

# Procedures In MySQL

## Stored Procedures in MySQL

- Stored procedures in MySQL are declared using the following syntax:

```

Create Procedure <proc-name>
    (param_spec1, param_spec2, ...,
    param_specn)
begin
    -- execution code
end;
    
```

## Stored Procedures in MySQL

where each param\_spec is of the form:

```

[in | out | inout] <param_name>
<param_type>
    
```

- in mode: allows you to pass values into the procedure,
- out mode: allows you to pass value back from procedure to the calling program

## Example

```
mysql> select * from employee;
```

id	name	superid	salary	bdate	dno
1	john	3	100000	1960-01-01	1
2	mary	3	50000	1964-12-01	3
3	bob	NULL	80000	1974-02-07	3
4	tom	1	50000	1978-01-17	2
5	bill	NULL	NULL	1985-01-20	1

```
mysql> select * from department;
```

dnumber	dname
1	Payroll
2	TechSupport
3	Research

## Example

- Suppose we want to keep track of the total salaries of employees working for each department

```
mysql> create table deptsal as
-> select dnumber, 0 as totalsalary from department;
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select * from deptsal;
```

dnumber	totalsalary
1	0
2	0
3	0

We need to write a procedure to update the salaries in the deptsal table

## Example


```
mysql> delimiter //
```

**Step 1:** Change the delimiter (i.e., terminating character) of SQL statement from semicolon (;) to something else (e.g., //) So that you can distinguish between the semicolon of the SQL statements in the procedure and the terminating character of the procedure definition

### Example


```
mysql> delimiter //
mysql> create procedure updateSalary (IN param1 int)
-> begin
-> update deptsal
-> set totalsalary = (select sum(salary) from employee where dno = param1)
-> where dnumber = param1;
-> end; //
```

Query OK, 0 rows affected (0.01 sec)



### Example

- **Step 2:**
  1. Define a procedure called updateSalary which takes as input a department number.
  2. The body of the procedure is an SQL command to update the totalsalary column of the deptsal table.
  3. Terminate the procedure definition using the delimiter you had defined in step 1 (//)




### Example

```
mysql> delimiter //
mysql> create procedure updateSalary (IN param1 int)
-> begin
-> update deptsal
-> set totalsalary = (select sum(salary) from employee where dno = param1)
-> where dnumber = param1;
-> end; //
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> delimiter ;
```

**Step 3:** Change the delimiter back to semicolon (;)




### Example

```
mysql> call updateSalary(1);
Query OK, 0 rows affected (0.00 sec)

mysql> call updateSalary(2);
Query OK, 1 row affected (0.00 sec)

mysql> call updateSalary(3);
Query OK, 1 row affected (0.00 sec)
```

**Step 4:** Call the procedure to update the totalsalary for each department




### Example

```
mysql> select * from deptsal;
```

dnumber	totalsalary
1	100000
2	50000
3	130000

3 rows in set (0.00 sec)

**Step 5:** Show the updated total salary in the deptsal table



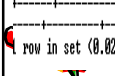
### Stored Procedures in MySQL

- Use **show procedure status** to display the list of stored procedures you have created

```
mysql> show procedure status;
```

Db	Name	Type	Definer	Modified	Created	Security
ptan	updateSalary0	PROCEDURE	ptan@%	2010-03-16 12:21:55	2010-03-16 12:21:55	DEFINER
			latin1	latin1_swedish_ci	latin1_swedish_ci	

1 row in set (0.02 sec)



## Stored Procedures in MySQL

- Use drop procedure to remove a stored procedure

```
mysql> drop procedure updateSalary;
Query OK, 0 rows affected (0.00 sec)
```



## Stored Procedures in MySQL

- You can declare variables in stored procedures
- You can use flow control statements (conditional IF-THEN-ELSE or loops such as WHILE and REPEAT)
- MySQL also supports cursors in stored procedures.
  - A cursor is used to iterate through a set of rows returned by a query so that we can process each individual row.



## IF

```
IF <condition> then
  <statements>
ELSEIF <condition> then
  <statements>
ELSE
  <statements>
END IF
```

- There can be any number of ELSIF clauses in your IF statement.



