Βάσεις Δεδομένων II (Ε)

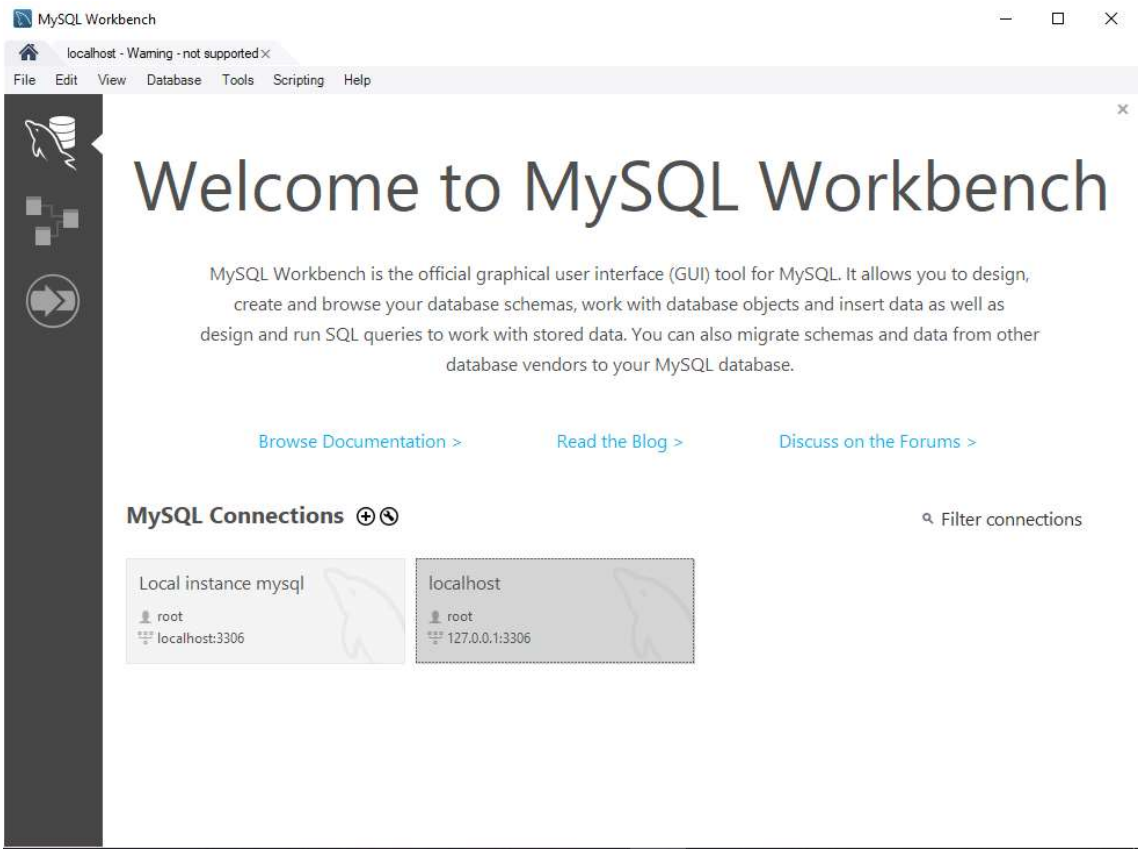
A.M.: ice18390094

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:

	Database
▶	information_schema
	mysql
	performance_schema
	personnel
	phpmyadmin
	test

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:

	Tables_in_my_accounts
▶	accounts

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:

	acctID	Balance
▶	101	1000
	202	2000
	303	2500
	404	3000
*	NULL	NULL

Τα περιεχόμενα συμβαδίζουν με αυτά που εισάγαμε με τις εντολές μας

Εμφάνιση δομής του πίνακα Accounts:

	Field	Type	Null	Key	Default	Extra
▶	acctID	int(11)	NO	PRI	NULL	
	Balance	int(11)	NO		NULL	

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:

	No	acctID	Balance
▶	1	101	1000
	2	202	2000
	3	303	2500
	4	404	3000

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 No στον πίνακα Accounts καθώς δεν ορίστηκε ενημέρωση, τροποποίηση η δημιουργία του πίνακα με στήλη με τίτλο No, η στήλη No ορίστηκε ξεχωριστά σε εντολή select σαν απαριθμητής εγγραφών.

