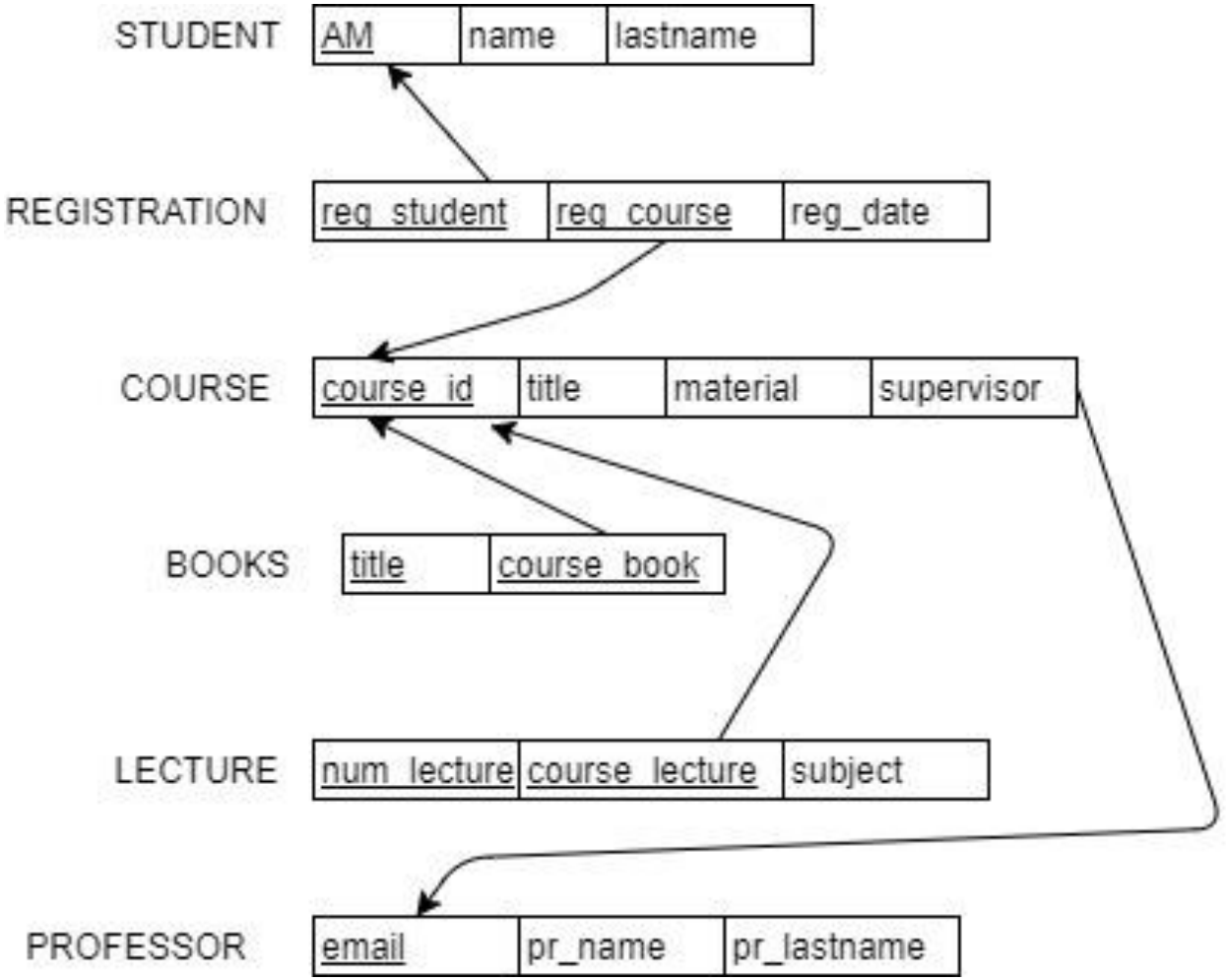
Data Bases LAB

Stored procedures

Schema


STUDENT
AM
name
lastname
REGISTRATION
reg_student
reg_course
reg_date
COURSE
course_id
title
material
supervisor
BOOKS
title
course_book
LECTURE
num_lecture
course_lecture
subject
PROFESSOR
email
pr_name
pr_lastname

Example : Create Tables

```
CREATE  TABLE student(
    name VARCHAR(25) DEFAULT 'unknown' NOT NULL,
    lastname VARCHAR(25) DEFAULT 'unknown' NOT NULL,
    AM INT(5) NOT NULL AUTO_INCREMENT,
    PRIMARY KEY(AM)
    );

CREATE  TABLE professor(
    pr_name VARCHAR(25) DEFAULT 'unknown' NOT NULL,
    pr_lastname VARCHAR(25) DEFAULT 'unknown' NOT NULL,
    email VARCHAR(255) NOT NULL,
    PRIMARY KEY(email)
    );
```