## Case Statement

- Two different syntaxes:

```
CASE <expression>
  WHEN <value> then
    <statements>
  WHEN <value> then
    <statements>
  ...
  ELSE
    <statements>
END CASE;
```



## CASE Statement (Continued)

```
CASE
  WHEN <condition> then
    <statements>
  WHEN <condition> then
    <statements>
  ...
  ELSE
    <statements>
END CASE;
```



## Looping

- [begin\_label:] LOOP
  - <statement list>
- END LOOP [end\_label]
- Note that the end\_label has to = the begin\_label
- Both are optional
- [begin\_label:] REPEAT
  - <statement list>
- UNTIL <search\_condition>
- END REPEAT [end\_label]



## While

- [begin\_label:] WHILE <condition> DO  
- <statements>
- END WHILE [end\_label]

## If Statement Example

```

1. DELIMITER //
2.
3. CREATE PROCEDURE `proc_IF` (IN param1 INT)
4. BEGIN
5.     DECLARE variable1 INT;
6.     SET variable1 = param1 + 1;
7.
8.     IF variable1 = 0 THEN
9.         SELECT variable1;
10.    END IF;
11.
12.    IF param1 = 0 THEN
13.        SELECT 'Parameter value = 0';
14.    ELSE
15.        SELECT 'Parameter value <> 0';
16.    END IF;
17. END //
    
```

## Case Example

```

1. DELIMITER //
2.
3. CREATE PROCEDURE `proc_CASE` (IN param1 INT)
4. BEGIN
5.     DECLARE variable1 INT;
6.     SET variable1 = param1 + 1;
7.
8.     CASE variable1
9.     WHEN 0 THEN
10.        INSERT INTO table1 VALUES (param1);
11.    WHEN 1 THEN
12.        INSERT INTO table1 VALUES (variable1);
13.    ELSE
14.        INSERT INTO table1 VALUES (99);
15.    END CASE;
16.
17. END //
    
```

## While Example

```

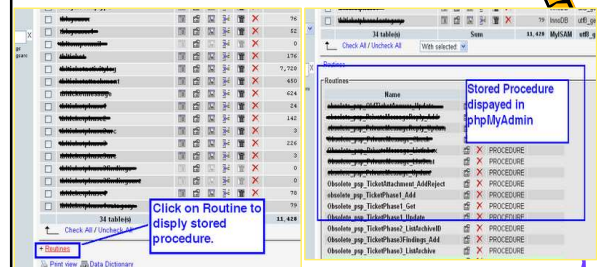
1. DELIMITER //
2.
3. CREATE PROCEDURE `proc_WHILE` (IN param1 INT)
4. BEGIN
5.     DECLARE variable1, variable2 INT;
6.     SET variable1 = 0;
7.
8.     WHILE variable1 < param1 DO
9.         INSERT INTO table1 VALUES (param1);
10.        SELECT COUNT(*) INTO variable2 FROM table1;
11.        SET variable1 = variable1 + 1;
12.    END WHILE;
13. END //
    
```

## Stored Procedures

How to View MySQL Stored Procedure in PhpMyAdmin?

1. In **PhpMyAdmin**, select the correct database on the left hand panel
2. You can see all the database tables in the right hand panel
3. Scroll down the right hand panel until the end, you will see **Routines**
4. Click on **Routines**
5. All the **Stored Procedures** will be displayed on screen

## Stored Procedures



## Stored Procedures

- Creating a MySQL Stored Procedure in PhpMyAdmin
- Developers may have some issues creating a stored procedure in **phpMyAdmin**, which seems has issues supporting (or rather, not supporting) stored procedures, but there is a simple fix.
- First off, let's look at some attempted stored procedure code:
  - Pretty easy, right? But it causes this error:

```

1 CREATE PROCEDURE spGetUsers()
2 BEGIN
3     SELECT UserName, FirstName, LastName
4     FROM tblUsers
5     ORDER BY LastName, FirstName;
6 END

```

#1064 – You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near " at line 3

## Stored Procedures

- **Changing the Delimiter**
  - The delimiter is the character or string of characters that you'll use to tell the MySQL client that you've finished typing in an SQL statement. For ages, the delimiter has always been a **semicolon (;)**. That, however, causes problems, because, in a stored procedure, one can have many statements, and each must end with a **semicolon**. What one can do is to change the delimiter to something else, something other than a semicolon.

## Changing the Delimiter

```

1 DELIMITER //
2 CREATE PROCEDURE GetAllProducts()
3 BEGIN
4     SELECT * FROM products;
5 END //
6 DELIMITER ;

```

## Creating a Stored Procedures

```

1 DELIMITER //
2 CREATE PROCEDURE GetAllProducts()
3 BEGIN
4     SELECT * FROM products;
5 END //
6 DELIMITER ;

```

## Calling a Stored Procedure

```

1. CALL stored_procedure_name (param1, param2, ...)
2.
3. CALL procedure1(10, 'string parameter', @parameter_var);

```

## Modify a Stored Procedure

- MySQL provides an **ALTER PROCEDURE** statement to modify a routine, but only allows for the ability to change certain characteristics. If you need to alter the body or the parameters, you must **drop** and recreate the procedure



## Drop (Delete) A Stored Procedure

```
1. DROP PROCEDURE IF EXISTS p2;
```

- This is a simple command. The **IF EXISTS** clause prevents an error in case the procedure does not exist.



