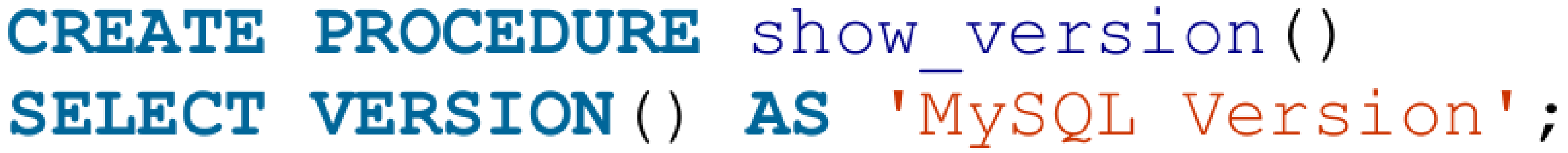
- Stored Procedures
- Triggers
- Events

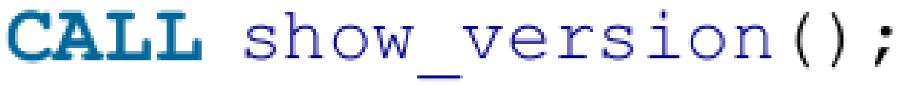
Stored functions and procedures

- Stored programs are database objects that are user-defined but stored on the server side for later execution.
- A stored function or procedure object encapsulates the code for performing an operation, enabling you to invoke the object easily by name rather than repeat all its code each time it’s needed.
- A stored function performs a calculation and returns a value that can be used in expressions just like a built-in function such as RAND(), NOW(), or LEFT().

Hello World for SP



```
CREATE PROCEDURE show_version()
SELECT VERSION() AS 'MySQL Version';
```



```
CALL show_version();
```

- A more complex one to follow
- It is a compound stmt using BEGIN ... END

```sql
CREATE PROCEDURE show_part_of_day()
BEGIN
  DECLARE cur_time, day_part TEXT;
  SET cur_time = CURTIME();
  IF cur_time < '12:00:00' THEN
    SET day_part = 'morning';
  ELSEIF cur_time = '12:00:00' THEN
    SET day_part = 'noon';
  ELSE
    SET day_part = 'afternoon or night';
  END IF;
  SELECT cur_time, day_part;
END;
```

Changing the delimiter

```
mysql> delimiter $$
mysql> CREATE FUNCTION avg_mail_size(user VARCHAR(8))
    -> RETURNS FLOAT READS SQL DATA
    -> BEGIN
    ->   DECLARE avg FLOAT;
    ->   IF user IS NULL
    ->   THEN # average message size over all users
    ->     SET avg = (SELECT AVG(size) FROM mail);
    ->   ELSE # average message size for given user
    ->     SET avg = (SELECT AVG(size) FROM mail WHERE srcuser =
user);
    ->   END IF;
    ->   RETURN avg;
    -> END;
    -> $$
Query OK, 0 rows affected (0.02 sec)
mysql> delimiter ;
```

Statewise Sales Tax

```
DROP PROCEDURE IF EXISTS name;
CREATE PROCEDURE name ...;
```

```
CREATE FUNCTION sales_tax_rate(state_code CHAR(2))
RETURNS DECIMAL(3,2) READS SQL DATA
BEGIN
  DECLARE rate DECIMAL(3,2);
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET rate = 0;
  SELECT tax_rate INTO rate FROM sales_tax_rate WHERE state =
state_code;
  RETURN rate;
END;
```

SP Characteristic-1

- READS SQL DATA means that the function reads data stored in databases, but does not modify any data. This happens if SELECTstatements are used, but there are no write operations are executed.
- CONTAINS SQL means that the function contains at least one SQL statement, but it does not read or write any data stored in a database. Examples include SET or DO.
- NO SQL means nothing, because MariaDB does not currently support any language other than SQL.
- MODIFIES SQL DATA means that the function contains statements that may modify data stored in databases. This happens if the function contains statements like DELETE, UPDATE, INSERT, REPLACE or DDL.

SP/F Characteristic-2

- DETERMINISTIC and NOT DETERMINISTIC apply only to functions.
- Specifying DETERMINISTC or NON-DETERMINISTIC in procedures has no effect.
- The default value is NOT DETERMINISTIC
- Functions are DETERMINISTIC when they always return the same value for the same input.
- For example, a truncate or substring function. Any function involving data, therefore, is always NOT DETERMINISTIC.

IN and OUT Param Spec

```
CREATE PROCEDURE mail_sender_stats(IN user VARCHAR(8),
                                   OUT messages INT,
                                   OUT total_size INT,
                                   OUT avg_size INT)
BEGIN
  # Use IFNULL() to return 0 for SUM() and AVG() in case there
are
  # no rows for the user (those functions return NULL in that
case).
  SELECT COUNT(*), IFNULL(SUM(size),0), IFNULL(AVG(size),0)
  INTO messages, total_size, avg_size
  FROM mail WHERE srcuser = user;
END;
```

```
mysql> CALL
mail_sender_stats('barb',@messages,@total_size,@avg_size);
```