Example : Create Tables

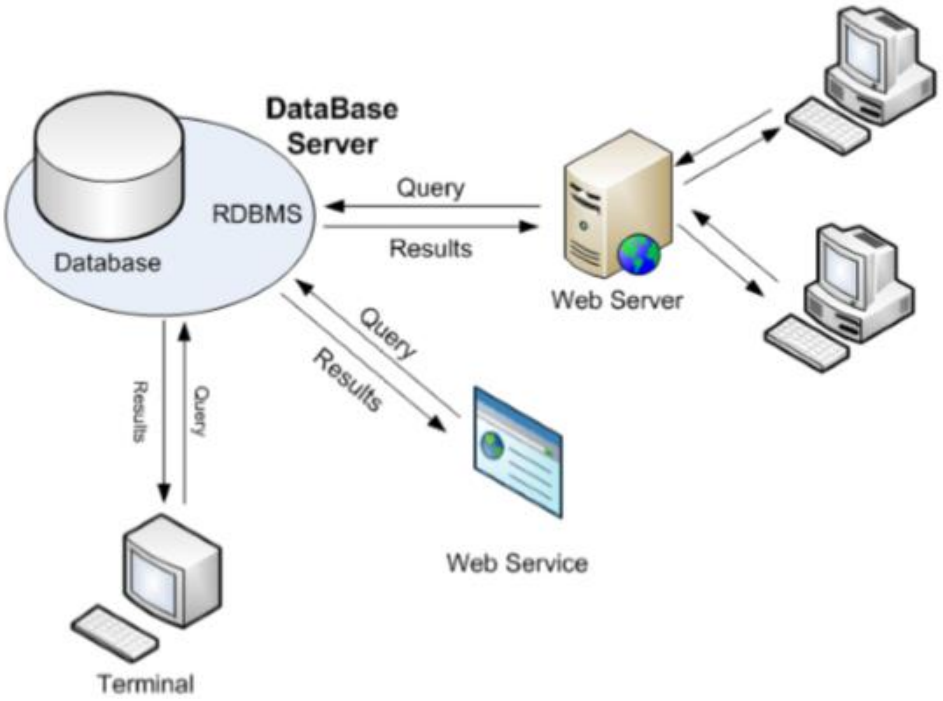
```
CREATE TABLE course (
    title VARCHAR(255) DEFAULT 'unknown' NOT NULL,
    material  TEXT,
    course_id  INT(4) NOT NULL  AUTO_INCREMENT,
    supervisor VARCHAR(255) NOT  NULL,
    PRIMARY KEY(course_id),
    UNIQUE(title),
    CONSTRAINT SUPERVISED
    FOREIGN KEY (supervisor) REFERENCES professor(email)
    ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE books (
    title VARCHAR(128) DEFAULT 'Title' NOT NULL,
    course_book INT(4) NOT NULL,
    PRIMARY KEY(title,course_book),
    CONSTRAINT CRSBOOK
    FOREIGN KEY (course_book) REFERENCES course(course_id)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

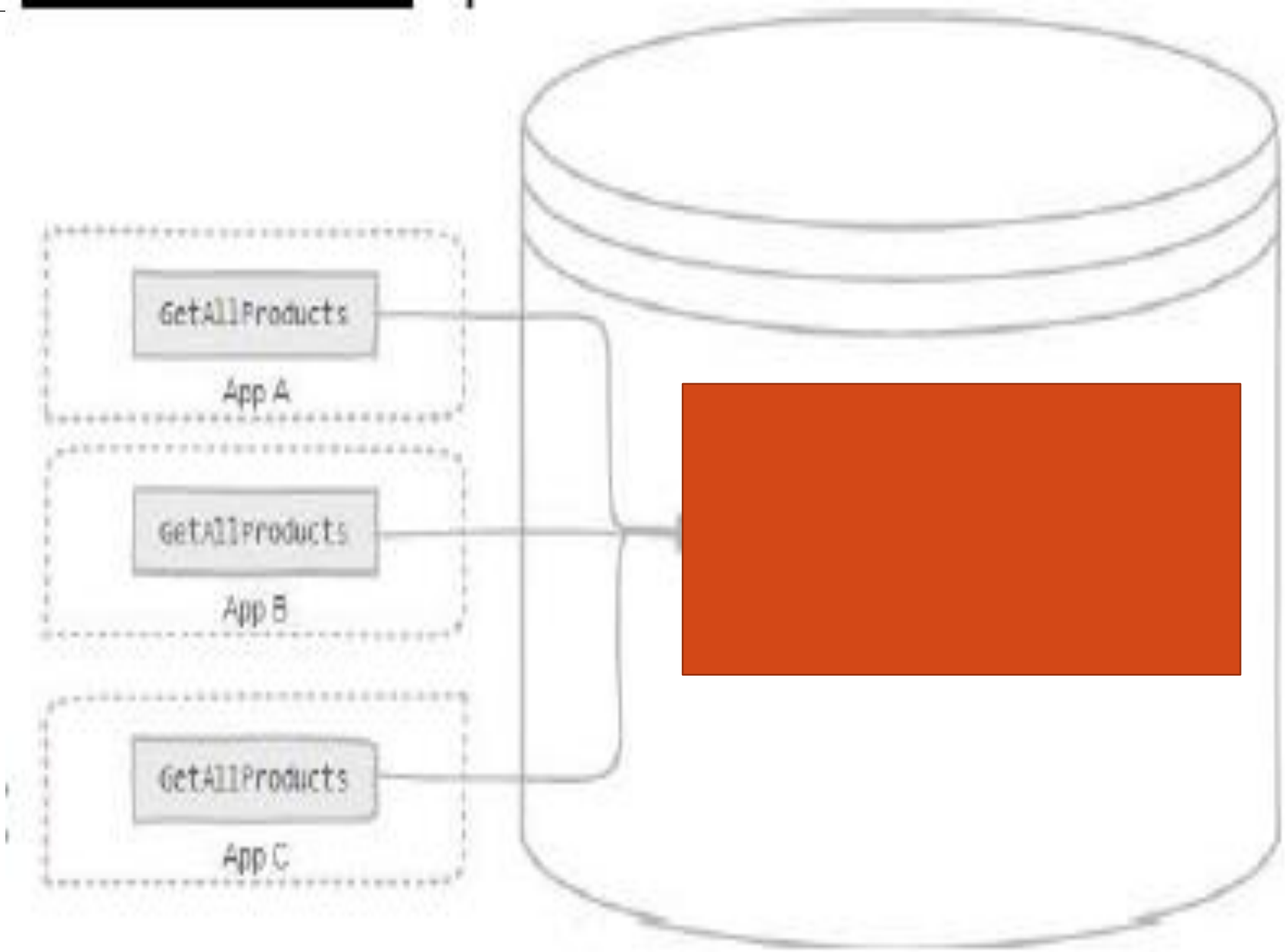
Example : Create Tables

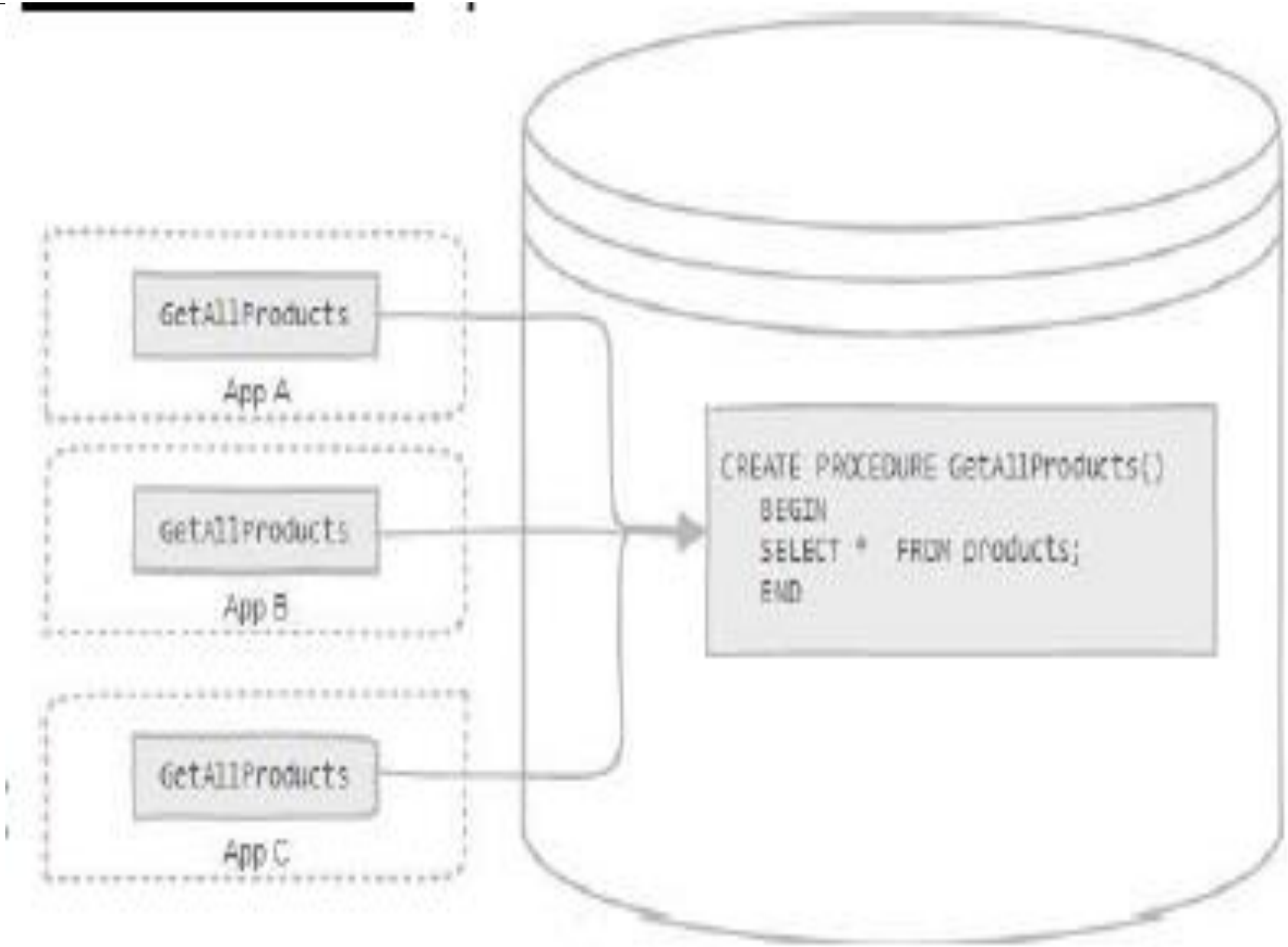
```
CREATE TABLE lecture (
    subject VARCHAR(128),
    num_lecture INT(2) NOT NULL,
    course_lecture INT(4) NOT NULL,
    PRIMARY KEY(num_lecture, course_lecture),
    CONSTRAINT CRSLECTURE
    FOREIGN KEY (course_lecture) REFERENCES course(course_id)
    ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE registration (
    reg_date DATE NOT NULL,
    reg_student INT(5) NOT NULL,
    reg_course INT(4) NOT NULL,
    PRIMARY KEY(reg_student,reg_course),
    CONSTRAINT CRSREGISTRATION
    FOREIGN KEY (reg_course) REFERENCES course(course_id)
    ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT STDNTREGISTRATION
    FOREIGN KEY (reg_student) REFERENCES student(AM)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

