Procedures

The syntax to create a procedure in MySQL is:

```
CREATE PROCEDURE procedure_name [ (parameter datatype [, parameter datatype]) ]

BEGIN

  declaration_section

  executable_section

END;
```

procedure_name: The name to assign to this procedure in MySQL.

Parameter Optional. One or more parameters passed into the procedure. When creating a procedure, there are three types of parameters that can be declared:

IN: An IN parameter lets you pass a value to the subprogram. It is a read-only parameter. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a constant, literal, initialized variable, or expression as an IN parameter. You can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. It is the default mode of parameter passing. Parameters are passed by reference.

OUT: An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. The actual parameter must be variable and it is passed by value.

IN OUT: An IN OUT parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read.

The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. Actual parameter is passed by value.

declaration_section
The place in the procedure where you declare local variables.
executable_section
The place in the procedure where you enter the code for the procedure.

The syntax to a drop a procedure in MySQL is:

```
DROP procedure [ IF EXISTS ] procedure_name;
```

Examples

```
--First pick a database to use (a procedure, like a table, is associated with
--a single database.) For these examples, I will use a database that is
populated
--with the tables from mydb:

USE mydb;


--Next, change the delimiter, because we will use the semicolon WITHIN the
--procedure declarations, and therefore it cannot be the delimiter anymore:

DELIMITER //



--OK, let's get started. Creating procedures is straightforward:

CREATE PROCEDURE myFirstProc()
  SELECT 'Hello World!' AS Output;
//

Query OK, 0 rows affected (0.00 sec)

--Whenever you create a procedure (successfully) you should get a 'Query OK'
message.

--Calling a procedure is also straightforward:

CALL myFirstProc() //

+--------------+
| Output       |
+--------------+
| Hello World! |
+--------------+
1 row in set (0.00 sec)

--By the way, procedure names are NOT case sensitive:

CALL myfirstproc() //

+--------------+
| Output       |
+--------------+
| Hello World! |
+--------------+
1 row in set (0.00 sec)
```

```
--OK, let's use some parameters:

DROP PROCEDURE IF EXISTS sayHello //


CREATE PROCEDURE sayHello(IN name VARCHAR(20))
  SELECT CONCAT('Hello ', name, '!') AS Greeting;
//

--The 'IN' keyword tells MySQL that is should be expecting an input value for
--the parameter......hunh? Why would a parameter NOT have an input value? You
will
--see in a little bit. First, let's see if sayHello works:

CALL sayHello('Venkat') //

+--------------+
| Greeting     |
+--------------+
| Hello Venkat! |
+--------------+
1 row in set (0.00 sec)


--Another example:

DROP PROCEDURE IF EXISTS saySomething //
CREATE PROCEDURE saySomething(IN phrase VARCHAR(20), IN name VARCHAR(20))
  SELECT CONCAT(phrase, ' ', name, '!') AS Output;
//

CALL saySomething('Go','Blue Jays') //
CALL saySomething('Do','my homework') //

+----------------+
| Output         |
+----------------+
| Go Blue Jays! |
+----------------+
1 row in set (0.00 sec)

+------------------+
| Output           |
+------------------+
| Do my homework! |
+------------------+
1 row in set (0.00 sec)
```

```
DROP PROCEDURE IF EXISTS calculate //

CREATE PROCEDURE calculate(IN x INT, IN y INT, OUT sum INT, OUT product INT)
BEGIN
  SET sum = x + y;
  SET product = x * y;
END;
//

--Did you notice the 'OUT' keyword for sum and product? This tells MySQL that
those
--two parameters are not 'input' parameters but are 'output' parameters
instead.
--Now, when calling the procedure, we need to provide four parameters: two
input
--values, and two MySQL *variables* where the results will be stored:

CALL calculate(4,5,@s,@p) //

Query OK, 0 rows affected (0.00 sec)

