MySQL Stored Procedure and User-Defined Function

Stored Procedure

- A stored procedure is a program with SQL code which is stored in the database catalog and can be invoked later by a program, a trigger or even a stored procedure.
- MySQL supports stored procedure since version 5.0 to allow MySQL more flexible and powerful.

Three Ways to Create A Procedure

- 1. Save the procedure commands in a text file.
- 2. Use the phpMyAdmin utility to enter commands
  - Routine/Add routine
- 3. Enter the commands using the MySQL command prompt.

Example of a command file

```
DELIMITER //

CREATE PROCEDURE Hello()
LANGUAGE SQL
DETERMINISTIC
SQL SECURITY DEFINER
COMMENT 'A procedure'
BEGIN
   SELECT 'Hello World !';
END
//
```

Optional characteristics

- Type: Procedure/Function
- Language :the default value is SQL.
- Deterministic : If the procedure always returns the same results, given the same input. The default value is NOT DETERMINISTIC.
- SQL Security : At call time, check privileges of the user. INVOKER is the user who calls the procedure. DEFINER is the creator of the procedure. The default value is DEFINER.
- Comment : For documentation purposes; the default value is ""

Run a procedure

- With the command prompt:

```
CALL stored_procedure_name (param1, param2, ....);
```

CREATE PROCEDURE ProcName()

- Stored procedure names are case insensitive
- A procedure may have parameters

Define parameters within a stored procedure

- Parameter list is empty
  - CREATE PROCEDURE proc1 () :
- Define input parameter with key word IN:
  - used to send values to stored procedures.
  - CREATE PROCEDURE proc1 (IN varname DATA-TYPE)
  - The word IN is optional because parameters are IN (input) by default.
- Define output parameter with OUT:
  - used to get values from stored procedures
  - CREATE PROCEDURE proc1 (OUT varname DATA-TYPE)
- A procedure may have input and output paramters:
  - used to send values and get values from stored procedures.
  - CREATE PROCEDURE proc1 (INOUT varname DATA-TYPE)

Executable Section

- BEGIN

    Statements

- END

Examples of parameters

```
CREATE PROCEDURE proc_IN (IN var1 INT)
BEGIN
   SELECT var1 + 2 AS result;
END
```

```
CREATE PROCEDURE proc_OUT(OUT var1 VARCHAR(100))
BEGIN
 SET var1 = 'This is a test';
END
```

```
CREATE PROCEDURE proc_INOUT (IN var1 INT,OUT
var2 INT)
BEGIN
   SET var2 = var1 * 2;
END
```

Variable Declaration

- DECLARE variable_name datatype(size) DEFAULT default_value;
- Variable naming rules: Identifiers can consist of any alphanumeric characters, plus the characters '_' and '$'. Identifiers can start with any character that is legal in an identifier, including a digit. However, an identifier cannot consist entirely of digits.
- Data types:A variable can have any MySQL data types. For example:
  - Character: CHAR(n), VARCHAR(n)
  - Number: INT, SMALLINT, DECIMAL(i,j), DOUBLE
  - Date: DATE, TIME, DATETIME
  - BOOLEAN
- http://www.mysqltutorial.org/mysql-data-types.aspx

Examples

```
DECLARE x, y INT DEFAULT 0;

DECLARE today TIMESTAMP DEFAULT CURRENT_DATE;

DECLARE ename VARCHAR(50);

DECLARE no_more_rows BOOLEAN;
SET no_more_rows = TRUE;
```

Assigning variables

- Using the SET command:

```
DECLARE total_count INT DEFAULT 0;
SET total_count = 10;
```

Using the SELECT INTO command:

```
DECLARE total_products INT DEFAULT 0;
SELECT COUNT(*) INTO total_products
FROM products;
```

SELECT … INTO

- SELECT columns separated by commas
- INTO variables separated by commas
- FROM tablename
- WHERE condition;
- Ex:
  - SELECT cid, cname INTO custID, customername
  - FROM customer
  - WHERE cid = ‘C01’;

Arithmetic and string operators

- Arithmetic operators:

  +, -, *, /

- Modulo operator:
  - % or mod
- Other math calculations use math functions:
  - Pow(x,y)
- Concatenation uses CONCAT function:
  - SELECT CONCAT('New ', 'York ', 'City');

MySQL Comparison Operators

- EQUAL(=)
- LESS THAN(<)
- LESS THAN OR EQUAL(<=)
- GREATER THAN(>)
- GREATER THAN OR EQUAL(>=)
- NOT EQUAL(<>,!=)
-

Logical Operators

- Logical AND:
  - AND, &&
  - UnitsInStock < ReorderLevel AND CategoryID=1
  - UnitsInStock < ReorderLevel && CategoryID=1
- Negates value:
  - NOT, !
- Logical OR:
  - ||, OR
  - CategoryID=1 OR CategoryID=8
  - CategoryID=1 || CategoryID=8

