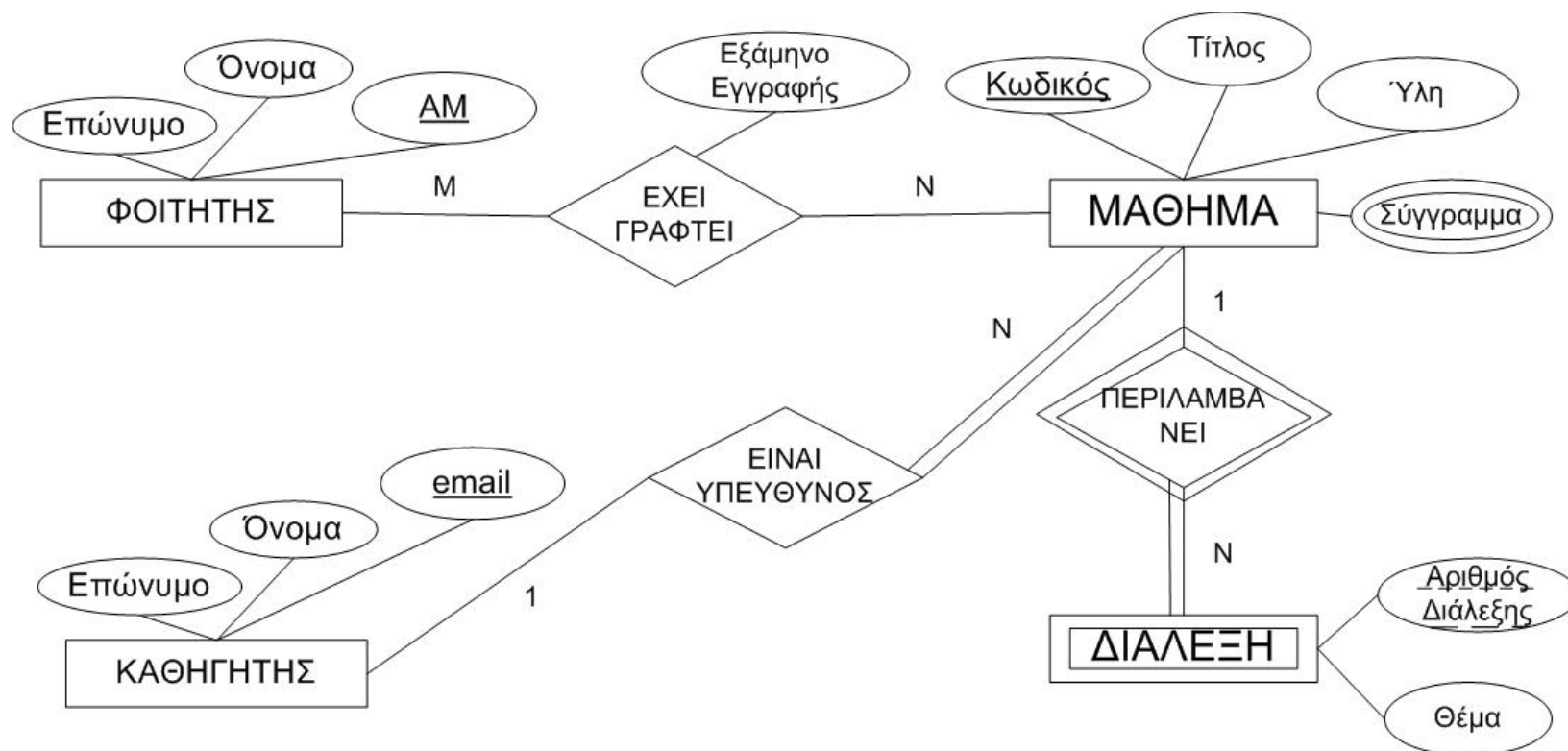


Εργαστήριο βάσεων δεδομένων

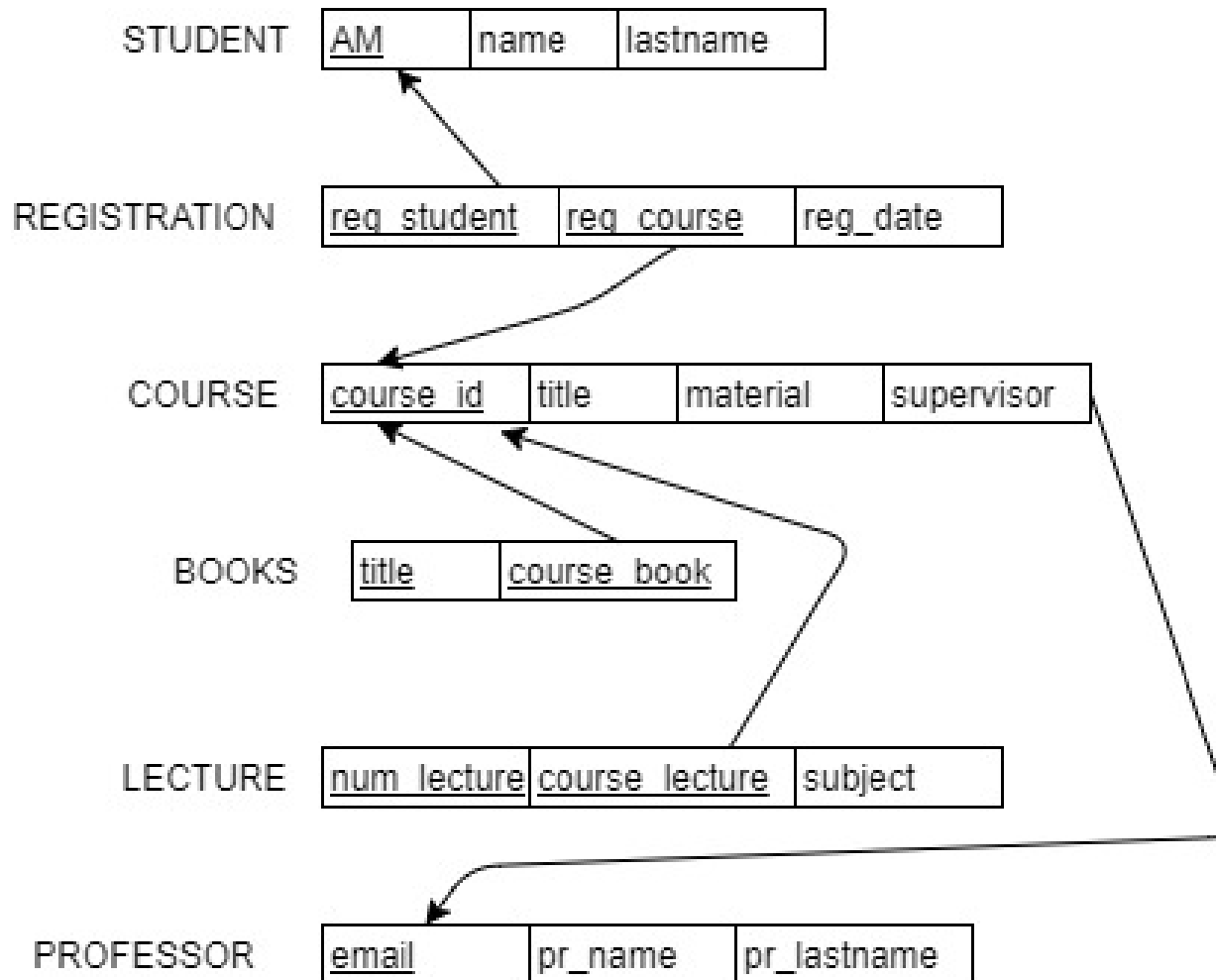
Stored procedures



Παράδειγμα - ER



Παράδειγμα-Σχεσιακό



Παράδειγμα – Δημιουργία Πινάκων

```
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)
```

```
);
```

Παράδειγμα – Δημιουργία Πινάκων

```
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);
```