Procedure 1

```
DELIMITER $$
CREATE PROCEDURE spGetIsAboveAverage(IN studentId INT, OUT isAboveAverage BOOLEAN)
BEGIN
    DECLARE avgMarks DECIMAL(5,2) DEFAULT 0;
    DECLARE studMarks INT DEFAULT 0;
    SELECT AVG(total_marks) INTO avgMarks FROM studentMarks;
    SELECT total_marks INTO studMarks FROM studentMarks WHERE stud_id = studentId;
    IF studMarks > avgMarks THEN
        SET isAboveAverage = TRUE;
    ELSE
        SET isAboveAverage = FALSE;
    END IF;
END$$
DELIMITER ;
```

Procedure 2 calling procedure 1

```
DELIMITER $$
CREATE PROCEDURE spGetStudentResult(IN studentId INT, OUT result VARCHAR(20))
BEGIN
      -- nested stored procedure call
    CALL spGetIsAboveAverage(studentId,@isAboveAverage);
    IF @isAboveAverage = 0 THEN
        SET result = "FAIL";
    ELSE
        SET result = "PASS";
    END IF;
END$$
DELIMITER ;
```

Test the procedures

```
CALL spGetStudentResult(2,@result);
SELECT @result;
```

What do you expect?

```
CALL spGetStudentResult(10,@result);
SELECT @result;
```

What do you expect?

PROCEDURE	FUNCTION
Supports different type of parameters like IN, OUT and INOUT.	Supports only input parameters.
They can call functions.	Functions cannot call procedures.
Exceptions can be handled in procedures.	No exception handling possible in FUNCTIONS.
Might or might not return a value.	A FUNCTION is expected to return a result always.
These cannot be called from within SELECT statements.	Functions can be called from within SELECT statement.
They are mainly used to process repeatable tasks.	FUNCTIONS are used to compute values and return results to the caller.
These are pre-compiled - i.e. they are compiled once and the compiled code is reused for subsequent calls being made to the procedure.	FUNCTIONS are compiled every time when they are called.

Exception Handling

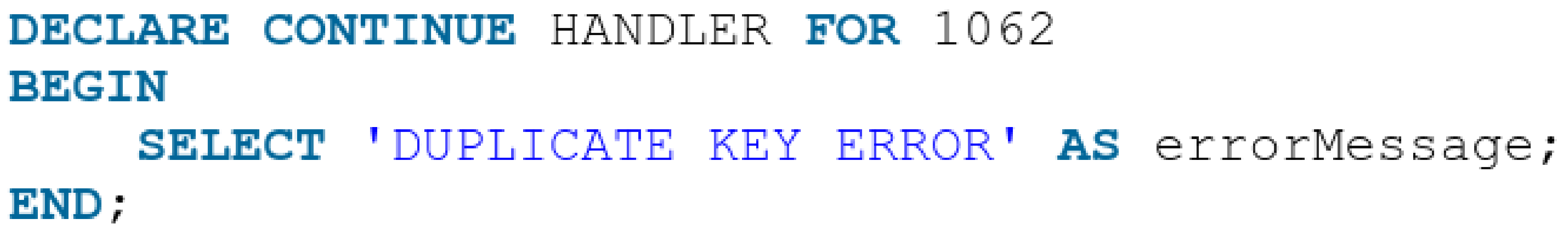
- DECLARE {action} HANDLER FOR {condition} {statement}
- {action} can have values
  - CONTINUE: This would still continue executing the current procedure.
  - EXIT: This procedure execution would halt and the flow would be terminated.
- {condition}: It’s the event that would cause the HANDLER to be invoked.
  - MySQL throw error code 1062 for a duplicate PRIMARY KEY violation

```
DELIMITER $$
CREATE PROCEDURE spInsertStudentData(IN studentId INT, IN total_marks INT,
IN grade VARCHAR(20),OUT rowCount INT)
BEGIN
 DECLARE EXIT HANDLER FOR 1062
   BEGIN
         SELECT 'DUPLICATE KEY ERROR' AS errorMessage;
   END;
 INSERT INTO studentMarks(stud_id, total_marks, grade)
VALUES(studentId,total_marks,grade);
   SELECT COUNT(*) FROM studentMarks INTO rowCount;
END$$
DELIMITER ;
```

Test EXIT / Define Continue

```
CALL
spInsertStudentData(1,450,'A+',@rowCount);

SELECT @rowCount
```



```
DECLARE CONTINUE HANDLER FOR 1062
BEGIN
    SELECT 'DUPLICATE KEY ERROR' AS errorMessage;
END;
```

```
DELIMITER $$

CREATE PROCEDURE my_proc_CASE

(INOUT no_employees INT, IN salary
INT)

BEGIN

CASE

WHEN (salary>10000)

THEN (SELECT COUNT(job_id) INTO
no_employees

FROM jobs

WHERE min_salary>10000);

WHEN (salary<10000)

THEN (SELECT COUNT(job_id) INTO
no_employees

FROM jobs

WHERE min_salary<10000);

ELSE (SELECT COUNT(job_id) INTO
no_employees

FROM jobs WHERE min_salary=10000);

END CASE;

END$$
```

- SHOW PROCEDURE STATUS WHERE name LIKE '%Student%'
- In MySQL, these are stored in a System table ‘informationschema.routines`
- SELECT * FROM information_schema.routines