**Βάσεις Δεδομένων ΙΙ (Ε)**

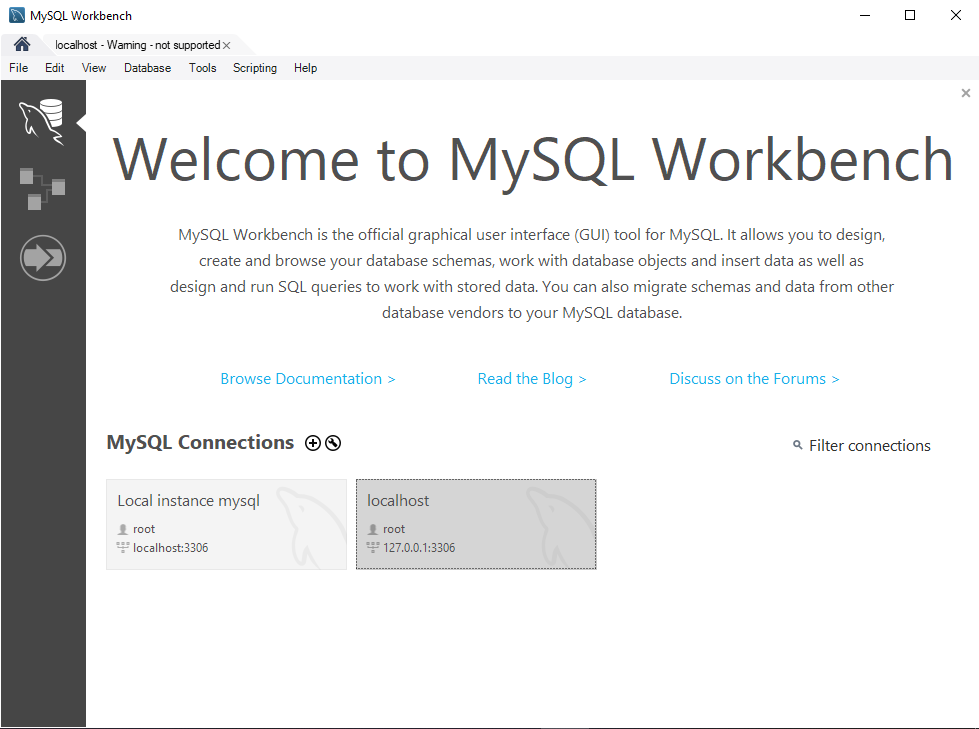
Α.Μ.: ice18390094

Ονοματεπώνυμο: Άγγελος Τζώρτζης

Εργαστηριακή άσκηση 5

Τμήμα: [00] Χωρίς παρακολούθηση

**1.** Σύνδεση στην MySQL του συστήματος μας:

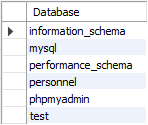


**2.** Ελέγχουμε εάν υπάρχει η ΒΔ με όνομα my\_accounts:

Εμφάνιση όλων των ΒΔ του συστήματος μας

show databases;

ΒΔ στο σύστημα μας:



Δεν υπάρχει η ΒΔ my\_accounts στο σύστημα μας.

**3.** Δημιουργία της ΒΔ my\_accounts, επιλογή της για χρήση και δημιουργία πίνακα με όνομα Accounts με δομή και περιεχόμενα, όπως φαίνονται στις ακόλουθες εντολές. Δείχνουμε το αποτέλεσμα εμφανίζοντας (α) τη λίστα πινάκων της ΒΔ, (β) τα περιεχόμενα και (γ) τη δομή του πίνακα Accounts:

Εντολές για την δημιουργία και χρήση της ΒΔ my\_accounts:

drop database if exists my\_accounts;

create database my\_accounts;

use my\_accounts;

Εντολές για για την δημιουργία του πίνακα Accounts:

create table Accounts (

acctID integer not null primary key,

Balance integer not null

);

Εντολές για την εισαγωγή δεδομένων στον πίνακα Accounts:

insert into Accounts (acctID, Balance) values (101, 1000);

insert into Accounts (acctID, Balance) values (202, 2000);

insert into Accounts (acctID, Balance) values (303, 2500);

insert into Accounts (acctID, Balance) values (404, 3000);

Λίστα πινάκων της ΒΔ:

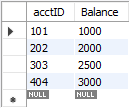


Βλέπουμε ότι ο μόνος πίνακας που υπάρχει είναι ο πίνακας Accounts καθώς είναι ο μόνος που φτιάξαμε για την συγκεκριμένη ΒΔ.