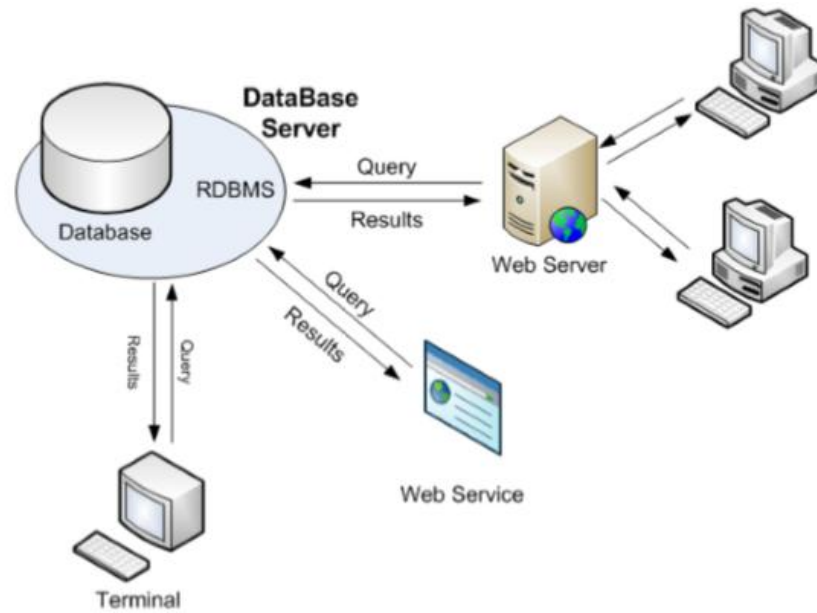
Παράδειγμα – Δημιουργία Πινάκων

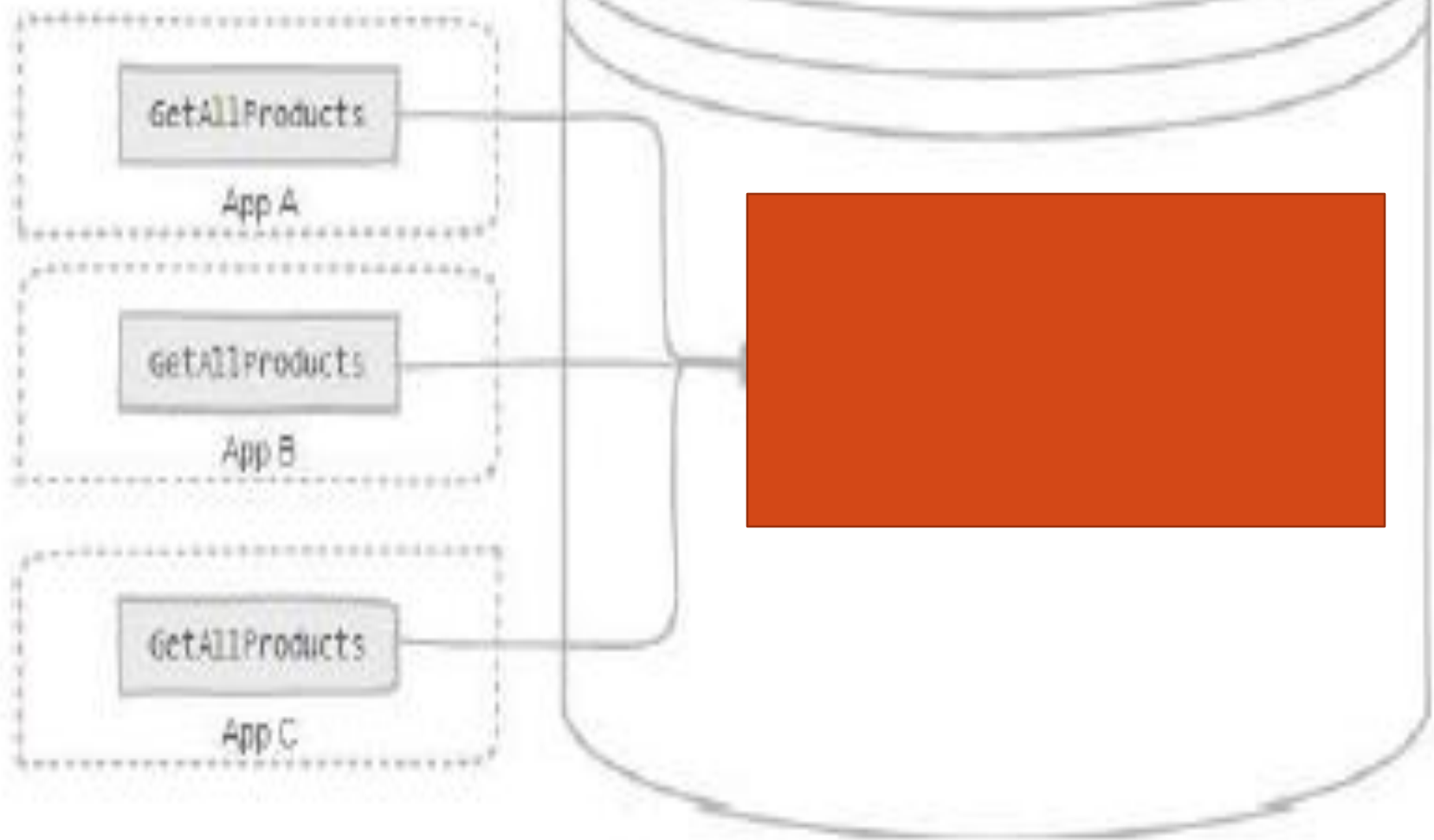
```
CREATE TABLE lecture (  
    subject VARCHAR(128),  
    num_lecture INT(2) NOT NULL,  
    course_lecture INT(4) NOT NULL,  
    PRIMARY KEY(num_lecture, course_lecture),  
    CONSTRAINT CRSLECTURE  
    FOREIGN KEY (course_lecture) REFERENCES course(course_id)  
    ON DELETE CASCADE ON UPDATE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

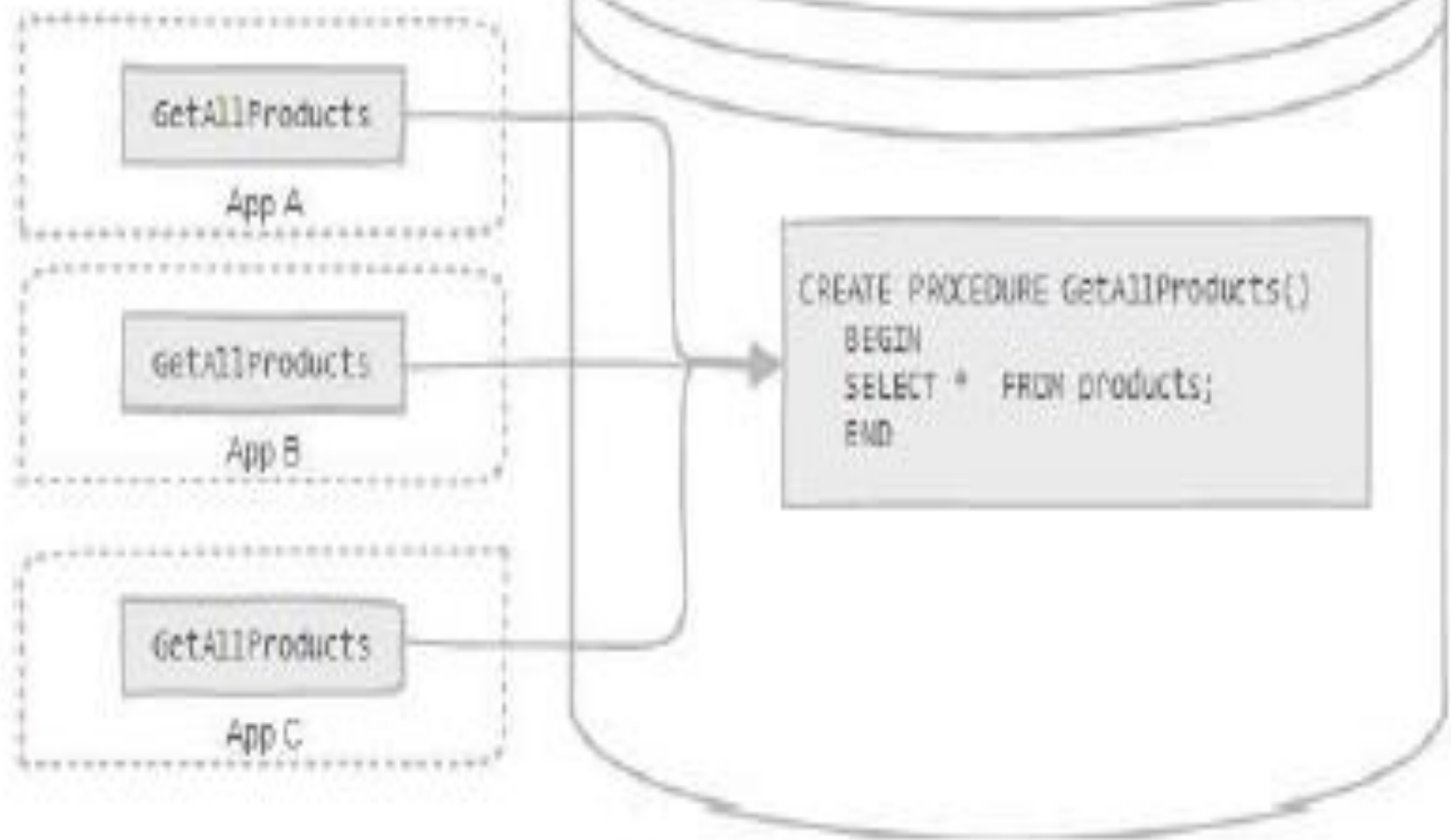
```
CREATE TABLE registration (  
    reg_date DATE NOT NULL,  
    reg_student INT(5) NOT NULL,  
    reg_course INT(4) NOT NULL,  
    PRIMARY KEY(reg_student, reg_course),  
    CONSTRAINT CRSREGISTRATION  
    FOREIGN KEY (reg_course) REFERENCES course(course_id)  
    ON DELETE CASCADE ON UPDATE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

Αρχιτεκτονική επικοινωνίας με τη ΒΔ

- Μια βάση χρησιμοποιείται μέσω του client-server μοντέλου
- Τα δεδομένα είναι αποθηκευμένα στον DBServer. Οι clients καλούν τον DBServer στέλνοντάς του sql εντολές για να πάρουν τα αποτελέσματα
- Η επικοινωνία γίνεται συνήθως μέσω δικτύου
- Υπάρχει επιβάρυνση για τη μεταφορά της sql εντολής προς τον DBServer και κυρίως των αποτελεσμάτων από τον DBServer.