Data base and applications

- Client Server Model
- Data are stored in Data base Server. Clients send sql request to DBServer via network




GetAllProducts
App A
GetAllProducts
App B
GetAllProducts
App C


GetAllProducts
App A
GetAllProducts
App B
GetAllProducts
App C
CREATE PROCEDURE GetAllProducts()
BEGIN
SELECT * FROM products;
END

Stored procedures: basics

- A stored procedure is a procedure (a subroutine)
- It is created and stored in the database, i.e. on the server
- It can be ‘called’ and executed by clients or applications that use the database.
- It has access to database data.
- It is written in a language which depends on the RDBMS used
- The differences in syntax between the various RDBMSs are relatively minor.
- In the lab we use MySQL
- The language usually includes all sql commands and additional commands for writing small programs.
  - if-then-else blocks
  - while loops , …..

Why stored procedures ;

- Part of the logic is implemented on the server so
  - Code and Application platform are independent
  - Reduced latency due to network communication
- Deductions are set for basic base functions
  - Clients do not need to know many details about the design of the DB

Create , call, drop

- Create

  ```
  CREATE PROCEDURE <όνομα procedure>
  ```

- Call

  ```
  CALL <όνομα procedure>
  ```

- Drop

  ```
  DROP PROCEDURE <όνομα procedure>
  ```

- Show procedure code

  ```
  SHOW CREATE PROCEDURE <όνομα procedure>
  ```

- Show list of procedures

  ```
  SHOW PROCEDURE STATUS
  ```

Create and call- example

Procedure name hello_world

Operation Only one select

- Create stored procedure:

```
mysql>CREATE PROCEDURE hello_world()
    ->SELECT * FROM student;
```

- Call stored procedure:

```
mysql>CALL hello_world();
```

- Show procedure code:

```
mysql>SHOW CREATE PROCEDURE hello_world;
```

- Drop stored procedure:

```
mysql>DROP PROCEDURE hello_world();
```

name	lastname	AM
unknown	papad	1
al	georgiou	2
unknown	eleftheriou	3
Maria	papad	4
Maria	Baliou	5
mariana	stergiou	6

Create procedure

```
mysql> create procedure hello_world()
select 'hello world';
Query OK, 0 rows affected (0.00 sec)
```

Call procedure

```
call  hello_world;
+-------------+
| hello world |
+-------------+
| hello world |
+-------------+
1 row in set (0.00 sec)
```

Create stored procedure

```
mysql> create procedure hello_world()

select 'hello world';

Query OK, 0 rows affected (0.00 sec)
```