Εμφάνιση περιεχομένων του πίνακα Accounts:

select \* from Accounts;

Περιεχόμενα του πίνακα Accounts:



Τα περιεχόμενα συμβαδίζουν με αυτά που εισάγαμε με τις εντολές μας

Εμφάνιση δομής του πίνακα Accounts:



Η δομή του πίνακα συμβαδίζει με αυτήν που ορίσαμε στην δημιουργία του.

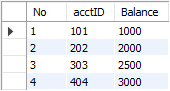
**4.** Εμφάνιση των περιεχομένων του πίνακα Accounts, προσθέτοντας αύξουσα αρίθμηση στις εγγραφές του:

set @row\_number = 0;

select (@row\_number:=@row\_number+1) as No, acctID, Balance

from Accounts order by acctID;

Αποτέλεσμα εντολών:



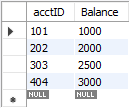
Βλέπουμε δίπλα σε κάθε εγγραφή υπάρχει αρίθμηση χωρίς όμως να γίνεται εισαγωγή σε κάποιον πίνακα, καθώς έχουμε ορίσει μία μεταβλητή που “μετράει” κάθε νέα εγγραφή.

**5.** Η αύξουσα αρίθμηση που εμφανίστηκε στην στήλη με τίτλο No του βήματος 4, θα πρέπει να υπάρχει και στον πίνακα Accounts; Αιτιολογήστε την απάντησή σας.

Όχι δεν θα έπρεπε να υπάρχει στήλη με όνομα Νο στον πίνακα Accounts καθώς δεν ορίστηκε ενημέρωση, τροποποίηση η δημιουργία του πίνακα με στήλη με τίτλο Νο, η στήλη No ορίστηκε ξεχωριστά σε εντολή select σαν απαριθμητής εγγραφών.

Εμφάνιση περιεχόμενα του πίνακα Accounts:

select \* from Accounts;



Βλέπουμε πως όντως δεν προστέθηκε στήλη No στον πίνακα Accounts.

**6.** Προσθέση του πίνακα CUSTOMERS που εμφανίζεται στην Εικόνα 1. Ορίζουμε τύπους δεδομένων των στηλών CUSTNO και CUST\_NAME ως integer και varchar(30) αντίστοιχα. Δείχνουμε το αποτέλεσμα εμφανίζοντας (α) τη λίστα πινάκων της ΒΔ, (β) τα περιεχόμενα και (γ) τη δομή του πίνακα CUSTOMERS:

Εντολές για για την δημιουργία του πίνακα CUSTOMERS:

create table CUSTOMERS (

CUSTNO integer not null primary key,

CUST\_NAME varchar(30)

);

Εντολές για την εισαγωγή δεδομένων στον πίνακα CUSTOMERS:

insert into CUSTOMERS (CUSTNO, CUST\_NAME) values (10, "101");

insert into CUSTOMERS (CUSTNO, CUST\_NAME) values (20, "202");

Λίστα πινάκων της ΒΔ:

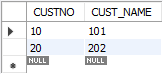


Βλέπουμε ότι προστέθηκε ο πίνακας CUSTOMERS στην ΒΔ.

Εμφάνιση περιεχομένων του πίνακα CUSTOMERS:

select \* from CUSTOMERS;

Περιεχόμενα του πίνακα CUSTOMERS:



Τα περιεχόμενα συμβαδίζουν με αυτά που εισάγαμε με τις εντολές μας

Εμφάνιση δομής του πίνακα CUSTOMERS:



Η δομή του πίνακα συμβαδίζει με αυτήν που ορίσαμε στην δημιουργία του.