Stored procedures: Βασικές έννοιες

- Μια stored procedure είναι μια υπορουτίνα, ένα πρόγραμμα
- Δημιουργείται και αποθηκεύεται στη βάση δεδομένων, δηλαδή στον server.
- Μπορεί να κληθεί και να εκτελεστεί από clients ή εφαρμογές που χρησιμοποιούν τη βάση.
- Έχει πρόσβαση στα δεδομένα της βάσης.
- Γράφεται σε μια ειδική γλώσσα η οποία εξαρτάται από το RDBMS που χρησιμοποιείται.
 - Οι διαφορές στη σύνταξη μεταξύ των διάφορων RDBMS είναι σχετικά μικρές.
 - Στο εργαστήριο χρησιμοποιούμε την MySQL
- Η γλώσσα συνήθως περιλαμβάνει όλες τις sql εντολές και επιπλέον εντολές για τη συγγραφή μικρών προγραμμάτων.
 - if-then-else blocks
 - while loops ,

Γιατί χρησιμοποιούνται;

- Μέρος της λογικής υλοποιείται στο server άρα
 - Κώδικας ανεξάρτητος από την πλατφόρμα των εφαρμογών
 - Μείωση της καθυστέρησης λόγω επικοινωνίας μέσω δικτύου
- Ορίζονται αφαιρέσεις για τις βασικές λειτουργίες της βάσης:
 - Οι clients δεν είναι απαραίτητο να γνωρίζουν πολλές λεπτομέρειες για το σχεδιασμό της ΒΔ
- Συνήθως χρησιμοποιούνται για:
 - Υλοποίηση λογικών ελέγχων ορθότητας δεδομένων πριν την εισαγωγή/επεξεργασία/διαγραφή
 - Έλεγχο δικαιωμάτων προσπέλασης
 - Ενοποίηση πολύπλοκων εργασιών που γίνονται συχνά και περιλαμβάνουν ακολουθίες πολλαπλών sql εντολών

Δημιουργία, κλήση και διαγραφή

- Δημιουργία

Εντολή `CREATE PROCEDURE <όνομα procedure>`

- Κλήση

Εντολή `CALL <όνομα procedure>`

- Διαγραφή

Εντολή `DROP PROCEDURE <όνομα procedure>`

- Εμφάνιση κώδικα

Εντολή `SHOW CREATE PROCEDURE <όνομα procedure>`

- Εμφάνιση λίστας procedures

Εντολή `SHOW PROCEDURE STATUS; 'H`

`SHOW PROCEDURE STATUS WHERE Db = database_name';`

Δημιουργία και κλήση - Παράδειγμα

Όνομα hello_world

Λειτουργία Εκτέλεση μιας select

- Δήλωση stored procedure:

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

- Κλήση stored procedure:

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

- Ανάκτηση του κώδικα:

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

- Διαγραφή 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

Δημιουργία

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

Κλήση

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

Δημιουργία

```
mysql> create procedure hello_world()
```

```
select 'hello world';
```

Query OK, 0 rows affected (0.00 sec)

Εμφάνιση κώδικα

```
mysql> show create procedure hello_world;
```

```
+-----+-----+-----+
-----+-----+
--+-----+
```

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

n |

```
+-----+-----+-----+
-----+-----+
--+-----+
```

```
| 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>

Δήλωση blocks κώδικα

- Στο σώμα των procedures είναι δυνατόν να περιλαμβάνονται πολλαπλές εντολές
 - Χρειάζεται η δυνατότητα ορισμού blocks εντολών
`BEGIN ...εντολές block... END`
- Ο χαρακτήρας τερματισμού των εσωτερικών εντολών πρέπει να είναι διαφορετικός από το χαρακτήρα τερματισμού της `CREATE PROCEDURE` εντολής
 - Αλλαγή του χαρακτήρα τερματισμού με την εντολή `DELIMITER`

Δήλωση blocks κώδικα - Παράδειγμα

- Δήλωση stored procedure με πολλαπλές εντολές

Όνομα hello_world2

Λειτουργία Εκτέλεση τριών select

- Αλλαγή χαρακτήρα τερματισμού εντολών από ; σε \$

```
mysql> DELIMITER $
```

- Ορισμός της procedure (Προσοχή: η CREATE PROCEDURE είναι μια εντολή, θα εκτελεστεί όταν δοθεί ο χαρακτήρας τερματισμού που έχει οριστεί)

```
mysql>CREATE PROCEDURE hello_world2() (δηλώνουμε την procedure)
```

```
->BEGIN
```

```
-> SELECT * FROM student ORDER BY am;
```

```
-> SELECT * FROM course ORDER BY course_id;
```

```
-> SELECT * FROM registration;
```

```
->END$
```

```
mysql> DELIMITER ;      (επαναφέρουμε τον delimiter)
```

```
mysql> CALL hello_world2();  (καλούμε την stored procedure)
```

Ορισμός μεταβλητών στο περιβάλλον της mysql

- Η mysql επιτρέπει τη δήλωση μεταβλητών

- Η εμβέλειά τους είναι το τρέχον session
- Το όνομά τους πρέπει να ξεκινάει με @
- Η ανάθεση τιμής γίνεται με την εντολή SET

```
mysql> SET @x=4;
```

```
mysql> SET @y=7;
```

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

- Η εκτύπωση της τιμής της @z γίνεται με την εντολή SELECT

```
mysql> SELECT @z;
```

```
+-----+
```

```
| @z |
```

```
+-----+
```

```
| -3 |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

Τι θα συμβεί ;

```
mysql> select @k;
```

```
mysql> +-----+
```

```
-> | @k |
```

```
-> +-----+
```

```
-> | NULL |
```

```
-> +-----+
```

```
-> 1 row in set (0.00 sec)
```

Είσοδος/Εξοδος σε procedures

- Οι stored procedures μπορούν να δεχτούν είσοδο και να δώσουν έξοδο στο περιβάλλον κατά την κλήση τους, μέσω παραμέτρων.
- Οι παράμετροι δηλώνονται σαν ορίσματα στην stored procedure. Ορίζονται τρία είδη παραμέτρων:
 - **IN - Παράμετροι εισόδου**
Η τιμή τους περνά σαν είσοδος στην procedure. Κάθε αλλαγή της τιμής στο εσωτερικό της procedure δεν μεταφέρεται στο περιβάλλον. Το default είδος παραμέτρων.
 - **OUT - Παράμετροι εξόδου**
Δεν παρέχεται τιμή στην procedure (θεωρείται NULL). Κάθε αλλαγή της τιμής στο εσωτερικό της procedure είναι διαθέσιμη στο περιβάλλον.
 - **INOUT - Παράμετροι εισόδου & εξόδου**
Συνδυάζει τα χαρακτηριστικά και των δύο τύπων

Είσοδος/Εξοδος σε procedures – παράδειγμα 1/2

Δήλωση stored procedure με

Όνομα `afairesi`

Λειτουργία Αφαίρεση δύο αριθμών και επιστροφή του αποτελέσματος

```
mysql> DELIMITER $
```

```
mysql> CREATE PROCEDURE afaresi (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 afaresi(5,4,@res);
```


