ΠΕΡΙΕΧΟΜΕΝΑ:

1. Αποθηκευμένες Διαδικασίες
    1. Μεταβλητές
    2. Δομές Ελέγχου
    3. Δομές Επανάληψης
    4. Παράμετροι και Επιστρεφόμενες Τιμές
2. Αποθηκευμένες Συναρτήσεις

Ευάγγελος Μ.

Σμαραγδένιος Χορηγός Μαθήματος

Κώστας Επταήμερος

Χρυσός Χορηγός Μαθήματος

1. Αποθηκευμένες Διαδικασίες

Μία Αποθηκευμένη Διαδικασία (Stored Procedure):

- Είναι το λογικό αντίστοιχο της **συνάρτησης** σε μία γλώσσα προγραμματισμού, δηλαδή μιας ρουτίνας που παίρνει ορίσματα εκτελεί κάποιες ενέργειες και έχει ένα αποτέλεσμα (π.χ. επιστρέφει ένα result set)
- Υποστηρίζει μεταβλητές, έλεγχο, επανάληψη κ.α. Μέσω των συντακτικών στοιχείων μπορούμε να ομαδοποιήσουμε ενέργειες SQL (π.χ. μία εισαγωγή επί πολλών πινάκων) σε ένα όνομα διαδικασίας, το οποίο μπορούμε να καλέσουμε, είτε μέσω π.χ. του MySQL Workbench, είτε μέσω ενός προγρ/τος.

Δημιουργία stored procedure (χωρίς ορίσματα):

```
DELIMITER $$
CREATE PROCEDURE proc_name()
BEGIN
    ...
END$$
DELIMITER ;
```

- Στο τέλος της διαδικασίας βάζουμε εντολή SELECT (που θα είναι και το επιστρεφόμενο αποτέλεσμα)
- Η **DELIMITER** αντικαθιστά το ερωτηματικό με αυτό που θέτουμε (εδώ $$). Η αλλαγή είναι υποχρεωτική, ώστε να μπορούμε να χρησιμοποιήσουμε το ερωτηματικό στο σώμα της διαδικασίας. Επαναφέρουμε το DELIMITER σε ερωτηματικό με το πέρας της διαδικασίας.

Κλήση stored procedure:

```
CALL proc_name();
```

Διαγραφή stored procedure:

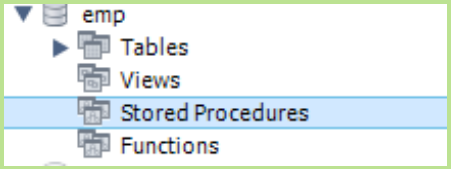
```
DROP PROCEDURE [IF EXISTS] proc_name;
```

Παράδειγμα 1: DB: emp, script: stored.sql

```
DROP PROCEDURE IF EXISTS message;
DELIMITER $$
CREATE PROCEDURE message()
BEGIN
    SELECT 'Hello World!" AS message;
END$$
DELIMITER ;
CALL message();
```

Stored Procedures στο MySQL Workbench:

- Βρίσκονται στα στοιχεία του πίνακα:



- και μπορούμε εύκολα να κατασκευάσουμε ή να διαγράψουμε μία stored procedure (βλέπε βίντεο)

Δήλωση μεταβλητής:

- Αμέσως μετά το BEGIN, με το συντακτικό:

```
DECLARE var_name data_type [DEFAULT value];
```

- Η χρήση της DEFAULT αναθέτει την τιμή value στη μεταβλητή

Ανάθεση τιμής σε μεταβλητής με εντολή:

```
SET var_name  = expression;
```

- όπου έκφραση είναι τιμή, υπολογιζόμενη ποσότητα, κλήση συνάρτησης ή ακόμη και υποερώτημα.

Ανάθεση τιμών σε μεταβλητές με SELECT:

```
SELECT val1, val2,… INTO var1,var2,… FROM …;
```

- Αναθέτει τη val1 στη var1, τη val2 στη var2 κ.οκ., όπως αυτές επιλέχθηκαν από το ερώτημα.