Show procedure code:

```
mysql> show create procedure hello_world;

+-------------+----------+-----------------------------------------------------
-----------------------------------+----------------------+------------------
--+------------------+

| Procedure   | sql_mode | Create Procedure
| character_set_client | collation_connectio                                                                  n |
Database Collation |

+-------------+----------+-----------------------------------------------------
-----------------------------------+----------------------+------------------
--+------------------+

| hello_world |          | CREATE DEFINER=`eleniv`@`150.140.141.181` PROCEDURE `
hello_world`()

select 'hello world' | latin1              | latin1_swedish_ci     | utf8_genera
l_ci    |

+-------------+----------+-----------------------------------------------------
-----------------------------------+----------------------+------------------
--+------------------+

1 row in set (0.00 sec)
```

```
mysql> create procedure hello_world()
    -> select * from user;
Query OK, 0 rows affected (0.01 sec)
```

```
call hello_world();
+----------+---------+----------+--------------+--------------------+------
---------------+
| username  | password | name      | surname       | reg_date            | email
|
+----------+---------+----------+--------------+--------------------+------
---------------+
| abrown    | w1lcoxon | Andrew    | McBrown       | 2018-01-27 16:02:56 | andre
wbr@yahoo.com   |
| bettyg    | jUn38@   | Betty     | Georgiou      | 2017-04-12 12:23:10 | georb
@softsol.gr     |
| cleogeo   | upL34r   | Cleomenis | Georgiadis    | 2018-02-13 12:23:34 | cleom
17@gmail.com    |
```

```
mysql> create procedure hello_world()
    -> select * from user;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> show create procedure hello_world;
+-------------+----------+-----------------------------------------------------------------------------------------+----------------------
-+----------------------+--------------------+
| Procedure   | sql_mode | Create Procedure                                                                        |
character_set_client | collation_connection | Database Collation |
+-------------+----------+-----------------------------------------------------------------------------------------+----------------------
-+----------------------+--------------------+
| hello_world |          | CREATE DEFINER=`eleni`@`150.140.141.181` PROCEDURE `hello_world`()
select * from user | latin1               | latin1_swedish_ci    | utf8_general_ci    |
+-------------+----------+-----------------------------------------------------------------------------------------+----------------------
-+----------------------+--------------------+
1 row in set (0.00 sec)
```

show procedure status;

```
+--------+--------------+-----------+----------------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+--------------------+
| Db     | Name         | Type      | Definer              | Modified            | Created             | Security_type | Comment | character_set_client | collation_connection | Database Collation |
+--------+--------------+-----------+----------------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+--------------------+
| eleniv | hello_world  | PROCEDURE | eleniv@150.140.141.181 | 2021-12-01 13:54:06 | 2021-12-01 13:54:06 | DEFINER       |         | latin1               | latin1_swedish_ci    | utf8_general_ci    |
| eleniv | hello_world_2 | PROCEDURE | eleniv@150.140.141.181 | 2021-12-01 14:01:16 | 2021-12-01 14:01:16 | DEFINER       |         | latin1               | latin1_swedish_ci    | utf8_general_ci    |
+--------+--------------+-----------+----------------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+--------------------+
2 rows in set (0.15 sec)
```

```
mysql> DROP PROCEDURE hello_world_2;
Query OK, 0 rows affected (0.00 sec)

mysql> show procedure status;
+--------+------------+-----------+----------------------+--------------------+--------------------+----
----------+---------+--------------------+--------------------+--------------------+
| Db     | Name       | Type      | Definer              | Modified           | Created            |
Security_type | Comment | character_set_client | collation_connection | Database
Collation |
+--------+------------+-----------+----------------------+--------------------+--------------------+----
----------+---------+--------------------+--------------------+--------------------+
| eleniv | hello_world | PROCEDURE | eleniv@150.140.141.181 | 2021-12-01
13:54:06 | 2021-12-01 13:54:06 | DEFINER       |         | latin1               |
latin1_swedish_ci    | utf8_general_ci    |
+--------+------------+-----------+----------------------+--------------------+--------------------+----
----------+---------+--------------------+--------------------+--------------------+
1 row in set (0.00 sec)

mysql>
```

blocks of statements

- Multiple statements in the procedure
  - Blocks of statements

    `BEGIN … block of statements… END`
- The delimiter of the block of statements should be different form the delimiter of the `CREATE PROCEDURE` statement
  - We have to change the delimiter using the statement : `DELIMITER`

blocks of statements - example

- mysql> DELIMITER $ (change delimiter)
- mysql>CREATE PROCEDURE hello_world2() (we declare the procedure)

```
        ->BEGIN
        ->  SELECT * FROM student ORDER BY am;
        ->  SELECT * FROM course ORDER BY course_id;
        ->  SELECT * FROM registration;
        ->END$
mysql> DELIMITER ;      (change delimiter again)
mysql> CALL hello_world2();   (call stored procedure)
```

Variables in mySQL

- In mysql we use session variables
  - Available in the current session, exist until the end of the session
  - Declaration not requiredn
  - their name starts with@
  - Assignment using SET

```
mysql> SET @x=4;
mysql> SET @y=7;
mysql> SET @z=@x-@y;
```

  - Print @z using SELECT

```
mysql>SELECT @z;
+------+
| @z   |
+------+
|   -3 |
+------+
1 row in set (0.00 sec)
```

What is the outpout;

mysql> select @k;

```
mysql> +------+
    -> | @k   |
    -> +------+
    -> | NULL |
    -> +------+
    -> 1 row in set (0.00 sec)
```

INPUT/ OUTPUT

- A stored procedure can have zero or more INPUT and OUTPUT parameters
- Each parameter is assigned a name, a data type, and direction like IN, OUT, or INOUT. If a direction is not specified, then by default, it is IN.
  - **IN – Input parameters**
    Their value is passed as input to the procedure. Any change of the value inside the procedure is not transferred to the environment..
  - **OUT – Output parameters**
    No value is provided to the procedure (assumed to be NULL). Any change to the value inside the procedure is available to the environment.
  - **INOUT – Input Output Parameters**
    Acombination of the two types above

Procedure (Input – Output) example 1/2

stored procedure afairesi

Subtract two numbers and return the result

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE afairesi (IN a INT, IN b INT, OUT result
INT)
    ->BEGIN
    ->  SET result=a-b;
    ->END$
Query OK, 0 rows affected (0.00 sec)
mysql>DELIMITER ;
mysql> CALL afairesi(5,4,@res);
```

Procedure (Input – Output) example 1/2

stored procedure afairesi

Subtract two numbers and return the result

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE afairesi (IN a INT, IN b INT, OUT result
INT)
    ->BEGIN
    ->  SET result=a-b;
    ->END$
Query OK, 0 rows affected (0.00 sec)
mysql>DELIMITER ;
mysql> CALL afairesi(5,4,@res);
mysql> SELECT @res;
+------+
| @res |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

Procedure (Input – Output) example 2/2

stored procedure : arnisi

Return the opposite of a number

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE arnisi(INOUT num INT)
    -> BEGIN
    ->   SET num=-num;
    -> END$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> SET @y=17;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL arnisi(@y);
Query OK, 0 rows affected (0.00 sec)
```