Είσοδος/Εξοδος σε procedures – παράδειγμα 1/2

Δήλωση stored procedure με

Όνομα `afairesi`

Λειτουργία Αφαίρεση δύο αριθμών και επιστροφή του αποτελέσματος

```
mysql> DELIMITER $
```

```
mysql> CREATE PROCEDURE afaresi (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 afaresi(5,4,@res);
```

```
mysql> SELECT @res;
```

```
+-----+
```

```
| @res |
```

```
+-----+
```

```
|    1 |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

Είσοδος/Εξοδος σε procedures – παράδειγμα 2/2

Δήλωση stored procedure με Όνομα: arnisi και Λειτουργία: Επιστροφή της άρνησης ενός αριθμού

```
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)
```

Είσοδος/Εξοδος σε procedures – παράδειγμα 2/2

Δήλωση stored procedure με Όνομα: arnisi και Λειτουργία: Επιστροφή της άρνησης ενός αριθμού

```
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)
```

Τοπικές μεταβλητές procedures

- Επιτρέπεται δήλωση μεταβλητών μέσα στο σώμα της procedure
 - Η εμβέλειά τους είναι η stored procedure.
 - Ορίζονται με την εντολή DECLARE.
 - Στην declare ορίζεται και ο τύπος δεδομένων τους

```
DECLARE id INT;  
DECLARE name VARCHAR(20);  
DECLARE birthday DATETIME;
```
- Η ανάθεση τιμής γίνεται με χρήση της εντολής SET
- Η δήλωση μεταβλητών πρέπει να προηγείται των υπόλοιπων εντολών!

Procedure swap();

Λειτουργία: ανταλλάσσει τις τιμές δύο μεταβλητών

```
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();

Λειτουργία: ανταλλάσσει τις τιμές δύο μεταβλητών

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE swap (INOUT name1 VARCHAR(20), INOUT name2
VARCHAR(20))
-> BEGIN
->
->     SET          =name1;
->     SET name1=name2;
->     SET name2=          ;
-> 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)
```

Procedure swap();

Λειτουργία: ανταλλάσσει τις τιμές δύο μεταβλητών

```
mysql> DELIMITER $
mysql> CREATE PROCEDURE swap (INOUT name1 VARCHAR(20), INOUT name2
VARCHAR(20))
-> BEGIN
->
->   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)
```

Procedure swap();

Λειτουργία: ανταλλάσσει τις τιμές δύο μεταβλητών

```
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)
```


Δομές ελέγχου ροής

- Η mysql υποστηρίζει δομές ελέγχου ροής υπό συνθήκη. Οι κυριότερες είναι:
 - **IF-THEN-ELSE**
Άλματα υπό συνθήκη
 - **CASE**
Άλματα με βάση διακριτές τιμές μεταβλητής
 - **WHILE**
Επανάληψη υπό συνθήκη
 - **REPEAT**
Επανάληψη με τουλάχιστον μια εκτέλεση του block

IF-THEN-ELSE

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

Παράδειγμα - Procedure που επιστρέφει το απόλυτο ενός αριθμού

```
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 ;
```

Δήλωση stored procedure με Όνομα: poinOfTime

Δέχεται μια ημερομηνία και εκτυπώνει αν είναι παρόν, μέλλον ή παρελθόν

```
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');      (κλήση της procedure)
```

Δήλωση stored procedure με Όνομα: poinOfTime

Δέχεται μια ημερομηνία και εκτυπώνει αν είναι παρόν, μέλλον ή παρελθόν

```
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');      (κλήση της procedure)
```

```
+-----+
| past |
+-----+
| past |
+-----+
```

```
1 row in set (0.00 sec)
```

Δήλωση stored procedure με Όνομα: testexist Τί κάνει ?

```
DROP PROCEDURE testexist;
DELIMITER $
CREATE PROCEDURE testexist()
BEGIN
IF EXISTS (SELECT 1 FROM professor WHERE email LIKE 'EV%')
THEN
SELECT 'Βρέθηκε ο χρήστης με email που αρχίζει με EV' AS message;
ELSE
SELECT 'ΔΕΝ βρέθηκε χρήστης με email που αρχίζει με EV' AS
message;
END IF;
END$
DELIMITER ;
```

```
CALL testexist();
(κλήση της procedure)
```

*το SELECT 1 ζητάει από τη βάση δεδομένων να επιστρέψει τον αριθμό 1 ως σταθερή τιμή για κάθε εγγραφή (γραμμή) που ταιριάζει στο ερώτημά σου, αντί να επιστρέψει πραγματικά δεδομένα από κάποια στήλη. Εδώ χρησιμοποιείται μαζί με το EXISTS για να ελέγξουμε αν υπάρχει **έστω και μία** εγγραφή που να ικανοποιεί το κριτήριο μας.*

Δήλωση stored procedure με Όνομα: **testexist**

*Αν υπάρχει εγγραφή που το όνομα του καθηγητή αρχίζει από EV εμφανίζει
σχετικό μήνυμα*

```
DROP PROCEDURE testexist;  
DELIMITER $  
  CREATE PROCEDURE testexist()  
BEGIN  
IF EXISTS (SELECT 1 FROM professor WHERE email LIKE 'EV%')  
THEN  
  SELECT 'Βρέθηκε ο χρήστης με email που αρχίζει με EV' AS message;  
ELSE  
  SELECT 'ΔΕΝ βρέθηκε χρήστης με email που αρχίζει με EV' AS  
message;  
END IF;  
END$  
DELIMITER ;
```

```
CALL testexist();  
(κλήση της procedure)
```

*το SELECT 1 ζητάει από τη βάση δεδομένων να επιστρέψει τον αριθμό 1 ως σταθερή τιμή για κάθε εγγραφή (γραμμή) που ταιριάζει στο ερώτημά σου, αντί να επιστρέψει πραγματικά δεδομένα από κάποια στήλη. Εδώ χρησιμοποιείται μαζί με το EXISTS για να ελέγξουμε αν υπάρχει **έστω και μία** εγγραφή που να ικανοποιεί το κριτήριο μας.*

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)
```

Διαχείριση δεδομένων της ΒΔ

- Η βασικότερη λειτουργία των stored procedures είναι η διαχείριση των δεδομένων της βάσης.
- Χρειαζόμαστε τρόπους αποθήκευσης των αποτελεσμάτων των select σε μεταβλητές, για να τα διαχειριστούμε προγραμματιστικά.
- Παρέχονται δύο μηχανισμοί ανάλογα με τον αριθμό των αποτελεσμάτων που επιστρέφονται:
 - **INTO**
 - Χρησιμοποιείται όταν το select επιστρέφει **μία εγγραφή**
 - Αποθηκεύουμε σε μεταβλητές τις τιμές που επιστρέφονται
 - **CURSORS**
 - Χρησιμοποιούνται όταν το select επιστρέφει **πολλαπλές εγγραφές** (δηλ. έναν πίνακα αποτελεσμάτων)
 - Προσπελάζουμε τις γραμμές των αποτελεσμάτων μία-μία

Διαχείριση μεμονωμένων τιμών- SELECT INTO

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

- Η λίστα μεταβλητών πρέπει να αντιστοιχίζεται 1-1 με τη λίστα πεδίων.
- Πρέπει να εξασφαλίζουμε ότι η select θα επιστρέφει το πολύ μια εγγραφή, αλλιώς προκαλείται σφάλμα!
- Μετά την εκτέλεση, οι μεταβλητές περιέχουν τις τιμές που έχουν επιστραφεί.
- Αν δεν επιστραφεί τίποτα, οι μεταβλητές περιέχουν NULL.