--Here, @s and @p are MySQL variables. Notice that they start with @,
although
--procedure *parameters* do not start with @

SELECT @s //
SELECT @p //

+-------+
| @s    |
+-------+
| 9     |
+-------+
1 row in set (0.00 sec)

+-------+
| @p    |
+-------+
| 20    |
+-------+
1 row in set (0.00 sec)

--Note: you can also have INOUT parameters, which serve as both input and
output
--parameters.
```

```
--OK, let's do some interesting stuff. First off, flow control:

DROP PROCEDURE IF EXISTS mySign //

CREATE PROCEDURE mySign(IN x INT)
BEGIN
  IF x > 0 THEN
    SELECT x AS Number, '+' AS Sign;
  ELSEIF x < 0 THEN
    SELECT x AS Number, '-' AS Sign;
  ELSE
    SELECT x AS Number, 'Zero' AS Sign;
  END IF;
END;
//

CALL mySign(2) //
CALL mySign(-5) //
CALL mySign(0) //

+--------+------+
| Number | Sign |
+--------+------+
|      2 | +    |
+--------+------+
1 row in set (0.00 sec)

+--------+------+
| Number | Sign |
+--------+------+
|     -5 | -    |
+--------+------+
1 row in set (0.00 sec)

+--------+------+
| Number | Sign |
+--------+------+
|      0 | Zero |
+--------+------+
1 row in set (0.00 sec)
```

```
--Using CASE:

DROP PROCEDURE IF EXISTS digitName //

CREATE PROCEDURE digitName(IN x INT)
BEGIN

  DECLARE result VARCHAR(20);

  CASE x
    WHEN 0 THEN SET result = 'Zero';
    WHEN 1 THEN SET result = 'One';
    WHEN 2 THEN SET result = 'Two';
    WHEN 3 THEN SET result = 'Three';
    WHEN 4 THEN SET result = 'Four';
    WHEN 5 THEN SET result = 'Five';
    WHEN 6 THEN SET result = 'Six';
    WHEN 7 THEN SET result = 'Seven';
    WHEN 8 THEN SET result = 'Eight';
    WHEN 9 THEN SET result = 'Nine';
    ELSE SET result = 'Not a digit';
  END CASE;

  SELECT x AS Digit, result AS Name;

END;
//

CALL digitName(0) //
CALL digitName(4) //
CALL digitName(100) //

+-------+------+
| Digit | Name |
+-------+------+
|     0 | Zero |
+-------+------+
1 row in set (0.00 sec)

+-------+------+
| Digit | Name |
+-------+------+
|     4 | Four |
+-------+------+
1 row in set (0.00 sec)

+-------+-------------+
| Digit | Name        |
+-------+-------------+
|   100 | Not a digit |
+-------+-------------+
1 row in set (0.00 sec)
```

```
--As you'd expect, we have loops. For example, WHILE loops:

DROP PROCEDURE IF EXISTS fact //

CREATE PROCEDURE fact(IN x INT)
BEGIN

  DECLARE result INT;
  DECLARE i INT;
  SET result = 1;
  SET i = 1;

  WHILE i <= x DO
    SET result = result * i;
    SET i = i + 1;
  END WHILE;

  SELECT x AS Number, result as Factorial;

END;
//

CALL fact(1) //
CALL fact(2) //
CALL fact(4) //
CALL fact(0) //

+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      1 |         1 |
+--------+-----------+
1 row in set (0.00 sec)

+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      2 |         2 |
+--------+-----------+
1 row in set (0.00 sec)

+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      4 |        24 |
+--------+-----------+
1 row in set (0.01 sec)

+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      0 |         1 |
+--------+-----------+
1 row in set (0.00 sec)
```

--There is also **REPEAT/UNTIL** loops:

```
DROP PROCEDURE IF EXISTS fact //

CREATE PROCEDURE fact(IN x INT)
BEGIN

  DECLARE result INT DEFAULT 1;  /* notice you can declare a variable*/
  DECLARE i INT DEFAULT 1;       /* and give it a value in one line */

  REPEAT
    SET result = result * i;
    SET i = i + 1;
  UNTIL i > x
  END REPEAT;

  SELECT x AS Number, result as Factorial;

END;
//

CALL fact(1) //
CALL fact(2) //
CALL fact(4) //
CALL fact(0) //
```

```
+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      1 |         1 |
+--------+-----------+
1 row in set (0.00 sec)

+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      2 |         2 |
+--------+-----------+
1 row in set (0.00 sec)

+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      4 |        24 |
+--------+-----------+
1 row in set (0.00 sec)

+--------+-----------+
| Number | Factorial |
+--------+-----------+
|      0 |         1 |
+--------+-----------+
1 row in set (0.00 sec)
```