Procedure (Input – Output) example 2/2

stored procedure : arnisi

Return the opposite of a number

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE arnisi(INOUT num INT)
    -> BEGIN
    ->   SET num=-num;
    -> END$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> SET @y=17;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL arnisi(@y);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @y;
+------+
| @y   |
+------+
|  -17 |
+------+
1 row in set (0.00 sec)
```

Local Variables in procedures

- Local variables can be used in stored procedures to hold immediate results. These variables are local to the stored procedure ,
- The DECLARE statement initializes a variable by assigning it a name and a data type.

```
DECLARE id INT;
DECLARE name VARCHAR(20);
DECLARE birthday DATETIME;
```

- Assign value with SET
- Declare should be before statements that use the variable

Procedure swap();

Exchange the values of two variables

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE swap (INOUT name1 VARCHAR(20), INOUT name2
VARCHAR(20))
    -> BEGIN
    ->   DECLARE nametemp VARCHAR(20);
    ->   SET nametemp=name1;
    ->   SET name1=name2;
    ->   SET name2=nametemp;
    -> END$
Query OK, 0 rows affected (0.00 sec)


mysql>DELIMITER ;
mysql> SET @n1='thanos';
mysql> SET @n2='pantelis';
mysql> CALL swap(@n1,@n2);  (κλήση της procedure)
```

Procedure swap();

Exchange the values of two variables

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE swap (INOUT name1 VARCHAR(20), INOUT name2
VARCHAR(20))
    -> BEGIN
    ->   DECLARE nametemp VARCHAR(20);
    ->   SET nametemp=name1;
    ->   SET name1=name2;
    ->   SET name2=nametemp;
    -> END$
Query OK, 0 rows affected (0.00 sec)


mysql>DELIMITER ;
mysql> SET @n1='thanos';
mysql> SET @n2='pantelis';
mysql> CALL swap(@n1,@n2);  (κλήση της procedure)
mysql> SELECT @n1,@n2;
+----------+--------+
| @n1      | @n2    |
+----------+--------+
| pantelis | thanos |
+----------+--------+
1 row in set (0.00 sec)
```

Flow Control Statements

-
  - IF-THEN-ELSE
  - CASE
  - WHILE
  - REPEAT

IF-THEN-ELSE

```
IF condition
   THEN statement/s
ELSEIF condition
   THEN statement/s
ELSE statement/s
END IF;
```

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE absolute(IN num INT, OUT abs_num INT)
    ->BEGIN
    ->   IF(num<0) THEN
    ->      SET abs_num=-num;
    ->   ELSE
    ->      SET abs_num=num;
    ->   END IF;
    ->END$
mysql>DELIMITER ;
```

stored procedure : poinOfTime

it receives a date as parameter and prints if this date is past, present or future

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE pointOfTime (IN inputDay DATE)
->  BEGIN
->   DECLARE currentDay DATE;
->   SET currentDay=CURDATE();
->     IF (inputDay>currentDay) THEN
->       SELECT 'Future';
->     ELSEIF (inputDay=currentDay) THEN
->       SELECT 'Present';
->     ELSE
->       SELECT 'Past';
->     END IF;
->  END$
mysql>DELIMITER;

mysql>CALL pointOfTime('2011-12-31');     (call)
```

stored procedure : poinOfTime

it receives a date as parameter and prints if this date is past, present or future

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE pointOfTime (IN inputDay DATE)
->  BEGIN
->   DECLARE currentDay DATE;
->   SET currentDay=CURDATE();
->     IF (inputDay>currentDay) THEN
->       SELECT 'Future';
->     ELSEIF (inputDay=currentDay) THEN
->       SELECT 'Present';
->     ELSE
->       SELECT 'Past';
->     END IF;
->  END$
mysql>DELIMITER;

mysql>CALL pointOfTime('2011-12-31');
+------+
| past |
+------+
| past |
+------+
```

CASE

```
CASE μεταβλητή
   WHEN condition1 THEN statement/s
   WHEN condition2 THEN statement/s
   ...
ELSE statement/s
END CASE;
```

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE option (IN input_num INT)
->BEGIN
-> CASE (input_num)
->  WHEN 1 THEN
->   SELECT 'Option 1 selected';
->  WHEN 2 THEN
->   SELECT 'Option 2 selected';
->  ELSE
->   SELECT ‘Unknown option selected';
-> END CASE;
->END $
mysql>DELIMITER ;
```

WHILE

```
WHILE condition
   DO statement/s
END WHILE;
```

```
mysql> DELIMITER $
mysql>CREATE PROCEDURE simpleFor(IN maxNum INT)
      ->BEGIN
      -> DECLARE i INT;
      -> SET i=0;
      -> WHILE (i<maxNum AND maxNum>=0) DO
      ->   SELECT i;
      ->   SET i=i+1;
      -> END WHILE;
      ->END $
mysql>DELIMITER ;
```

```
mysql>CALL simpleFor(2);
+------+
| i    |
+------+
|    0 |
+------+
1 row in set (0.00 sec)
+------+
|  i   |
+------+
|    1 |
+------+
1 row in set (0.01 sec
```

REPEAT

```
REPEAT statement/s
   UNTIL condition
END REPEAT;
```

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE simpleForAlt(IN maxNum INT)
->BEGIN
-> DECLARE i INT;
-> SET i=0;
-> REPEAT
->  SELECT i;
->  SET i=i+1;
-> UNTIL (i>=maxNum OR maxNum<0)
-> END REPEAT;
->END$
mysql>DELIMITER;
```

```
mysql>CALL simpleForAlt(2);
+------+
| i    |
+------+
|    0 |
+------+
1 row in set (0.00 sec)
+------+
|  i   |
+------+
|    1 |
+------+
1 row in set (0.01 sec
```

Stored procedures handling results

- Stored procedures can have parameters for both passing values into the procedure and returning values from the call. Results can be returned as a result set, or as an OUT parameter cursor.
- We need to store and handle the results using variables
- Depending on the results there are two approaches:
  - **INTO**
    - When select results only one row
    - We store the values in local variables
  - **CURSORS**
    - When select results more than one rows (a table of results)
    - We access each row one by one

One row SELECT INTO

```
SELECT <λίστα πεδίων>
INTO <λίστα μεταβλητών>
FROM <λίστα πινάκων>
WHERE <συνθήκη>
;
```

- List of variables is related with list of fields (1-1)
- select should result **UP to one record (row),** otherwise ERROR!
- After the execution variables hold the values .
- If no result, the variables are NULL.