#1172 - Result
consisted of more
than one row

SELECT INTO – Παράδειγμα

Show the info of the student with stAM

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

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

SELECT INTO – Παράδειγμα

Show the info of the student with stAM

STUDENT	AM	name	lastname
---------	----	------	----------



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

->END$
mysql>DELIMITER ;
```

SELECT INTO – Παράδειγμα

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 – Παράδειγμα

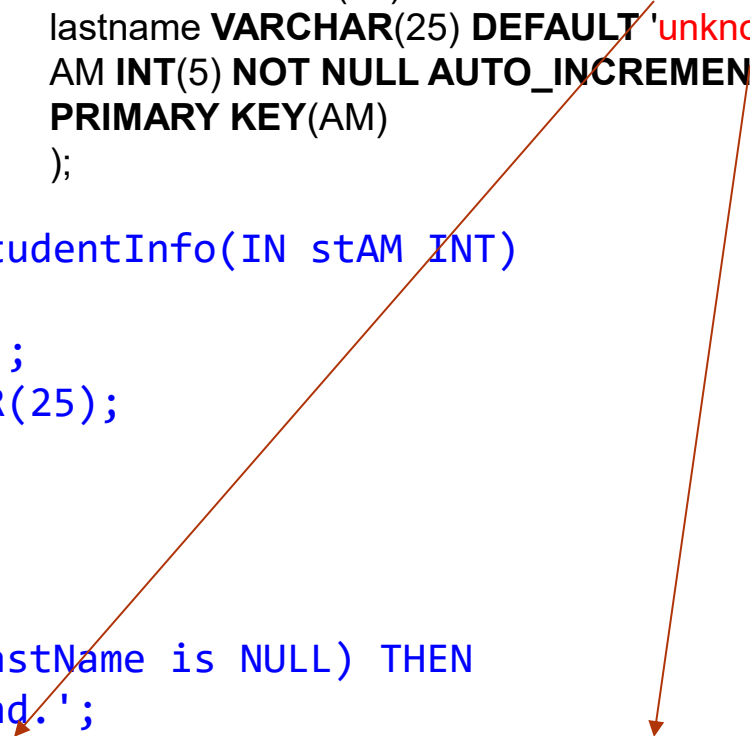
```
mysql>DELIMITER $
mysql>CREATE PROCEDURE showStudentInfo(IN stAM INT)
->BEGIN
-> DECLARE stName VARCHAR(25);
-> DECLARE stLastName VARCHAR(25);
-> SELECT name, lastname
-> INTO stName, stLastName
-> FROM student
-> WHERE am=stAM;
-> IF(                                     ) THEN
->   SELECT 'Student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

SELECT INTO

- Παράδειγμα

```
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 – Παράδειγμα

```
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)
```

Διαχείριση πινάκων αποτελεσμάτων

- Όταν η select επιστρέφει πολλαπλές εγγραφές, χρησιμοποιούμε **Cursors**
 - Συντομογραφία του **CURrent Set Of Records**
- Προσπελάνουμε μία εγγραφή των αποτελεσμάτων κάθε φορά
- Διαδικασία:
 1. Ορίζουμε μια μεταβλητή τύπου **cursor**.
 2. Την αντιστοιχίζουμε στη select που θα επιστρέψει τα αποτελέσματα.
 3. Ορίζουμε μια εντολή που θα εκτελεστεί όταν έχουν διαβαστεί όλα τα αποτελέσματα.
 4. Εκτελούμε την εντολή FETCH που μεταφέρει τον cursor στην επόμενη γραμμή, μέχρι να εκτελεστεί η εντολή τερματισμού.

Ο cursor (δείκτης) είναι ένας μηχανισμός που χρησιμοποιείται για την επεξεργασία ενός συνόλου δεδομένων (result set) γραμμή προς γραμμή. Οι δείκτες είναι ιδιαίτερα χρήσιμοι όταν χρειάζεται να εκτελεστεί κάποια λειτουργία ή λογική για κάθε γραμμή ενός πίνακα ή ενός αποτελέσματος ερωτήματος, κάτι που δεν μπορεί εύκολα να επιτευχθεί με κανονικά ερωτήματα SQL.

Διαχείριση πινάκων αποτελεσμάτων-Εντολές

- Ορίζουμε ένα cursor και δηλώνουμε τη select στην οποία θα εφαρμοστεί

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

- Δηλώνουμε την εντολή που συνδέεται με το τέλος του διαβάσματος των αποτελεσμάτων (συνήθως θέτουμε ένα flag ίσο με 1)

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

- Ανοίγουμε τον cursor, δηλαδή εκτελούμε το select που συνδέεται με αυτόν

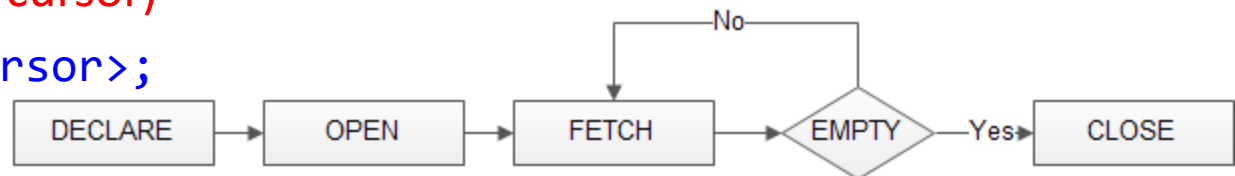
OPEN <όνομα cursor>;

- Διαβάζουμε την επόμενη γραμμή στα αποτελέσματα

FETCH <όνομα cursor> INTO <λίστα μεταβλητών>;

- Σταματάμε να διαβάζουμε όταν το flag που ορίσαμε γίνει 1 (και κλείνουμε τον cursor)

CLOSE <όνομα cursor>;



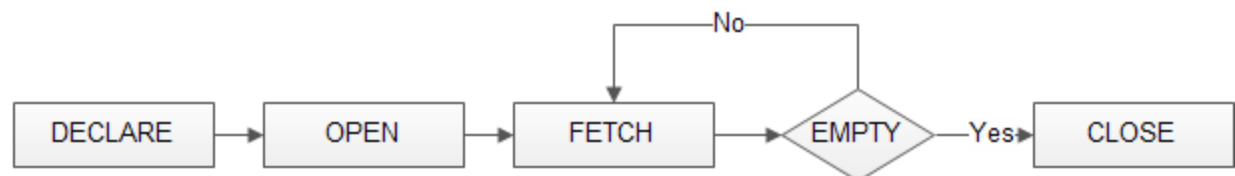
Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