**7.** Στον πίνακα Accounts προσθέτουμε στήλη με όνομα Custno, τύπο δεδομένων integer και την ορίζουμε ως FK του πίνακα Accounts για τη σύνδεση των εγγραφών του με τις εγγραφές του πίνακα CUSTOMERS. Ενημερώνουμε τα περιεχόμενα της στήλης Custno, ώστε ο λογαριασμός με AcctID=202 να αντιστοιχεί στον κωδικό πελάτη 20 και όλοι οι υπόλοιποι λογαριασμοί να αντιστοιχούν στον κωδικό πελάτη 10. Δείχνουμε το αποτέλεσμα εμφανίζοντας (α) τα περιεχόμενα και (β) τη δομή του πίνακα Accounts.

alter table Accounts add Custno integer;

alter table Accounts add foreign key(Custno)

references CUSTOMERS(CUSTNO);

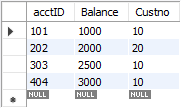
update Accounts set Custno = 20 where acctID = 202;

update Accounts set Custno = 10 where acctID != 202;

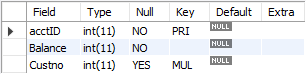
select \* from Accounts;

describe Accounts;

Περιεχόμενα του πίνακα Accounts:



Βλέπουμε την προσθήκη της στήλης Custno στον πίνακα account με τις τιμές Custno που θέλουμε από την εκφώνηση.

Δομή του πίνακα Accounts:  


Βλέπουμε ότι έγιναν οι τροποποιήσεις στην δομή του πίνακα με βάση τις εντολές μας.

**8.** Εκτέλεση και ερμήνευση των δηλώσεων SQL:

select CUSTNO, count(\*), sum(Balance)

from Accounts

where CUSTNO not in (20)

group by CUSTNO;

-- Παραλλαγή με χρήση μεταβλητής.

set @CUSTNO = 20;

select CUSTNO, count(\*), sum(Balance)

from Accounts

where CUSTNO not in (@CUSTNO)

group by CUSTNO;

Οι παραπάνω δηλώσεις μας εμφανίζουν απο τον πίνακα Accounts τον αριθμό των εγγραφών που δέν έχουν CUSTNO = 20, το άθροισμα του Balance αυτών των εγγραφών και τον CUSTNO αυτών των εγγραφών ομαδοποιημένα με βάση τον CUSTNO.

Αποτέλεσμα της εκτέλεσης των εντολών:



Η ομαδοποίηση δεν φαίνεται στην συγκεκριμένη περίπτωση καθώς οι εγγραφές που δέν έχουν CUSTNO = 20 έχουν την ίδια τιμή.

**9.** Εκτέλεση και ερμήνευση των δηλώσεων SQL:

select count(\*), sum(Balance) from Accounts;

Με αυτή την εντολή εμφανίζονται όλες οι εγγραφές και το άθροισμα των Balance τους από τον πίνακα Accounts.

Αποτέλεσμα εκτέλεσης εντολής:



-- Παραλλαγή με χρήση μεταβλητής.

set @COUNT\_acctID = 0, @SUM\_acctID = 0, @AVG\_acctID = 0;

Ορίζουμε τις μεταβλητές και τις αρχικοποιούμε.

select count(\*), sum(Balance), avg(Balance)

into @COUNT\_acctID, @SUM\_acctID, @AVG\_acctID

from Accounts;

Βάζουμε τις τιμές από τον πίνακα Accounts με την select στις αντίστοιχες μεταβλητές.

select @COUNT\_acctID, @SUM\_acctID, @AVG\_acctID,

@MY\_AVG := @SUM\_acctID / @COUNT\_acctID;

Εμφανίζουμε τις τιμές των μεταβλητών καθώς και ορίζουμε νέα μεταβλητή @ΜΥ\_AVG και υπολογίζουμε με δικό μας τύπο τον μέσο όρο με χρήση των μεταβλητών και τον εμφανίζουμε.

Αποτέλεσμα εκτέλεσης εντολής:



Βλέπουμε ότι ο μέσος όρος βγαίνει ίδιο και με τον δικό μας τύπο και με χρήση της συνάρτησης avg.