#1172 - Result consisted of more than one row

SELECT INTO – example

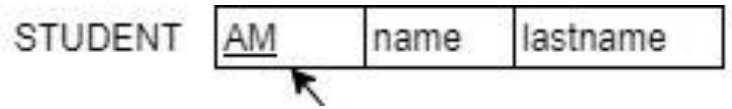
Show the info of the student with stAM

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE showStudentInfo(IN stAM INT)
->BEGIN
->
```

```
->END$
mysql>DELIMITER ;
```

SELECT INTO – example

Show the info of the student with stAM




```
mysql>DELIMITER $
mysql>CREATE PROCEDURE showStudentInfo(IN stAM INT)
->BEGIN
->

-> SELECT name, lastname
->
-> FROM student
-> WHERE am=stAM;









->END$
mysql>DELIMITER ;
```

SELECT INTO – example

Show the info of the student with stAM

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE showStudentInfo(IN stAM INT)
->BEGIN
-> DECLARE stName VARCHAR(25);
-> DECLARE stLastName VARCHAR(25
-> SELECT name, lastname
-> INTO stName, stLastName
-> FROM student
-> WHERE am=stAM;

->END$
mysql>DELIMITER ;
```

SELECT INTO – example

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE showStudentInfo(IN stAM INT)
->BEGIN
-> DECLARE stName VARCHAR(25);
-> DECLARE stLastName VARCHAR(25);
-> SELECT name, lastname
-> INTO stName, stLastName
-> FROM student
-> WHERE am=stAM;
-> IF(                                        ) THEN
->   SELECT 'Student not found.';
-> ELSEIF(                                                THEN
->   SELECT 'Partial info available:';
->
-> ELSE
->   SELECT 'Student found:';
->
-> END IF;
->END$
mysql>DELIMITER ;
```

SELECT INTO – example

```
CREATE  TABLE student(
    name VARCHAR(25) DEFAULT 'unknown' NOT NULL,
    lastname VARCHAR(25) DEFAULT 'unknown' NOT NULL,
    AM INT(5) NOT NULL AUTO_INCREMENT,
    PRIMARY KEY(AM)
    );
```

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE showStudentInfo(IN stAM INT)
->BEGIN
-> DECLARE stName VARCHAR(25);
-> DECLARE stLastName VARCHAR(25);
-> SELECT name, lastname
-> INTO stName, stLastName
-> FROM student
-> WHERE am=stAM;
-> IF(stName is NULL AND stLastName is NULL) THEN
->   SELECT 'Student not found.';
-> ELSEIF(stName LIKE '%unknown%' OR stLastName LIKE '%unknown%')THEN
->   SELECT 'Partial info available:';
->   SELECT stName, stLastName;
-> ELSE
->   SELECT 'Student found:';
->   SELECT stName, stLastName;
-> END IF;
->END$
mysql>DELIMITER ;
```

SELECT INTO – example

```
mysql>CALL showStudentInfo(2191);
+----------------+
| Student found: |
+----------------+
| Student found: |
+----------------+
1 row in set (0.01 sec)
+--------+------------+
| stName | stLastName |
+--------+------------+
| Βιβή   | Τζέκου     |
+--------+------------+
1 row in set (0.02 sec)
mysql>CALL showStudentInfo(2192);
+-------------------------+
| Partial info available: |
+-------------------------+
| Partial info available: |
+-------------------------+
1 row in set (0.00 sec)
+---------+------------+
| stName  | stLastName |
+---------+------------+
| unknown |  Ντούρου   |
+---------+------------+
1 row in set (0.01 sec)
mysql>CALL showStudentInfo(123);
+--------------------+
| Student not found. |
+--------------------+
| Student not found. |
+--------------------+
1 row in set (0.00 sec)
```

Multiple rows

- We use **Cursors**
  - **CUR**rent **S**et **O**f **R**ecord**S**
- We access one row of results (record) at a time
- we:
  1. Define a **cursor.**
  2. **Cursor is related to a select that returns the results**.
  3. Define what happens when results have finished**.**
  4. **Execute FETCH (Fetches each row from the response set until the end of results)**.

Multiple rows

- Declare a cursor and describe the select statement

```
DECLARE CURSOR <όνομα cursor> CURSOR FOR <εντολή select>;
```

- Declare the statement that is describes what will happen when we finish to access the results (usually set a flag =1)

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET <όνομα flag>=1;
```

- Open cursor: execute select related with this cursor

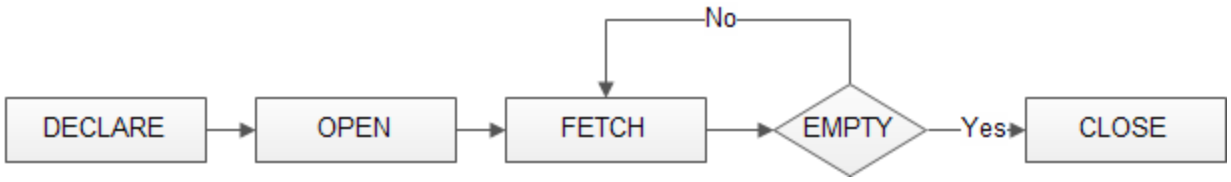
```
OPEN <όνομα cursor>;
```

- Read next row of results

```
FETCH <όνομα cursor> INTO <λίστα μεταβλητών>;
```

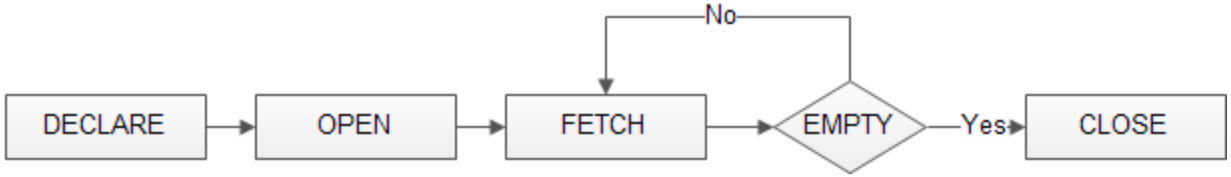
- Stop when flag =1 and close cursor

