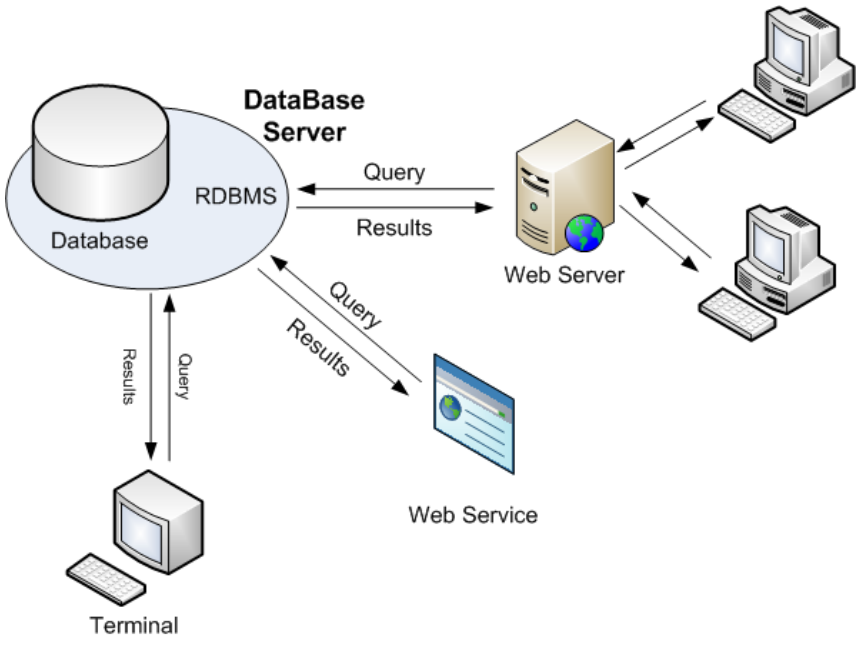
Εργαστήριο Βάσεων Δεδομένων

Stored Procedures

Αρχιτεκτονική επικοινωνίας με τη βάση δεδομένων

- Μια βάση χρησιμοποιείται μέσω του client-server μοντέλου
  - Τα δεδομένα είναι αποθηκευμένα στον DB Server.
  - Οι clients καλούν τον DB Server 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 DB Server και κυρίως των αποτελεσμάτων από τον DB Server.



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 στον 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.
- 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.

Δημιουργια, κλήση και διαγραφή της stored procedure

Δημιουργία

Εντολή CREATE PROCEDURE <όνομα procedure>

Κλήση

Εντολή CALL <όνομα procedure>

Διαγραφή

Εντολή DROP PROCEDURE <όνομα procedure>

Ανάκτηση Κώδικα

Εντολή SHOW CREATE PROCEDURE <όνομα procedure>

Ανάκτηση Λίστας Procedures

Εντολή SHOW PROCEDURE STATUS

Δημιουργία και κλήση stored procedure - Παράδειγμα

- Δήλωση μιας stored procedure

  Όνομα hello_world

  Λειτουργία Εκτέλεση μιας select

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

- Κλήση stored procedure

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

Επιστρέφει:

name	lastname	am
Βάσω	unknown	1845
Βιβή	Τζέκου	2191
unknown	Ντούρου	2192
Αθανασία	Κουμπούρη	2193

Ανάκτηση και διαγραφή μιας stored procedure - Παράδειγμα

- Ανάκτηση του κώδικα μιας stored procedure

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

- Ανάκτηση της λίστας όλων των stored procedures

```
mysql>SHOW PROCEDURE STATUS;
```

- Διαγραφή stored procedure

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

Δήλωση blocks κώδικα

- Στο σώμα των procedures είναι δυνατόν να περιλαμβάνονται πολλαπλές εντολές.
  - Χρειάζεται η δυνατότητα ορισμού blocks εντολών

=> Χρήση της δομής **BEGIN** ...*εντολές του block*... **END**

  - Ο χαρακτήρας τερματισμού των εσωτερικών εντολών πρέπει να είναι διαφορετικός από τον χαρακτήρα τερματισμού της CREATE PROCEDURE εντολής.
    - Γιατί?