**10.** Ορίζουμε και χρησιμοποιούμε τη συνάρτηση factorial που υπολογίζει το n!:

drop function if exists factorial;

delimiter !

create function factorial(N int)

returns int

deterministic

begin

declare F int default 1;

while N > 0 do

set F = N \* F;

set N = N - 1;

end while;

return F;

end !

delimiter ;

select factorial(4);

select factorial(15);

Αποτέλεσμα εκτέλεσης εντολών:



Το αποτέλεσμα είναι σωστό καθώς 4! = 24.



Το αποτέλεσμα είναι λάθος καθώς 15! = 1307674368000. Το αποτέλεσμα που προκύπτει είναι λάθος καθώς η πραγματική τιμή του 15! ξεπερνάει το όριο τον int και υπάρχει υπερχείλιση.

**11.** Ορίζουμε και χρησιμοποιούμε την διαδικασία my\_procedure\_Local\_Variables για υπολογισμούς με χρήση τοπικών μεταβλητών.

drop procedure if exists my\_procedure\_Local\_Variables;

delimiter $$

create procedure my\_procedure\_Local\_Variables()

begin

set @X = 25;

set @Y = 10;

select @X, @Y, @X\*@Y;

end $$

delimiter ;

-- Κλήση της διαδικασίας

call my\_procedure\_Local\_Variables();

Αποτέλεσμα μετά την κλήση της διαδικασίας:



**12.** Ακολουθούμε τα ακόλουθα για τη δημιουργία μιας αποθηκευμένης διαδικασίας και χρήση των commit/rollback. Επεξήγηση της διαδικασία my\_proc.

SET @p\_no=3;

SELECT MOD(@p\_no, 2);

SET @p\_no=8;

SELECT MOD(@p\_no, 2);

-- Δημιουργία βάσης και πίνακα

DROP TABLE IF EXISTS myTrace;

CREATE TABLE myTrace (

t\_no INT,

t\_user CHAR(20),

t\_date DATE,

t\_time TIME,

t\_proc VARCHAR(16),

t\_what VARCHAR(30)

);

-- Δημιουργία αποθηκευμένης διαδικασίας myProc

DROP PROCEDURE IF EXISTS myProc;

DELIMITER !

CREATE PROCEDURE myProc (

IN p\_no INT,

IN p\_in VARCHAR(30),

OUT p\_out VARCHAR(30)

)

LANGUAGE SQL

BEGIN

SET p\_out=p\_in;

INSERT INTO myTrace (t\_no, t\_user, t\_date, t\_time, t\_proc, t\_what)

VALUES (p\_no, current\_user, current\_date, current\_time, 'myProc', p\_in);

IF (MOD(p\_no, 2)=0) THEN

COMMIT;

ELSE ROLLBACK;

END IF;

END !

DELIMITER ;

-- Κλήση της διαδικασίας

SET AUTOCOMMIT=0;

CALL myProc(1, 'hello1', @out);

CALL myProc(2, 'hello2', @out);

CALL myProc(3, 'hello3', @out);

CALL myProc(4, 'hello4', @out);

CALL myProc(5, 'hello5', @out);

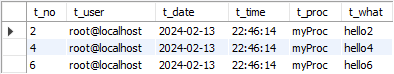
CALL myProc(6, 'hello6', @out);

CALL myProc(7, 'hello7', @out);

SELECT \* FROM myTrace;

Η διαδικασία my\_proc δέχεται 2 ορίσματα εισόδου p\_no, p\_in με τύπο δεδομένων int και varchar(30) αντίστοιχα και παράμετρο εξόδου p\_out με τύπο δεδομένων varchar(30). Αρχικά ορίζει την τιμή p\_out ώς την τιμή του p\_in, στην συνέχεια στον πίνακα my\_trace εισαγει τα εξής δεδομένα: p\_no, τον τρέχον χρήστη,την τρέχουσα ημερομηνία, την ακριβής ώρα που έγινε κλήση της διαδικασίας, την συμβολοσειρά ‘my\_proc’, p\_in. Μετα ελέγχει αν το αποτέλεσμα του p\_no modulo 2 ισούται με 0. Αν ναι τότε πραγματοποιούνται οι αλλαγές(COMMIT), αλλιώς αναιρούνται όποιες αλλαγές έχουν γίνει.