IF statement: The IF statement can have THEN, ELSE, and ELSEIF clauses, and it is terminated with END IF.

```
IF variable1 = 0 THEN
        SELECT variable1;
END IF;
```

```
IF param1 = 0 THEN
        SELECT 'Parameter value = 0';
ELSE
        SELECT 'Parameter value <> 0';
END IF;
```

CASE Statement

```
CREATE PROCEDURE proc_CASE(IN param1 INT)
BEGIN
    DECLARE variable1 INT;
    SET variable1 = param1 + 1;
   CASE variable1
        WHEN 0 THEN
            INSERT INTO table1 VALUES (param1);
        WHEN 1 THEN
           INSERT INTO table1 VALUES (variable1);
         ELSE
             INSERT INTO table1 VALUES (99);
  END CASE;
 END
```

WHILE *cond* DO statement

```
CREATE PROCEDURE proc_WHILE (IN param1 INT)
BEGIN
   DECLARE variable1, variable2 INT;
    SET variable1 = 0;
    WHILE variable1 < param1 DO
            INSERT INTO table1 VALUES (param1);
            SELECT COUNT(*) INTO variable2 FROM table1;
            SET variable1 = variable2;
   END WHILE;
END
```

Comment Syntax

- From a /* sequence to the following */ sequence.
- From a “#” character to the end of the line.
- From a “-- ” sequence to the end of the line. In MySQL, the “-- ” (double-dash) comment style requires the second dash to be followed by at least one whitespace

  -- Programmer: John Smith

A Procedure to compute tax that takes sidIN and taxRate as inputs and return taxOut as output

```
DELIMITER //
CREATE PROCEDURE Caltax(sidIN char(5), taxRate double, out
taxOut double)

BEGIN
    DECLARE tax DOUBLE;
    DECLARE empSalary DOUBLE;
    select Salary into empSalary from salesreps where sid = sidIN;
    set taxOut=taxRate*empSalary;
END
//
```

- Note 1: No need to surround the sidIN with quotation mark:

```
    select Salary into empSalary from salesreps
where sid = sidIN;
```

- Note 2: The delimiter is changed to // to enable the entire definition to be passed to the server as a single statement. It can be restored to “;”
- mysql>delimiter ;

User-Defined Temporary Variables

- User variables are written as @var_name.

```
mysql> SET @t1=1, @t2=2, @t3:=4;
mysql> SELECT @t1, @t2, @t3, @t4 :=
@t1+@t2+@t3;
+------+------+------+--------------------+
| @t1  | @t2  | @t3  | @t4 := @t1+@t2+@t3 |
+------+------+------+--------------------+
|    1 |    2 |    4 |                  7 |
+------+------+------+--------------------+
```

Example of running the procedure from the command prompt

```
mysql> delimiter ;
mysql> set @tax=0;
Query OK, 0 rows affected (0.00 sec)

mysql> call caltax('S1',0.1,@tax);
Query OK, 1 row affected (0.00 sec)

mysql> select @tax;
+------+
| @tax |
+------+
|  650 |
+------+
1 row in set (0.00 sec)
```

First, check if the customer exist before adding a new order

```
DELIMITER //
CREATE PROCEDURE addOrder(oidIN char(5), cidIN char(5), sidIN char(5),
odateIN date)


BEGIN
    DECLARE cidTemp char(5) default "x";
    select cid into cidTemp from customers where cid = cidIN;
    IF cidTemp=cidIN THEN
        insert into orders values(oidIN,cidIN,sidIN,odateIN);
    END IF;
END
//


mysql> call addOrder('O8','C12','S1','2013-06-10');
Query OK, 0 rows affected, 1 warning (0.00 sec) because C12 not exist!
```

Example:Procedure showCustomers

```
DELIMITER //
DROP PROCEDURE IF EXISTS showCustomers;
CREATE PROCEDURE showCustomers ()

BEGIN
    Select * from Customers;
END
//

DELIMITER ;
```

User Defined Functions

- Stored functions differ from stored procedures in that stored functions actually return a value.
- Stored functions have only input parameters (if any parameters at all), so the IN , OUT , and INOUT keywords aren’t used.
- Stored functions have no output parameters; instead, you use a RETURN statement to return a value whose type is determined by the RETURNS type statement, which precedes the body of the function.

Example

```
DELIMITER //
DROP FUNCTION IF EXISTS empTax;
CREATE FUNCTION empTax(Salary Decimal(10,2)) RETURNS
    Decimal(10,2)
BEGIN
Declare tax decimal(10,2);
if salary < 3000.00 then
     set tax=salary*0.1;
elseif Salary <5000.00 then
     set tax=Salary*0.2;
else
     set tax=Salary*0.3;
end if;
return tax;
END
//
```

Invoking a Stored Function

- No CALL like stored procedure
- Just as a regular function

```
mysql> SELECT get_store_id(1);
```

Using the User-defined Function with SQL

```
mysql> delimiter ;
mysql> select sname, emptax(Salary) as tax from salesreps;
+-------+---------+
| sname | tax     |
+-------+---------+
| PETER | 1950.00 |
| PAUL  | 2160.00 |
| MARY  | 2250.00 |
+-------+---------+
3 rows in set (0.00 sec)
```