=> Αλλαγή του χαρακτήρα τερματισμού με την εντολή **DELIMITER**

Δήλωση blocks κώδικα - Παράδειγμα

- Δήλωση μιας stored procedure με πολλαπλές εντολές στο σώμα της
  - Όνομα: hello_world2
  - Λειτουργία: Εκτέλεση τριών select
- Αλλαγή του 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()
    ->BEGIN
    ->    SELECT * FROM student ORDER BY am;
    ->    SELECT * FROM course ORDER BY course_id;
    ->    SELECT * FROM registration;
    ->END$
```

- Επαναφορά του χαρακτήρα τερματισμού εντολών από $ σε ;

```
mysql>DELIMITER ;
```

- Κλήση της stored procedure

```
mysql>CALL hello_world2();
```

Ορισμός μεταβλητών στο περιβάλλον της mysql

- Η mysql επιτρέπει τη δήλωση μεταβλητών στο περιβάλλον της
  - Η εμβέλειά τους είναι το τρέχον session.
  - Το όνομά τους πρέπει να ξεκινά με @ για να διαχωρίζονται από τις μεταβλητές του συστήματος.
  - Η ανάθεση τιμής γίνεται με χρήση της εντολής **SET**

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

  - Η εκτύπωση της τιμής γίνεται με την εντολή **SELECT**

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

Είσοδος/Έξοδος 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/Έξοδος σε procedures - Παράδειγμα 1/2

- Stored procedure με:
  - Όνομα afairesi
  - Λειτουργία Αφαίρεση δύο αριθμών και επιστροφή του αποτελέσματος
- Δημιουργία της stored procedure:

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE afairesi(IN a INT, IN b INT, OUT result INT)
    ->BEGIN
    ->   SET result=a-b;
    ->END$
mysql>DELIMITER ;
```

- Κλήση:

```
mysql>CALL afairesi(5,4,@res);
```

- Αποτέλεσμα:

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

Είσοδος/Έξοδος σε procedures - Παράδειγμα 2/2

- Stored procedure με:
  - Όνομα arnisi
  - Λειτουργία Επιστροφή της άρνησης ενός αριθμού
- Δημιουργία της stored procedure:

```
mysql>DELIMITER $
mysql>CREATE PROCEDURE arnisi(INOUT num INT)
    ->BEGIN
    ->   SET num=-num;
    ->END$
mysql>DELIMITER ;
```

- Κλήση:

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

Τοπικές μεταβλητές σε 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.

Παραδείγματα DECLARE

```
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 όλων των υπόλοιπων εντολών!

Τοπικές μεταβλητές σε procedures - Παράδειγμα

- Stored procedure με:
  - Όνομα swap
  - Λειτουργία Δέχεται δύο μεταβλητές και ανταλλάσσει τις τιμές τους
- Δημιουργία της stored procedure:

```
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$
mysql>DELIMITER ;
```

- Κλήση:

```
mysql>SET @n1='vivi';
mysql>SET @n2='athanasia';
mysql>CALL swap(@n1,@n2);
mysql>SELECT @n1,@n2;
+-----------+------+
| @n1       | @n2  |
+-----------+------+
| athanasia | vivi |
+-----------+------+
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

Σύνταξη

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 ;
```

IF-THEN-ELSE - Παράδειγμα

- Stored procedure με:
  - Όνομα pointOfTime
  - Λειτουργία Δέχεται μια ημερομηνία και εκτυπώνει αν είναι παρόν, μέλλον ή παρελθόν
- Δημιουργία της stored procedure:

```
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 ;
```

- Κλήση:

```
mysql>CALL pointOfTime('2011-12-31');
+------+
| past |
+------+
| past |
+------+
1 row in set (0.00 sec)
```

CASE

Σύνταξη

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

Παράδειγμα

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

Δομές επανάληψης

Σύνταξη WHILE

WHILE condition
  DO statement/s
END WHILE;

Σύνταξη REPEAT

REPEAT statement/s
  UNTIL condition
END REPEAT;

WHILE

Παράδειγμα - Procedure απαρίθμησης

