a parameter has one of three modes: IN, OUT, or INOUT

IN parameters

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. In addition, 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.

OUT parameters

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.

INOUT parameters

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.

The IN parameter example:

The following example creates a stored procedure that finds all records that locate in a city specified by the input parameter cityname.

```
DELIMETER//
CREATE PROCEDURE getRecordByCity( IN cityname varchar(50))
BEGIN
    SELECT * FROM Employee WHERE eaddress=cityname;
END//
DELIMETER;
```

Suppose now you want to find records locating pokhara, you need to pass an argument (pokhara) to the stored procedure as shown in the following query:

```
CALL getRecordByCity('Pokhara');
```

The OUT parameter example

The following stored procedure returns the number of employees by address

```
DELIMETER//
CREATE PROCEDURE getTotalEmployee( IN eadd varchar(50),
                                   OUT Total int)
BEGIN
    SELECT COUNT(*) INTO Total FROM Employee WHERE address=eadd;
END//
DELEMETER;
```

To get the number of employees whose address is in pokhara, you call the stored procedure getTotalEmployee() as follows :-

```
CALL getTotalEmployee('Pokhara',@Total);
SELECT @Total as TotalEmployee;
```

Inserting Records Using Parameters

Inserting sid, name and address in student table.

```
DELIMITER //
CREATE PROCEDURE insertRecords(
        IN id INT,
        IN nm VARCHAR(30),
        IN ad VARCHAR(30)
)
BEGIN
        INSERT INTO student VALUES(id,nm,ad);
END //
DELIMETER;
```

Executing,

CALL insertRecords(5,'Sunil','Ktm');

Updating Records Using Parameters

Update name and address on the basis of id in student table.

```
DELIMITER //
CREATE PROCEDURE updateRecords(
 IN id INT,
 IN nm VARCHAR(30),
 IN ad VARCHAR(30)
)
BEGIN
    UPDATE student set name=nm, address=ad WHERE sid=id;
END //
DELIMITER;
```

Executing,

```
CALL updateRecords(5,'Rejens','BTM');
```

Deleting Records Using Parameters

Delete records on the basis of id in student table.

```
DELIMITER //
CREATE PROCEDURE deleteRecords(
 IN id INT
)
BEGIN
 DELETE FROM student WHERE sid=id;
END //
```

Executing,

```
CALL deleteRecords(3);
```

INOUT parameter

```
DELIMITER $$

CREATE PROCEDURE counter(
        INOUT count INT,
    IN increment INT
)
BEGIN
        SET count = count + increment;
END$$
DELIMITER ;
set @count=100;
call counter(@count,10);
SELECT @count;
```