Τύποι Μεταβλητών:

- Συνήθεις μεταβλητές: έχουν εμβέλεια στο begin…end
- Μεταβλητές χρήστη (αρχίζουν με @): Είναι μεταβλητές που διατηρούν την τιμή τους και μετά την κλήση της συνάρτησης.
  - Αρχικοποιούνται με κάποιο τρόπο εκτός της διαδικασίας.
  - Καλούνται και μεταβλητές χρήστη (κάθε χρήστης έχει δικές του τιμές σε αυτές τις μεταβλητές)
  - Είναι ορατές και εκτός της διαδικασίας.
- Μεταβλητές συστήματος (αρχίζουν με @@): Είναι μεταβλητές με πληροφορίες συστήματος, ορατές σε όλους.

Παράδειγμα 2: DB: emp, script: variables.sql

```
CREATE PROCEDURE variable()
BEGIN
    DECLARE x INT;
    SET x=1;
    SELECT x;
END
```

Παράδειγμα 3: DB: emp, script: uservariables.sql

```
CREATE PROCEDURE variable()
BEGIN
     SET @x = @x + 1;
     SELECT @x;
END
```

Άσκηση 1: DB: world

Δημιουργήστε stored procedure με όνομα min_max η οποία θα επιστρέφει:

- Τη χώρα με το μικρότερο πληθυσμό (που να μην είναι 0)
- Τη χώρα με το μεγαλύτερο πληθυσμό.

IF…ELSE:

```
IF condition1 THEN
   statements1
[ELSEIF condition2 THEN
   statements2
…
]
[ELSE
   statementsN
…
]
END IF;
```

CASE (επί τιμών μεταβλητής):

```
CASE var THEN
    WHEN value1 THEN
        statements1
    [WHEN value2 THEN
        statements2]
    …
    ELSE
        statementsN
END CASE;
```

CASE (με πολλαπλές συνθήκες):

```
CASE
    WHEN condition1 THEN
        statements1
    [WHEN condition2 THEN
        statements2]
    …
    ELSE
        statementsN
END CASE;
```

Παράδειγμα 4: DB: emp, script: if.sql

```
CREATE PROCEDURE independence()
BEGIN
    DECLARE vyear INT DEFAULT 1700;
    DECLARE vdiff INT DEFAULT 100;

    IF vyear+vdiff > EXTRACT(YEAR FROM NOW()) THEN
        SELECT 'invalid values' AS error_message;
    ELSE
        SELECT name, indepYear
        FROM country
        WHERE indepYear BETWEEN vyear-vdiff AND vyear+vdiff;
    END IF;
END
```

Προσφέρονται οι συνήθεις δομές επανάληψης:

- WHILE και REPEAT…UNTIL

καθώς και μια λιγότερο συνηθισμένη (LOOP) που εκτελεί έναν ατέρμονα βρόχο.

WHILE:

```
WHILE condition DO
   statements
END WHILE;
```

REPEAT…UNTIL:

```
REPEAT
   statements
UNTIL condition
END REPEAT;
```

LOOP:

```
loop_name: LOOP
   statements
END LOOP loop_name;
```

- Επαναλαμβάνει διαρκώς το σώμα (statements)
- προσφέρονται οι εντολές:
  - LEAVE loop_name; (διακόπτει την εκτέλεση του βρόχου)
  - ITERATE loop_name; (επαναλαμβάνει το βρόχο)

Παράδειγμα 4: DB: emp, script: loops.sql

```
CREATE PROCEDURE loops()
BEGIN
    DECLARE str1,str2,str3 VARCHAR(80);
    DECLARE i INT;

    SET i = 0;
    SET str1 = '';
    WHILE i<5 DO
        SET str1 = CONCAT(str1,' ',i);
        SET i = i+1;
    END WHILE;
    SET i = 0;
    SET str2 = '';
    REPEAT
        SET str2 = CONCAT(str2,' ',i);
        SET i = i+1;
    UNTIL i>5
    END REPEAT;

    SET i = 0;
    SET str3 = '';
    myloop: LOOP
        SET str3 = CONCAT(str3,' ',i);
        IF i>5 THEN
            LEAVE myloop;
        END IF;
        SET i = i+1;
    END LOOP myloop;

    SELECT str1, str2, str3;
END
```