```
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 ;
```

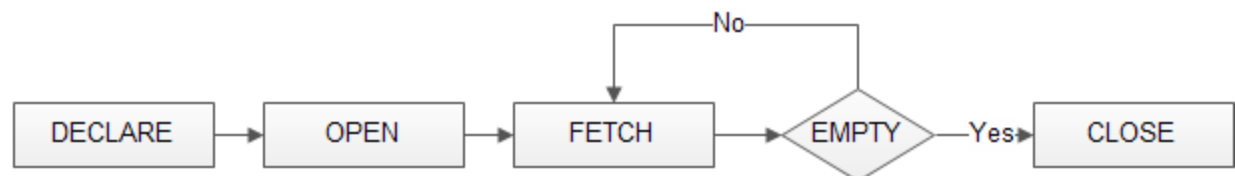


Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

```
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 ;
```

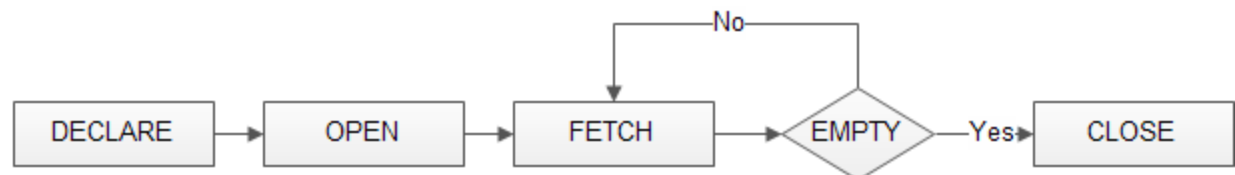


Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

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

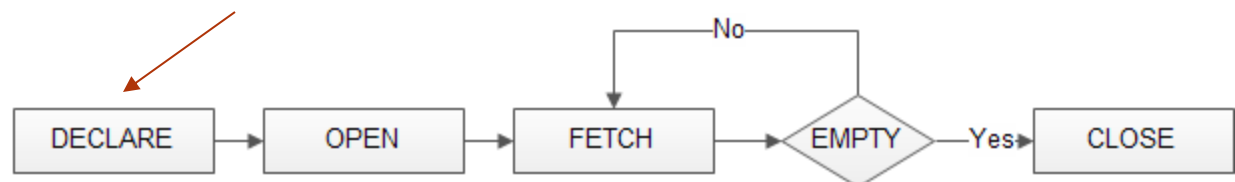


Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

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

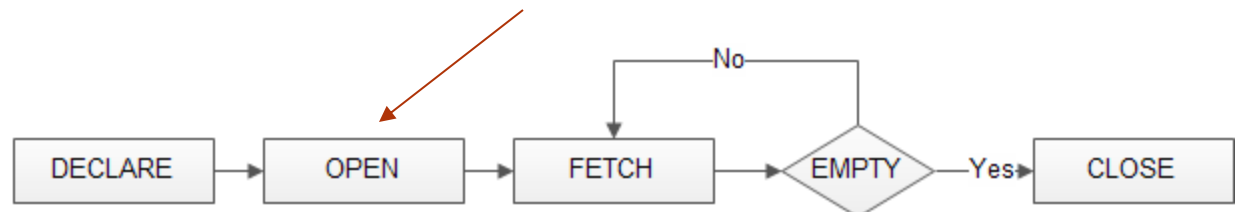


Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

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

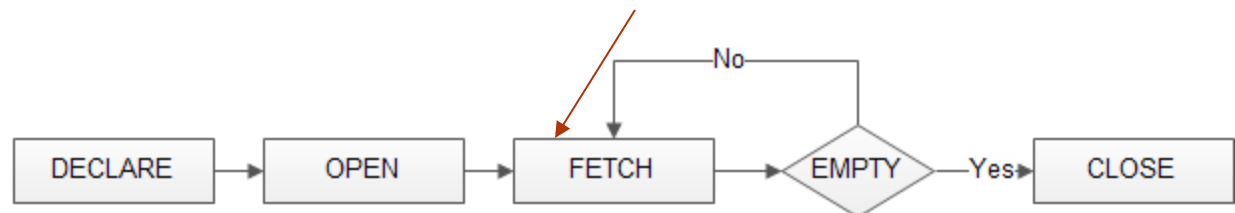
```
-> CLOSE lectCursor;
->END$
mysql>DELIMITER ;
```



Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

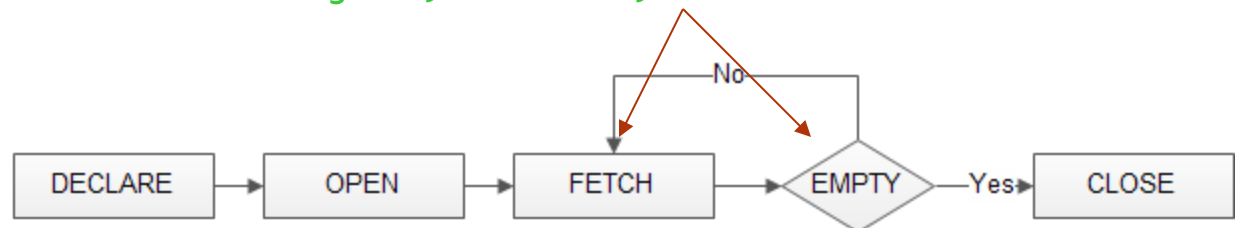
```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
->
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM lecture WHERE course_lecture=courseId;
-> DECLARE CONTINUE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



Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

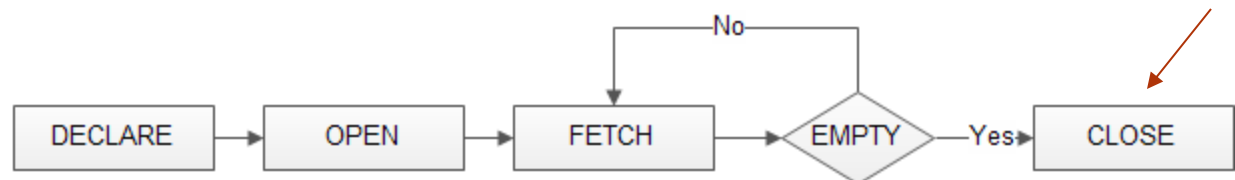
```
mysql>DELIMITER $
mysql>DROP PROCEDURE IF EXISTS showCourseLectures$
mysql>CREATE PROCEDURE showCourseLectures(IN courseId INT)
->BEGIN
-> DECLARE lectSubject VARCHAR(128);
-> DECLARE lectNum INT(2);
-> DECLARE finishedFlag INT;
-> DECLARE lectCursor CURSOR FOR
->   SELECT subject,num_lecture FROM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



Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

```
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 ;
```



- CALL showCourseLectures(2);

```
+-----+-----+
| Lecture Number | Subject |
+-----+-----+
|          1 | create |
+-----+-----+
1 row in set (0.001 sec)
```

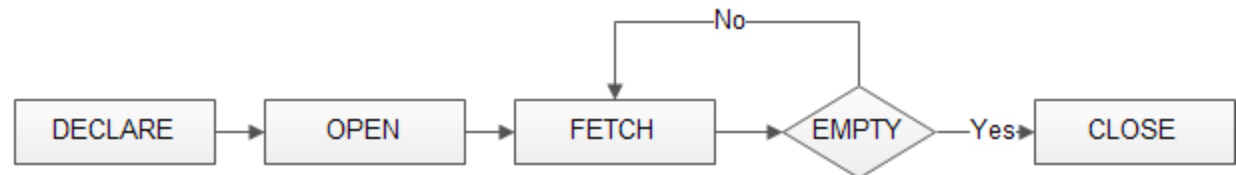
```
+-----+-----+
| Lecture Number | Subject |
+-----+-----+
|          2 | insert |
+-----+-----+
1 row in set (0.005 sec)
```

```
+-----+-----+
| Lecture Number | Subject |
+-----+-----+
|          3 | select |
+-----+-----+
1 row in set (0.008 sec)
```

Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

```
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 ;
```



Διαχείριση πινάκων

αποτελεσμάτων-Παράδειγμα 1

```
mysql> CALL showCourseLecturesAlt(2);
```

```
+-----+-----+  
| Lecture Number | Subject |  
+-----+-----+  
|           1 | create |  
+-----+-----+  
1 row in set (0.001 sec)
```

```
+-----+-----+  
| Lecture Number | Subject |  
+-----+-----+  
|           2 | insert |  
+-----+-----+  
1 row in set (0.005 sec)
```

```
+-----+-----+  
| Lecture Number | Subject |  
+-----+-----+  
|           3 | select |  
+-----+-----+  
1 row in set (0.008 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);

    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_ACCESS
showStudentInfo3	PROCEDURE	root@localhost	2022-11-29 18:19:47	DEFINER	CONTAINS SQL

1 row in set (0.00 sec)

Procedures in our database

```
select routine_name from information_schema.routines  
where routine_schema= 'university';
```

+-----+

Database name

ROUTINE_NAME

+-----+

afairesi
arnisi
hello_world
hello_world2
insertStuden2t
insertStuden3
insertStudent
insertStudent2
option_PROC
pointOfTime
pointOfTime1
shoStudentInfo
showCourseLectures
showCourseLecturesAlt

Προσοχή

- Για να γράψετε μια stored procedure πρέπει να κατανοείτε τι ακριβώς κάνει.
- Συχνά η δυσκολία είναι στο να κατανοήσετε και να υλοποιήσετε τα sql queries που σας δίνουν το αποτέλεσμα που θέλετε.
- Δουλέψτε αρκετά σε select
 - Γράψτε select πάνω στη βάση με hardcoded values
 - ελέγξτε ότι φέρνουν το αποτέλεσμα που επιθυμείτε
 - αντικαταστήστε τις hardcoded values με παραμέτρουςΚαι ολοκληρώστε την procedure

Προσοχή

- Δουλέψτε αρκετά σε select
 - Γράψτε select πάνω στη βάση με hardcoded values
 - ελέγξτε ότι φέρνουν το αποτέλεσμα που επιθυμείτε
 - αντικαταστήστε τις hardcoded values με παραμέτρους και ολοκληρώστε την procedure (το οποίο απαιτεί δουλειά

```
SELECT STUDENT.NAME  
FROM `STUDENT` WHERE  
student.AM >'1';
```



```
SET @STam = '1';  
SELECT STUDENT.NAME  
FROM `STUDENT`  
WHERE student.AM >@sTAM;
```

Προσοχή

- Δουλέψτε αρκετά σε select
 - Γράψτε select πάνω στη βάση με hardcoded values
 - ελέγξτε ότι φέρνουν το αποτέλεσμα που επιθυμείτε
 - αντικαταστήστε τις hardcoded values με παραμέτρους και ολοκληρώστε την procedure (το οποίο απαιτεί δουλειά

```
SELECT STUDENT.NAME  
FROM `STUDENT` WHERE  
student.AM >'1';
```



```
SET @STam = '1';  
SELECT STUDENT.NAME  
FROM `STUDENT`  
WHERE student.AM >@sTAM;
```



```
DELIMITER $  
CREATE PROCEDURE checkName(IN  
sTAM int, OUT stName varchar (25))  
BEGIN  
SELECT STUDENT.NAME into stName  
FROM `STUDENT` WHERE student.AM  
>sTAM;  
end$
```



**#1172 - Result consisted of more
than one row**

?!

Επόμενα βήματα

- Δευτέρα 8/12/25 Διάλεξη Triggers 9:00-11:00 στη Δ2
- Δευτέρα 8/12/25 Εξέταση Stored (ΥΚ)
 - Ομάδα 1 εξέταση 11:10 - 12:40
 - Ομάδα 2 εξέταση 12:40 - 14:10
 - Ομάδα 3 εξέταση 14:10 – 15:40
- Παρασκευή 12/12 υποβολή αναφοράς Triggers στο eclass
- Δευτέρα 15/12 Εξέταση Triggers (ΥΚ)

ΠΡΟΣΟΧΗ Για την παράδοση άσκησης και εξέταση Triggers ΧΡΕΙΑΖΕΤΑΙ να φτιάξετε τη βάση και να εισαγάγετε εγγραφές στους πίνακες. (ΙΔΙΑ βάση με τη βάση του Project)