Πίνακας myTrace μετά την κλήση των διαδικασιών:



Βλέπουμε ότι έγιναν μόνο οι αλλαγές από τις διαδικασίες με p\_no που διαιρείται ακριβώς με το 2.

**13.** Στον πίνακα Accounts (Εικόνα 1) η μεταφορά χρημάτων από ένα λογαριασμό σε έναν άλλο θα μπορούσε να υλοποιηθεί με δύο δηλώσεις UPDATE. Ακολουθεί παράδειγμα επίλυσης με χρήση συναλλαγής (transaction). Η συναλλαγή αυτή χαρακτηρίζεται ως αναξιόπιστη, καθώς δεν γίνεται έλεγχος σχετικά: (α) με την ύπαρξη του λογαριασμού στον οποίο μεταφέρονται τα χρήματα και (β) την επάρκεια του λογαριασμού από τον οποίο μεταφέρονται τα χρήματα:

DROP TABLE IF EXISTS Accounts;

CREATE TABLE Accounts (

acctID INTEGER NOT NULL PRIMARY KEY,

balance INTEGER NOT NULL,

CONSTRAINT unloanable\_account CHECK (balance >= 0)

);

INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);

INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);

COMMIT;

SELECT \* FROM accounts;

-- Συναλλαγή.

START TRANSACTION;

UPDATE Accounts SET balance = balance - 100

WHERE acctId = 101;

UPDATE Accounts SET balance = balance + 100

WHERE acctId = 202;

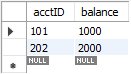
COMMIT;

-- Αποτέλεσμα συναλλαγής.

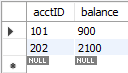
SELECT \* FROM accounts;

Αποτέλεσμα της εκτέλεσης του κώδικα:

Πρίν:



Μετά:



**14.** Ακολουθεί λύση του προβλήματος στο βήμα 15 με χρήση της procedure BankTransfer:

DELIMITER //

DROP PROCEDURE if exists BankTransfer //

CREATE PROCEDURE BankTransfer (

IN fromAcct INT,

IN toAcct INT,

IN amount INT,

OUT msg VARCHAR(100)

)

P1: BEGIN

DECLARE rows1 INT ;

DECLARE newbalance INT;

SELECT COUNT(\*) INTO rows1 FROM Accounts WHERE acctID = fromAcct;

UPDATE Accounts SET balance = balance - amount WHERE acctID = fromAcct;

SELECT balance INTO newbalance FROM Accounts WHERE acctID = fromAcct;

IF rows1 = 0 THEN

ROLLBACK;

SET msg = CONCAT('rolled back because of missing account ', fromAcct);

ELSEIF newbalance < 0 THEN

ROLLBACK;

SET msg = CONCAT('rolled back because of negative balance of account ', fromAcct);

ELSE

SELECT COUNT(\*) INTO rows1 FROM Accounts WHERE acctID = toAcct;

UPDATE Accounts SET balance = balance + amount WHERE acctID = toAcct;

IF rows1 = 0 THEN

ROLLBACK;

SET msg = CONCAT('rolled back because of missing account ', toAcct);

ELSE

COMMIT;

SET msg = 'committed';

END IF;

END IF;

END P1 //

DELIMITER ;

-- Δοκιμή μεταφοράς 100 από acctID=101 σε acctID=202

SET AUTOCOMMIT=0;

SET @out = ' ';

CALL BankTransfer (101, 202, 100, @out);

SELECT @OUT;

Select \* from accounts;

COMMIT;

-- Δοκιμή μεταφοράς 100 από acctID=101 σε acctID=201 (ανύπαρκτος)

SET autocommit=0;

SET @out = ' ';

CALL BankTransfer (100, 201, 100, @out);

SELECT @OUT;

Select \* from accounts;

COMMIT;

-- Δοκιμή μεταφοράς 100 από acctID=100 (ανύπαρκτος) σε acctID=201

SET autocommit=0;