Παράμετροι Διαδικασίας:

```
CREATE PROCEDURE proc_name(
    [IN|OUT|INOUT] param_name1 datatype1,
    [IN|OUT|INOUT] param_name2 datatype2,
    ...
)
BEGIN
    ...
END
```

- **IN**: παράμετρος εισόδου. Δεν μπορεί να αλλάξει η τιμή της. Default αν παραληφθεί ο προσδιοριστής.
- **OUT**: παράμετρος εξόδου.
- **INOUT**: παράμετρος εισόδου/εξόδου.

Παράδειγμα 5: DB: sakila, script: inparams.sql

```
CREATE PROCEDURE input(IN actor_id_param INT)
BEGIN
    SELECT CONCAT(first_name,' ', last_name) AS full_name,

           COUNT(fa.film_id) AS movies
    FROM actor a JOIN film_actor fa
         ON a.actor_id = fa.actor_id
    WHERE a.actor_id = actor_id_param;
END
```

Παράδειγμα 6: DB: emp, script: outparams.sql

```
CREATE PROCEDURE output(
    IN actor_id_param INT,
    OUT actor_movies INT
)
BEGIN
    SELECT COUNT(fa.film_id) AS movies
    INTO actor_movies
    FROM actor a JOIN film_actor fa
         ON a.actor_id = fa.actor_id
    WHERE a.actor_id = actor_id_param;
END
```

Παρατήρηση:

- Η επιστρεφόμενη τιμή μπορεί να αποθηκευτεί σε μία μεταβλητή χρήστη για να έχουμε πρόσβαση στην τιμή μετά την κλήση.
- ή μπορούμε να τη διαχειριστούμε μέσω της γλώσσας προγραμματισμού σε ένα πρόγραμμα.

Παράδειγμα 6 (συνέχεια…)

```
CALL output(1, @cnt);
SELECT @cnt;
```

Αποθηκευμένες Συναρτήσεις (Stored Functions):

- Χρησιμοποιώντας το ίδιο συντακτικό με τις διαδικασίες, μπορούμε να κατασκευάσουμε συναρτήσεις χρήστη (με ίδια χρήση με αυτή των συναρτήσεων της MySQL - βλ. μάθημα 1.10-1.11)
- Χρησιμοποιείται η λέξη-κλειδί **FUNCTION** αντί της PROCEDURE
- Παίρνει μόνο ορίσματα εισόδου (όχι εξόδου) και δεν γράφουμε το προσδιοριστικό IN κατά τη δήλωση και ο τύπος επιστροφής δηλώνεται στην κεφαλίδα της συνάρτησης.

```
DELIMITER $$
CREATE FUNCTION proc_name(params) STATEMENT
BEGIN
    …
    RETURN value;
END$$
DELIMITER ;
```

- όπου STATEMENT, πρέπει να είναι ένα (τουλάχιστον) από τα επόμενα:
  - DETERMINISTIC: παράγει το ίδιο αποτέλεσμα για κάθε είσοδο.
  - NO SQL: Δεν περιέχει εντολές SQL
  - READS SQL DATA: Διαβάζει (μόνο) δεδομένα SQL

Παράδειγμα 7: DB: emp, script: myfunc.sql

```
CREATE FUNCTION myfunc(n INT)
RETURNS INT
DETERMINISTIC NO SQL
BEGIN
    DECLARE i INT DEFAULT 1;
    DECLARE fact INT DEFAULT 1;
    WHILE i<=N DO
        SET fact = fact*i;
        SET i=i+1;
    END WHILE;
    RETURN fact;
END
```

Παράδειγμα 8: DB: emp, script: myfunc.sql

```
CREATE FUNCTION fname(last_name_param VARCHAR(80))
RETURNS VARCHAR(80)
READS SQL DATA
BEGIN
    DECLARE first_name_var VARCHAR(80);
    SELECT first_name
    INTO first_name_var
    FROM actor
    WHERE last_name = last_name_param
    LIMIT 1;
    RETURN first_name_var;
END
```