Εμφάνιση περιεχόμενα του πίνακα Accounts:

```
select * from Accounts;
```

	acctID	Balance
▶	101	1000
	202	2000
	303	2500
	404	3000
*	NULL	NULL

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:

	Tables_in_my_accounts
▶	accounts
	customers

Βλέπουμε ότι προστέθηκε ο πίνακας CUSTOMERS στην ΒΔ.

Εμφάνιση περιεχομένων του πίνακα CUSTOMERS:

```
select * from CUSTOMERS;
```

Περιεχόμενα του πίνακα CUSTOMERS:

	CUSTNO	CUST_NAME
▶	10	101
	20	202
*	NULL	NULL

Τα περιεχόμενα συμβαδίζουν με αυτά που εισάγαμε με τις εντολές μας

Εμφάνιση δομής του πίνακα CUSTOMERS:

	Field	Type	Null	Key	Default	Extra
▶	CUSTNO	int(11)	NO	PRI	NULL	
	CUST_NAME	varchar(30)	YES		NULL	

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:

	acctID	Balance	Custno
▶	101	1000	10
	202	2000	20
	303	2500	10
	404	3000	10
*	NULL	NULL	NULL

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

Δομή του πίνακα Accounts:

	Field	Type	Null	Key	Default	Extra
▶	acctID	int(11)	NO	PRI	NULL	
	Balance	int(11)	NO		NULL	
	Custno	int(11)	YES	MUL	NULL	

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:

	CUSTNO	count(*)	sum(Balance)
▶	10	3	6500

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:

	count(*)	sum(Balance)
▶	4	8500

```
-- 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 @MY_AVG και υπολογίζουμε με δικό μας τύπο τον μέσο όρο με χρήση των μεταβλητών και τον εμφανίζουμε.

Αποτέλεσμα εκτέλεσης εντολής:

	@COUNT_acctID	@SUM_acctID	@AVG_acctID	@MY_AVG := @SUM_acctID / @COUNT_acctID
▶	4	8500	2125.000000000	2125.000000000

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:

	factorial(4)
▶	24

Το αποτέλεσμα είναι σωστό καθώς 4! = 24.

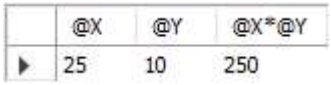
	factorial(15)
▶	2147483647

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:



	@X	@Y	@X*@Y
▶	25	10	250

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:

	t_no	t_user	t_date	t_time	t_proc	t_what
▶	2	root@localhost	2024-02-13	22:46:14	myProc	hello2
	4	root@localhost	2024-02-13	22:46:14	myProc	hello4
	6	root@localhost	2024-02-13	22:46:14	myProc	hello6

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:

	acctID	balance
▶	101	1000
	202	2000
*	NULL	NULL

Μετά:

	acctID	balance
▶	101	900
	202	2100
*	NULL	NULL

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:

@OUT
committed

acctID	balance
101	800
202	2200
NULL	NULL

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

@OUT
rolled back because of missing account 100

acctID	balance
101	800
202	2200
NULL	NULL

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

@OUT
rolled back because of missing account 100

acctID	balance
101	800
202	2200
NULL	NULL

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

@OUT
rolled back because of negative balance of acco...

acctID	balance
101	800
202	2200
NULL	NULL

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:

	@msg
▶	missing account 201

	@msg
▶	missing account 100

	@msg
▶	committed

	@msg
▶	negative balance (?) in 101