```
CLOSE <όνομαcursor>;
```



Example : create the showCourse Lectures Procedure

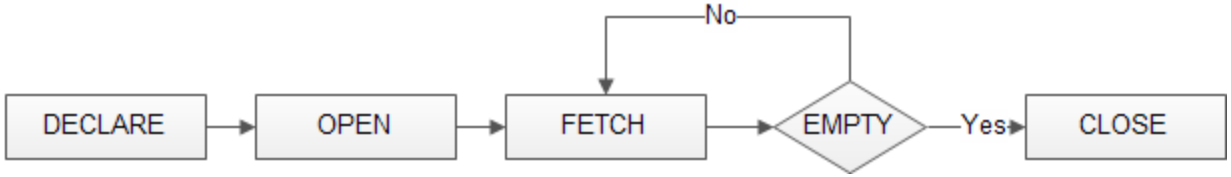
```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
->


->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;


->END$
mysql>DELIMITER ;
```



Example : create the showCourse Lectures Procedure

```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
->

->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
```

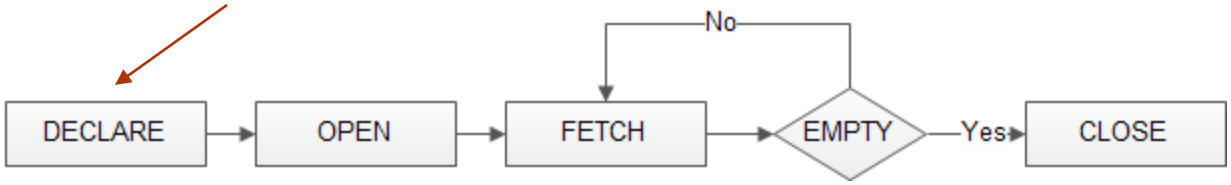
```
->END$
mysql>DELIMITER ;
```



Example : create the showCourse Lectures Procedure

```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
```

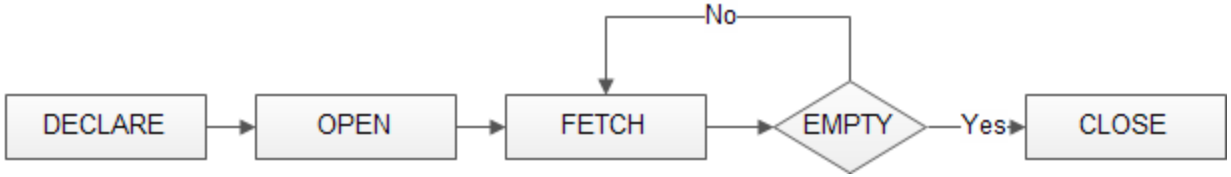
```
->END$
mysql>DELIMITER ;
```



Example : create the showCourse Lectures Procedure

```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1;
```

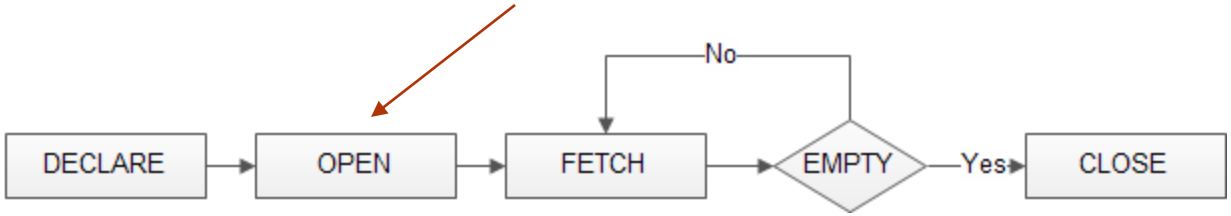
```
->END$
mysql>DELIMITER ;
```



Example : create the showCourse Lectures Procedure

```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1
-> OPEN lectCursor;
->
```

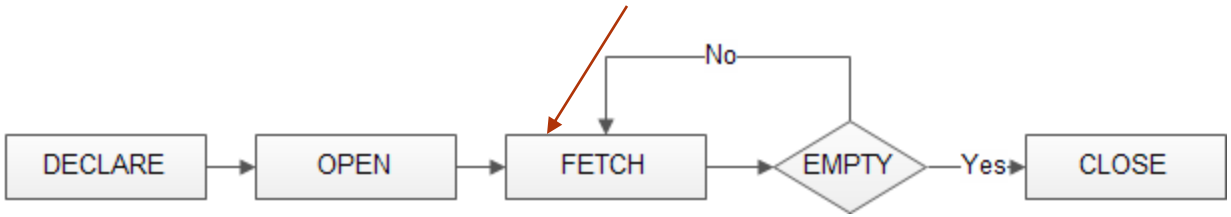
```
-> CLOSE lectCursor;
->END$
mysql>DELIMITER ;
```



Example : create the showCourse Lectures Procedure

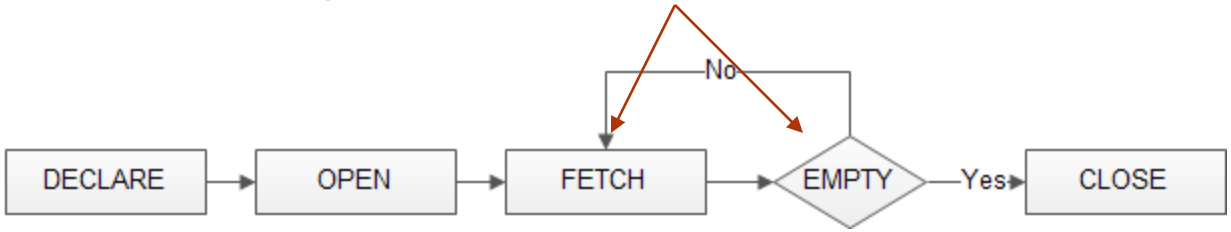
```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1
-> OPEN lectCursor;
-> SET finishedFlag=0;
-> FETCH lectCursor INTO lectSubject, lectNum;
->
```