```
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 ;
```

Κλήση

```
mysql>CALL simpleFor(2);

+------+
| i    |
+------+
|    0 |
+------+
1 row in set (0.00 sec)

+------+
| i    |
+------+
|    1 |
+------+
1 row in set (0.01 sec)


mysql>CALL simpleFor(0);

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

REPEAT

Παράδειγμα - Procedure απαρίθμησης

```
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 ;
```

Κλήση

```
mysql>CALL simpleForAlt(2);

+------+
| i    |
+------+
|    0 |
+------+
1 row in set (0.00 sec)
+------+
| i    |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql>CALL simpleFor(0);

+------+
| i    |
+------+
|    0 |
+------+
1 row in set (0.00 sec)
```

Διαχείριση αποτελεσμάτων select με κώδικα

- Η βασικότερη λειτουργία των stored procedures είναι η διαχείριση 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 επιστρεφονται:

INTO

- Χρησιμοποιείται όταν το select επιστρέφει μία εγγραφή
- Αποθηκεύουμε σε μεταβλητές τις τιμές που επιστρέφονται

CURSORS

- Χρησιμοποιούνται όταν το select επιστρέφει πολλαπλές εγγραφές
- Προσπελαύνουμε τις γραμμές των αποτελεσμάτων μία-μια

Διαχείριση μεμονωμένων τιμών - SELECT INTO

Σύνταξη

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.

Διαχείριση μεμονωμένων τιμών - Παράδειγμα

Procedure που επιστρέφει στοιχεία ενός φοιτητή

```
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 ;
```

Διαχείριση μεμονωμένων τιμών - Κλήση

Κλήση της showStudentInfo

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
- Προσπελαύνουμε μία εγγραφή των αποτελεσμάτων κάθε φορά,
- Βασικά σημεία:
  - Ορίζουμε μια μεταβλητή τύπο cursor.
  - Την αντιστοιχίζουμε στη select που θα επιστρέψει τα αποτελέσματα.
  - Ορίζουμε μια εντολή που θα εκτελεστεί όταν έχουν διαβαστεί όλα τα αποτελέσματα.
  - Εκτελούμε την εντολή FETCH που μεταφέρει τον cursor στην επόμενη γραμμή, μέχρι να εκτελεστεί η παραπάνω εντολή.

Διαχείριση πινάκων αποτελεσμάτων - Εντολές

- Ορισμός ενός cursor και δήλωση της select στην οποία θα εφαρμοστεί

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

- Δήλωση εντολής 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;

- Άνοιγμα του cursor, δηλαδή εκτέλεση του select που συνδέεται με αυτόν

OPEN <όνομα cursor>;

- Ανάγνωση της επόμενης γραμμής στα αποτελέσματα

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

- Συνθήκη τερματισμού
  - Σταματάμε να διαβάζουμε όταν το flag που ορίσαμε γίνει 1
- Κλείσιμο του cursor

CLOSE <όνομα cursor>;

Διαχείριση πινάκων αποτελεσμάτων - Παράδειγμα 1/2

Procedure που επιστρέφει τα στοιχεία των lectures ενός μαθήματος

```
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 'Αριθμός Διάλεξης', lectSubject AS 'Θέμα';
    ->      FETCH lectCursor INTO lectSubject, lectNum;
    ->   END WHILE;
    ->   CLOSE  lectCursor;
    ->END$
mysql>DELIMITER ;
```

Διαχείριση πινάκων αποτελεσμάτων - Παράδειγμα 2/2

Procedure που επιστρέφει τα στοιχεία των lectures ενός μαθήματος

```
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 'Αριθμός Διάλεξης', lectSubject AS 'Θέμα';
    ->      END IF;
    ->   UNTIL(finishedFlag=1)
    ->   END REPEAT;
    ->   CLOSE  lectCursor;
    ->END$
mysql>DELIMITER ;
```