## Variables (Stored Procedures)

- The following step will teach you how to define variables, and store values inside a procedure.
- You must declare them explicitly at the start of the **BEGIN/END** block, along with their data types. Once you've declared a variable, you can use it anywhere that you could use a session variable, or literal, or column name.



## Variables (Stored Procedures)

- Declare a variable using the following syntax:

```
1. DECLARE varname DATA-TYPE DEFAULT defaultvalue;
```

```
1. DECLARE a, b INT DEFAULT 5;
2.
3. DECLARE str VARCHAR(50);
4.
5. DECLARE today TIMESTAMP DEFAULT CURRENT_DATE;
6.
7. DECLARE v1, v2, v3 TINYINT;
```



## Variables (Stored Procedures)

### Working with Variables

Once the variables have been declared, you can assign them values using the **SET** or **SELECT** command:

```
1. DELIMITER //
2.
3. CREATE PROCEDURE `var_proc` (IN paramstr VARCHAR(20))
4. BEGIN
5.   DECLARE a, b INT DEFAULT 5;
6.   DECLARE str VARCHAR(50);
7.   DECLARE today TIMESTAMP DEFAULT CURRENT_DATE;
8.   DECLARE v1, v2, v3 TINYINT;
9.
10.  INSERT INTO table1 VALUES (a);
11.  SET str = 'I am a string';
12.  SELECT CONCAT(str,paramstr), today FROM table2 WHERE b >=5;
13. END //
```



## Example using Cursors

- The previous procedure updates one row in **deptsal** table based on input parameter
- Suppose we want to update all the rows in **deptsal** simultaneously
  - First, let's reset the totalsalary in **deptsal** to zero



### Example using Cursors

```
mysql> update deptsal set totalsalary = 0;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 3 Changed: 0 Warnings: 0

mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
+-----+-----+
3 rows in set (0.00 sec)
```

### Example using Cursors

```
mysql> delimiter $$
mysql> drop procedure if exists updateSalary;
Query OK, 0 rows affected (0.00 sec)

mysql> create procedure updateSalary()
-> begin
->     declare done int default 0;
->     declare current_dnum int;
->     declare dnumcur cursor for select dnumber from deptsal;
->     declare continue handler for not found set done = 1;
->
->     open dnumcur;
->
->     repeat
->         fetch dnumcur into current_dnum;
->         update deptsal
->             set totalsalary = (select sum(salary) from employee
->                                 where dno = current_dnum)
->         until done
->         end repeat;
->     close dnumcur;
-> end$$
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Drop the old procedure

Use cursor to iterate the rows

### Example using Cursors

- Call procedure

```
mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
+-----+-----+
3 rows in set (0.01 sec)

mysql> call updateSalary;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
| 1 | 100000 |
| 2 | 50000 |
| 3 | 130000 |
+-----+-----+
3 rows in set (0.00 sec)
```

### Another Example

- Create a procedure to give a raise to all employees

```
mysql> select * from emp;
+-----+-----+-----+-----+-----+-----+
| id | name | superid | salary | bdate | dno |
+-----+-----+-----+-----+-----+-----+
| 1 | john | 3 | 100000 | 1960-01-01 | 1 |
| 2 | mary | 3 | 50000 | 1964-12-01 | 3 |
| 3 | bob | NULL | 80000 | 1974-02-07 | 3 |
| 4 | tom | 1 | 50000 | 1978-01-17 | 2 |
| 5 | bill | NULL | NULL | 1985-01-20 | 1 |
| 6 | lucy | NULL | 90000 | 1981-01-01 | 1 |
| 7 | george | NULL | 45000 | 1971-11-11 | NULL |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

### Another Example

```
mysql> delimiter |
mysql> create procedure giveRaise (in amount double)
-> begin
->     declare done int default 0;
->     declare eid int;
->     declare sal int;
->     declare emprec cursor for select id, salary from employee;
->     declare continue handler for not found set done = 1;
->
->     open emprec;
->     repeat
->         fetch emprec into eid, sal;
->         update employee
->             set salary = sal + round(sal * amount)
->             where id = eid;
->     until done
->     end repeat;
-> end |
Query OK, 0 rows affected (0.00 sec)
```

### Another Example

```
mysql> delimiter ;
mysql> call giveRaise(0.1);
Query OK, 0 rows affected (0.00 sec)

mysql> select * from employee;
+-----+-----+-----+-----+-----+-----+
| id | name | superid | salary | bdate | dno |
+-----+-----+-----+-----+-----+-----+
| 1 | john | 3 | 110000 | 1960-01-01 | 1 |
| 2 | mary | 3 | 55000 | 1964-12-01 | 3 |
| 3 | bob | NULL | 88000 | 1974-02-07 | 3 |
| 4 | tom | 1 | 55000 | 1978-01-17 | 2 |
| 5 | bill | NULL | NULL | 1985-01-20 | 1 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

