Part 3

- Welcome back… sorry I'm a little quiet when doing the demo in phpMyAdmin.
- In this video I'll explain how we can create a procedure that uses two select statements. The procedure uses two variables, one to get something IN and one to get something OUT.
  - A little more complicated but not too complex and a nice example of IN and OUT
- We end with flow control and cursors

```
BEGIN
       SELECT
             customerNumber,
        SUM(amount)
        FROM payments
        GROUP BY customerNumber
        HAVING SUM(amount)> tot_amount
        ORDER BY
             SUM(amount) DESC;

    SELECT SUM(T.tot)
    INTO g_total
    FROM (
            SELECT
                customerNumber,
                SUM(amount) as tot
            FROM payments
            GROUP BY customerNumber
            HAVING SUM(amount)> tot_amount
        ) AS T;
END
```

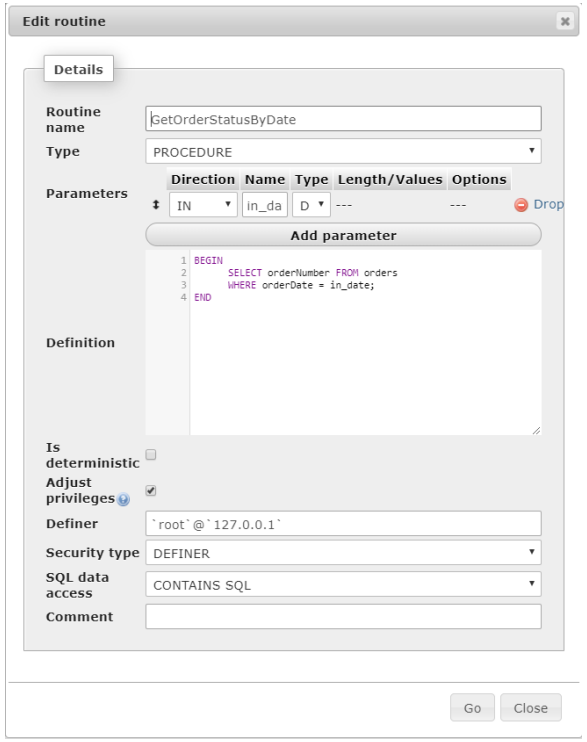
INOUT Example

```
DELIMITER $$
CREATE PROCEDURE SetCounter(
    INOUT counter INT,
    IN inc INT
)
BEGIN
    SET counter = counter + inc;
END $$
DELIMITER ;

SET @countValue = 10;
CALL SetCounter(@countValue, 50);
SELECT @countValue;
```

Altering a Stored Procedure

- MySQL does not have any statement that allows you to directly modify the parameters and body of the stored procedure.
- To make such changes, you must drop and re-create the stored procedure using the DROP PROCEDURE and CREATE PROCEDURE statements.
- However, MySQL Workbench, phpMyAdmin does offer this functionality



Variables

- Just like in programming languages we can create variables
- To declare a variable you use the DECLARE keyword
- The name of your variable
- Then you specify the data type
- Then the default value (if you omit DEFAULT the value will be NULL)
- To assign a value to a variable use the SET keyword
- You can also assign the result of a query to a variable using SELECT INTO

Variable Example

```
DELIMITER $$
CREATE PROCEDURE GetTotalOrder()
BEGIN
    DECLARE totalOrder INT DEFAULT 0;

    SELECT COUNT(*)
    INTO totalOrder
    FROM orders;

    SELECT totalOrder;
END $$
DELIMITER ;

CALL GetTotalOrder();
```

Showing the Stored Procedures

```
+-------------------------+
| routine_name            |
+-------------------------+
| GetNumEmployeeBySurname |
| GetOrderStatusByDate    |
| GetTotalOrder           |
| SetCounter              |
+-------------------------+
4 rows in set (0.04 sec)
```

```
SELECT
    routine_name
FROM
    information_schema.routines
WHERE
    routine_type = 'PROCEDURE'
    AND routine_schema = 'classicmodels';
```

Demo

Showing the Stored Procedures

```
+-------------------------+
| routine_name            |
+-------------------------+
| GetNumEmployeeBySurname |
| GetOrderStatusByDate    |
| GetTotalOrder           |
| SetCounter              |
+-------------------------+
4 rows in set (0.04 sec)
```

```
SELECT
    routine_name
FROM
    information_schema.routines
WHERE
    routine_type = 'PROCEDURE'
    AND routine_schema = 'classicmodels';
```

Flow Control

- MySQL supports various forms of flow control, all of which should be familiar (albeit slightly different syntax).
- IF-THEN, IF-THEN-ELSE, IF-THEN-ELSEIF-ELSE
- CASE-WHEN-THEN, ELSE
- LOOP – LEAVE
- WHILE-DO
- REPEAT-UNTIL

Probably not a good idea to show you the SQL for all the variations. You know how conditional statements work.

Let's look at an example of an IF and one of the loops

IF THEN Example

```
IF credit > 50000 THEN
    SET pCustomerLevel = 'PLATINUM';
ELSEIF credit <= 50000 AND credit > 10000 THEN
    SET pCustomerLevel = 'GOLD';
ELSE
    SET pCustomerLevel = 'SILVER';
END IF;
```

```
CREATE PROCEDURE GetCustomerLevel(
    IN  pCustomerNumber INT,
    OUT pCustomerLevel  VARCHAR(20))
BEGIN
    DECLARE credit DECIMAL(10,2) DEFAULT 0;

    SELECT creditLimit INTO credit FROM customers
    WHERE customerNumber = pCustomerNumber;

    IF credit > 50000 THEN
        SET pCustomerLevel = 'PLATINUM';
    END IF;
END
```

Case Example

```
CASE customerCountry
    WHEN  'USA' THEN
       SET pShipping = '2-day Shipping';
    WHEN 'Canada' THEN
       SET pShipping = '3-day Shipping';
    ELSE
       SET pShipping = '5-day Shipping';
END CASE;
```

MySQL Cursor

- A cursor allows you to iterate through a record set within a stored procedure

- Read Only
- Non-Scrollable
- Asensitive

Cursor Example

```
DELIMITER $$
CREATE PROCEDURE createEmailList (
    INOUT emailList varchar(4000)
)
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE emailAddress varchar(100) DEFAULT "";

    -- declare cursor for employee email
    DECLARE curEmail
        CURSOR FOR
            SELECT email FROM employees;
```

Cursor Example

```
 -- declare NOT FOUND handler
    DECLARE CONTINUE HANDLER
        FOR NOT FOUND SET finished = 1;
    OPEN curEmail;
    getEmail: LOOP
        FETCH curEmail INTO emailAddress;
        IF finished = 1 THEN
            LEAVE getEmail;
        END IF;
        -- build email list
        SET emailList = CONCAT(emailAddress,";",emailList);
    END LOOP getEmail;
    CLOSE curEmail;
END$$
DELIMITER ;
```

Acknowledgements

- A special thanks to Duy Pham at mysqltutorial.org for the permission to use the examples and database

- Really nice website with lots of examples and explanations.

Summary

- Stored Procedures
  - Creating
  - Calling
  - Fixing the thread_stack limit
  - Deleting
  - Parameters (IN / OUT / INOUT)
  - Altering Procedures
  - Variables
  - Flow Control
  - MySQL Cursor

MySQL

Stored Procedures

Dr Stewart Blakeway
Lecturer in Computer Science

COMP23111
Fundamentals of Databases