```
->END$
mysql>DELIMITER ;
```



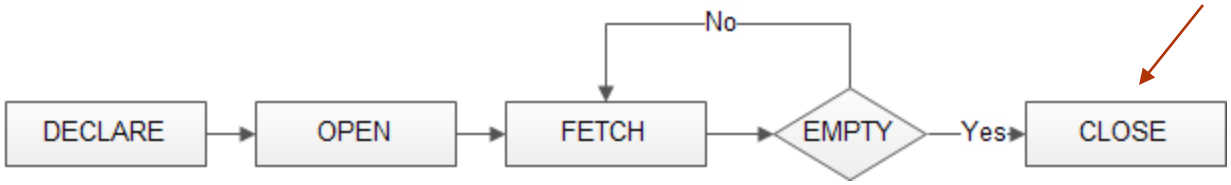
Example : create the showCourse Lectures Procedure

```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1
-> OPEN lectCursor;
-> SET finishedFlag=0;
-> FETCH lectCursor INTO lectSubject, lectNum;
-> WHILE (finishedFlag=0) DO
->   SELECT lectNum AS ‘Lecture Number’, lectSubject AS ‘Subject’;
->   FETCH lectCursor INTO lectSubject, lectNum;
-> END WHILE;
->
->END$
mysql>DELIMITER ;
```



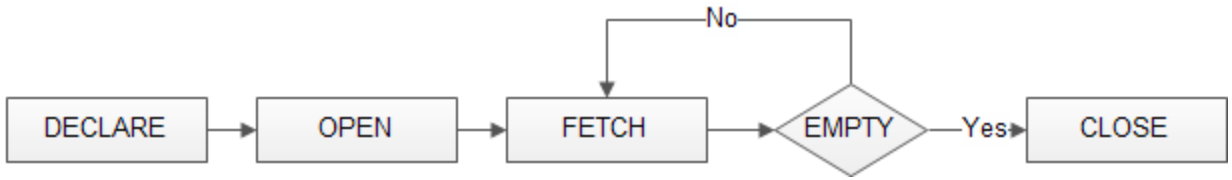
Example : create the showCourse Lectures Procedure

```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1;
-> OPEN lectCursor;
-> SET finishedFlag=0;
-> FETCH lectCursor INTO lectSubject, lectNum;
-> WHILE (finishedFlag=0) DO
->   SELECT lectNum AS 'Lecture Number', lectSubject AS 'Subject';
->   FETCH lectCursor INTO lectSubject, lectNum;
-> END WHILE;
-> CLOSE lectCursor;
->END$
mysql>DELIMITER ;
```



Example : create the showCourse Lectures Procedure

```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLecturesAlt$
mysql>CREATE PROCEDURE showCourseLecturesAlt(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->  SELECT subject, num_lecture FROM lecture WHERE course_lecture=courseId;
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1;
-> OPEN lectCursor;
-> SET finishedFlag=0;
-> REPEAT
->   FETCH lectCursor INTO lectSubject, lectNum;
->   IF (finishedFlag=0) THEN
->     SELECT lectNum AS 'Lecture Number', lectSubject AS 'Subject';
->   END IF;
->   UNTIL (finishedFlag=1)
-> END REPEAT;
-> CLOSE lectCursor;
->END$
mysql>DELIMITER ;
```



Results

```
mysql> CALL showCourseLecturesAlt(2);
+---------------------------------+---------------------+
| Αριθμός Διάλεξης                | Θέμα                |
+---------------------------------+---------------------+
|                               1 | Introduction to DBs |
+---------------------------------+---------------------+
1 row in set (0.00 sec)

+---------------------------------+----------------------+
| Αριθμός Διάλεξης                | Θέμα                 |
+---------------------------------+----------------------+
|                               2 | Requirements Analysis |
+---------------------------------+----------------------+
1 row in set (0.00 sec)

+---------------------------------+--------------------------+
| Αριθμός Διάλεξης                | Θέμα                     |
+---------------------------------+--------------------------+
|                               3 | ER and relational model |
+---------------------------------+--------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected, 0 warnings (0.00 sec)
```

```
mysql> select * from student;
+---------+-------------+----+
| name    | lastname    | AM |
+---------+-------------+----+
| unknown | papad       |  1 |
| al      | georgiou    |  2 |
| unknown | eleftheriou |  3 |
| Maria   | papad       |  4 |
| Maria   | Baliou      |  5 |
| mariana | stergiou    |  6 |
+---------+-------------+----+
6 rows in set (0.01 sec)
```

```
drop procedure if exists insertStudent;
DELIMITER $
CREATE PROCEDURE insertStudent(IN stAM INT)
BEGIN
 DECLARE stName VARCHAR(25);
 DECLARE stLastName VARCHAR(25);
 declare num Varchar(1);
 SELECT name, lastname
 INTO stName, stLastName
 FROM student
 WHERE am=stAM;
 insert into student values (stLastName,stName, null);
 END $
delimiter ;
CALL insertStudent(1);
```

```
mysql> select * from student;
+---------+------------+----+
| name    | lastname   | AM |
+---------+------------+----+
| unknown | papad      |  1 |
| al      | georgiou   |  2 |
| unknown | eleftheriou |  3 |
| Maria   | papad      |  4 |
| Maria   | Baliou     |  5 |
| mariana | stergiou   |  6 |
| papad   | unknown    |  8 |
+---------+------------+----+
7 rows in set (0.00 sec)
```

```
drop procedure if exists insertStudent2;
DELIMITER $
CREATE PROCEDURE insertStudent2(IN stAM INT)
BEGIN
 DECLARE stName VARCHAR(25);
 DECLARE stLastName VARCHAR(25);
 SELECT name, lastname
 INTO stName, stLastName
 FROM student
 WHERE am=stAM;
 insert into student values ( (concat(stLastName,'1'), concat(stName,'1'), null) ;
 END $
delimiter ;
```

Procedures in our database

```
select routine_name,
routine_type,definer,created,security_type,SQL_Data_A
ccess
from information_schema.routines
where routine_type='PROCEDURE' and
routine_schema= 'university';
```

Database name

```
+-----------------+-------------+---------------+---------------------+--------------+----------------+
| ROUTINE_NAME    | ROUTINE_TYPE | DEFINER       | CREATED             | SECURITY_TYPE | SQL_DATA_AC
+-----------------+-------------+---------------+---------------------+--------------+----------------+
| showStudentInfo3 | PROCEDURE   | root@localhost | 2022-11-29 18:19:47 | DEFINER      | CONTAINS SQL   |
+-----------------+-------------+---------------+---------------------+--------------+----------------+
1 row in set (0.00 sec)
```