SET @out = ' ';

CALL BankTransfer (100, 201, 100, @out);

SELECT @OUT;

select \* from accounts;

-- Δοκιμή μεταφοράς 1500 από acctID=101 (ανεπαρκής) σε acctID=201

SET AUTOCOMMIT=0;

SET @out = ' ';

CALL BankTransfer (101, 201, 1500, @out);

SELECT @OUT;

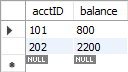
Select \* from accounts;

COMMIT;

Αποτελέσματα εκτέλεσης κώδικα:

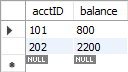
Δοκιμή μεταφοράς 100 από acctID=101 σε acctID=202:





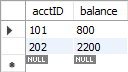
Δοκιμή μεταφοράς 100 από acctID=101 σε acctID=201 (ανύπαρκτος)





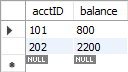
Δοκιμή μεταφοράς 100 από acctID=100 (ανύπαρκτος) σε acctID=201





Δοκιμή μεταφοράς 1500 από acctID=101 (ανεπαρκής) σε acctID=201





**15.** Ακολουθεί δεύτερη λύση του προβλήματος στο βήμα 15:

-- Δημιουργία πίνακα Accounts

DROP TABLE IF EXISTS Accounts;

CREATE TABLE Accounts (acctID INTEGER NOT NULL PRIMARY KEY, balance INTEGER NOT NULL);

INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);

INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);

COMMIT;

SELECT \* FROM accounts;

-- Δημιουργία trigger Accounts\_upd\_trg για έλεγχο των updates

delimiter !

CREATE TRIGGER Accounts\_upd\_trg

BEFORE UPDATE ON Accounts

FOR EACH ROW

BEGIN

IF NEW.balance < 0 THEN

SIGNAL SQLSTATE '23513'

SET MESSAGE\_TEXT = 'Negative balance not allowed';

END IF;

END; !

delimiter ;

-- Δημιουργία trigger Accounts\_ins\_trg για έλεγχο των inserts

delimiter !

CREATE TRIGGER Accounts\_ins\_trg

BEFORE INSERT ON Accounts

FOR EACH ROW

BEGIN

IF NEW.balance < 0 THEN

SIGNAL SQLSTATE '23513'

SET MESSAGE\_TEXT = 'Negative balance not allowed';

END IF;

END; !

delimiter ;

// Δημιουργία procedure BankTransfer

DELIMITER !

CREATE PROCEDURE BankTransfer (IN fromAcct INT,

IN toAcct INT,

IN amount INT,

OUT msg VARCHAR(100))

LANGUAGE SQL MODIFIES SQL DATA

P1: BEGIN

DECLARE acct INT;

DECLARE balance\_v INT;

DECLARE EXIT HANDLER FOR NOT FOUND

BEGIN ROLLBACK;

SET msg = CONCAT('missing account ', CAST(acct AS CHAR));

END;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN ROLLBACK;

SET msg = CONCAT('negative balance (?) in ', fromAcct);

END;

SET acct = fromAcct;

SELECT acctID INTO acct FROM Accounts WHERE acctID = fromAcct ;

UPDATE Accounts SET balance = balance - amount

WHERE acctID = fromAcct;

SET acct = toAcct;

SELECT acctID INTO acct FROM Accounts WHERE acctID = toAcct ;

UPDATE Accounts SET balance = balance + amount

WHERE acctID = toAcct;

SELECT balance INTO balance\_v

FROM accounts

WHERE acctID = fromAcct;

IF balance\_v < 0 THEN

ROLLBACK;

SET msg = CONCAT(' negative balance in ', fromAcct);

ELSE

COMMIT;

SET msg = 'committed';

END IF;

END P1 !

DELIMITER ;

CALL BankTransfer (101, 201, 100, @msg);

Select @msg;

CALL BankTransfer (100, 202, 100, @msg);

Select @msg;

CALL BankTransfer (101, 202, 100, @msg);

Select @msg;

CALL BankTransfer (101, 202, 2000, @msg);

Select @msg;

Αποτελέσματα εκτέλεσης:







