

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ

5^η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

ΜΕΤΑΒΛΗΤΕΣ, ΣΥΝΑΡΤΗΣΕΙΣ ΚΑΙ ΔΙΑΔΙΚΑΣΙΕΣ

ΣΤΟΙΧΕΙΑ ΕΡΓΑΣΙΑΣ

ΤΜΗΜΑ ΕΡΓΑΣΤΗΡΙΟΥ: [06] ΤΕΤΑΡΤΗ 13:00-14:00
ΥΠΕΥΘΥΝΗ ΕΡΓΑΣΤΗΡΙΟΥ : ΓΑΡΟΦΑΛΑΚΗ ΡΑΝΙΑ
ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ : 07/02/2024
ΠΡΟΘΕΣΜΙΑ ΥΠΟΒΟΛΗΣ : 13/02/2024

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ

ΦΩΤΟΓΡΑΦΙΑ ΦΟΙΤΗΤΗ:



ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΑΘΑΝΑΣΙΟΥ ΒΑΣΙΛΕΙΟΣ ΕΥΑΓΓΕΛΟΣ

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ : 19390005

ΕΞΑΜΗΝΟ ΦΟΙΤΗΤΗ : 9^ο

ΚΑΤΑΣΤΑΣΗ ΦΟΙΤΗΤΗ : ΠΡΟΠΤΥΧΙΑΚΟ

ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ : ΠΑΔΑ

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

ΠΕΡΙΕΧΟΜΕΝΑ

ΒΔ personnel.....	6
personnel.sql.....	6
personnel.png	8
ΒΔ my_accounts	8
my_accounts.sql	8
my_accounts.png	9
Δραστηριότητες.....	10
1. Συνδεθείτε στην MySQL του συστήματος σας με όποιον από τους προαναφερόμενους τρόπους επιθυμείτε	10
1.2. Στιγμιότυπο	10
2. Ελέγξτε αν υπάρχει ΒΔ με την ονομασία my_accounts.....	10
2.1. Δήλωση.....	10
2.2. Στιγμιότυπο	11
3. Δημιουργήστε τη ΒΔ my_accounts, επιλέξτε την για χρήση και δημιουργήστε και πίνακα με όνομα Accounts με δομή και περιεχόμενα, όπως φαίνονται στις ακόλουθες εντολές Δείξτε το αποτέλεσμα εμφανίζοντας (α) τη λίστα πινάκων της ΒΔ, (β) τα περιεχόμενα και (γ) τη δομή του πίνακα Accounts.....	11
3.1. Δήλωση.....	11
3.2. Στιγμιότυπο	12
4. Εμφανίστε τα περιεχόμενα του πίνακα Accounts, προσθέτοντας αύξουσα αρίθμηση στις εγγραφές του	13
4.1. Δήλωση.....	13
4.2. Στιγμιότυπο	13
5. Η αύξουσα αρίθμηση που εμφανίστηκε στην στήλη με τίτλο No του βήματος 4, θα πρέπει να υπάρχει και στον πίνακα Accounts; Αιτιολογήστε την απάντηση σας.	13
5.1. Δήλωση.....	13
5.2. Στιγμιότυπο	14
5.3. Αιτιολόγηση	14
6. Προσθέστε τον πίνακα CUSTOMERS που εμφανίζεται στην Εικόνα 1. Ορίστε τύπους δεδομένων των στηλών CUSTNO και CUST_NAME ως integer και varchar(30) αντίστοιχα. Δείξτε το αποτέλεσμα εμφανίζοντας (α) τη λίστα πινάκων της ΒΔ, (β) τα περιεχόμενα και (γ) τη δομή του πίνακα CUSTOMERS.	14
6.1. Δήλωση.....	14
6.2. Στιγμιότυπο	15

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

7. Στον πίνακα Accounts προσθέστε στήλη με όνομα Custno, τύπο δεδομένων integer και ορίστε την ως FK του πίνακα Accounts για τη σύνδεση των εγγραφών του με τις εγγραφές του πίνακα CUSTOMERS. Ενημερώστε 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. Δείξτε το αποτέλεσμα εμφανίζοντας (α) τα περιεχόμενα και (β) τη δομή του πίνακα Accounts.	16
7.1. Δήλωση.....	16
7.2. Στιγμιότυπο	17
8. Εκτελέστε και ερμηνεύστε τις ακόλουθες δηλώσεις SQL	17
8.1. Δήλωση & Ερμηνεία	17
8.2. Στιγμιότυπο	18
9. Εκτελέστε και ερμηνεύστε τις ακόλουθες δηλώσεις SQL	18
9.1. Δήλωση & Ερμηνεία	18
9.2. Στιγμιότυπο	19
10. Ορίστε και χρησιμοποιήστε τη συνάρτηση factorial που υπολογίζει το $n!=1*2*\dots*n$	19
10.1. Δήλωση.....	19
10.2. Στιγμιότυπο	21
11. Ορίστε και χρησιμοποιήστε την διαδικασία my_procedure_Local_Variables για υπολογισμούς με χρήση τοπικών μεταβλητών	21
11.1. Δήλωση.....	21
11.2. Στιγμιότυπο	22
12. Ακολουθήστε τα ακόλουθα για τη δημιουργία μιας αποθηκευμένης διαδικασίας και χρήση των commit/rollback. Εξηγείστε τι κάνει η διαδικασία myProc	22
12.1. Δήλωση.....	22
12.2. Στιγμιότυπο	24
12.3. Αιτιολόγηση	24
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.	25
13.1. Δήλωση.....	25
13.2. Στιγμιότυπο	26
14. Ακολουθεί τη λύση του προβλήματος στο βήμα 15 με χρήση της procedure BankTransfer.....	26
14.1. Δήλωση.....	26
14.2. Στιγμιότυπο	29

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

15. Ακολουθεί δεύτερη λύση του προβλήματος	30
15.1. Δήλωση.....	30
15.2. Στιγμιότυπο	34

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

ΒΔ personnel

personnel.sql

```
drop database if exists personnel;
```

```
create database personnel;
```

```
use personnel;
```

```
create table
```

```
DEPT(
```

```
DEPTNO numeric(2),
```

```
DNAME varchar(24),
```

```
LOC char(23)
```

```
);
```

```
insert into
```

```
DEPT
```

```
(DEPTNO, DNAME, LOC)
```

```
values
```

```
(50, 'SALES', 'ATHENS'),
```

```
(60, 'ACCOUNTING', 'ATHENS'),
```

```
(70, 'PAYROL', 'VOLOS');
```

```
create table
```

```
JOB(
```

```
JOB_CODE numeric(3),
```

```
JOB_DESCR varchar(24),
```

```
SAL numeric(10,2)
```

```
);
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
insert into
```

```
JOB
```

```
(JOB_CODE, JOB_DESCR, SAL)
```

```
values
```

```
(100, 'SALESMAN', 2000),
```

```
(200, 'ANALYST', 2000),
```

```
(300, 'DBA', 3000);
```

```
create table
```

```
EMP(
```

```
EMPNO numeric(4),
```

```
NAME varchar(255),
```

```
JOBNO numeric(3),
```

```
DEPTNO numeric(2),
```

```
COMM numeric(10,2)
```

```
);
```

```
insert into
```

```
EMP
```

```
(EMPNO, NAME, JOBNO, DEPTNO, COMM)
```

```
values
```

```
(10, 'Codd', 100, 50, NULL),
```

```
(20, 'Navathe', 200, 50, 450),
```

```
(30, 'Elmasri', 300, 60, NULL),
```

```
(40, 'Date', 100, 50, NULL);
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

personnel.png

Στήλες	Τύπος δεδομένων
DEPT.DEPTNO, EMP.DEPTNO	numeric(2)
DNAME, JOB_DESCR	varchar(24)
LOC	char(23)
JOBCODE, JOBNO	numeric(3)
SAL, COMM	numeric(10,2)
EMPNO	numeric(4)
PROJECT.P_ID	int
PROJECT.P_NAME	varchar(255)

Πίνακας 1. Τύποι δεδομένων πινάκων EMP, JOB, DEPT

EMP

EMPNO	NAME	JOBNO	DEPTNO	COMM
10	CODD	100	50	
20	NAVATHE	200	50	450
30	ELMASRI	300	60	
40	DATE	100	50	

JOB

JOBCODE	JOB_DESCR	SAL
100	SALESMAN	2000
200	ANALYST	2000
300	DBA	3000

DEPT

DEPTNO	DNAME	LOC
50	SALES	ATHENS
60	ACCOUNTING	ATHENS
70	PAYROL	VOLOS

ΒΔ my_accounts

my_accounts.sql

```
DROP DATABASE IF EXISTS my_accounts;
```

```
CREATE DATABASE my_accounts;
```

```
USE my_accounts;
```

```
CREATE TABLE Accounts (acctID int not null primary key,  
                          Balance int not null);
```

```
INSERT INTO Accounts (acctID, Balance) VALUES (101, 1000);
```


ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
INSERT INTO Accounts (acctID, Balance) VALUES (202, 2000);
```

```
INSERT INTO Accounts (acctID, Balance) VALUES (303, 2500);
```

```
INSERT INTO Accounts (acctID, Balance) VALUES (404, 3000);
```

```
CREATE TABLE Customers (custno int not null, cust_name varchar(30), primary  
key(custno));
```

```
INSERT INTO Customers (custno, cust_name) VALUES (10, '101');
```

```
INSERT INTO Customers (custno, cust_name) VALUES (20, '202');
```

my_accounts.png

Accounts

acctID	Balance
101	1000
202	2000
303	2500
404	3000

CUSTOMERS

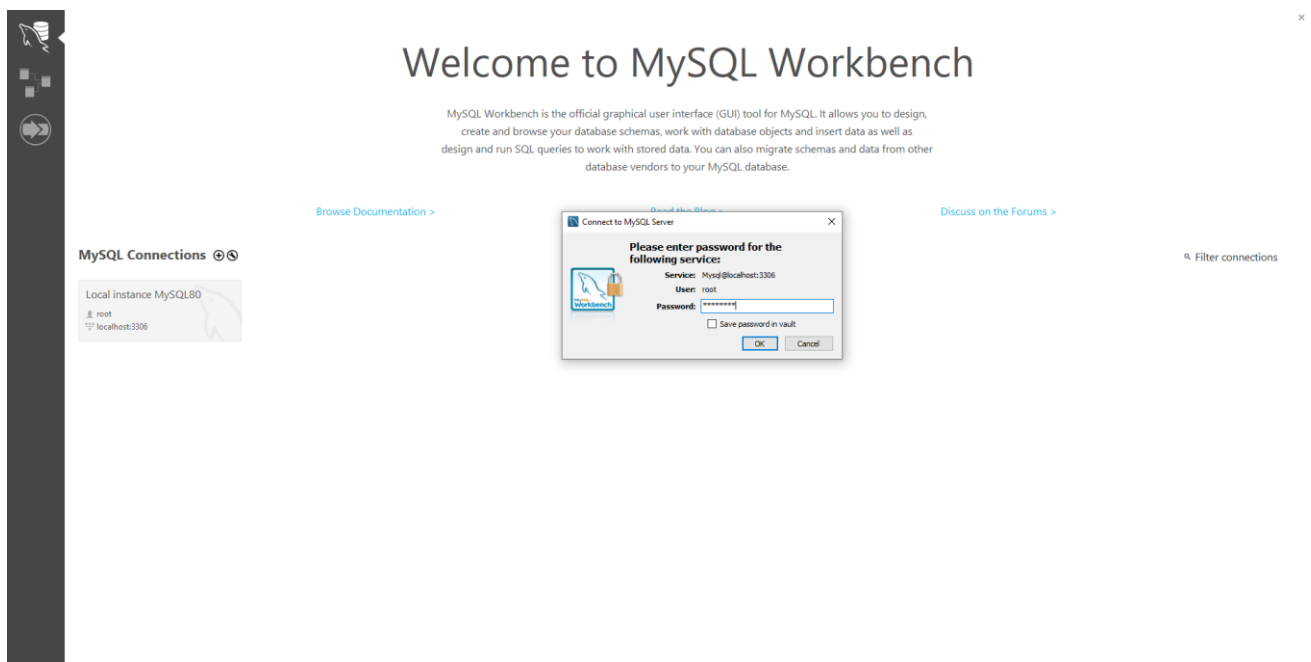
CUSTNO	CUST_NAME
10	101
20	202

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

Δραστηριότητες

1. Συνδεθείτε στην MySQL του συστήματός σας με όποιον από τους προαναφερόμενους τρόπους επιθυμείτε

1.2. Στιγμιότυπο



2. Ελέγξτε αν υπάρχει ΒΔ με την ονομασία my_accounts

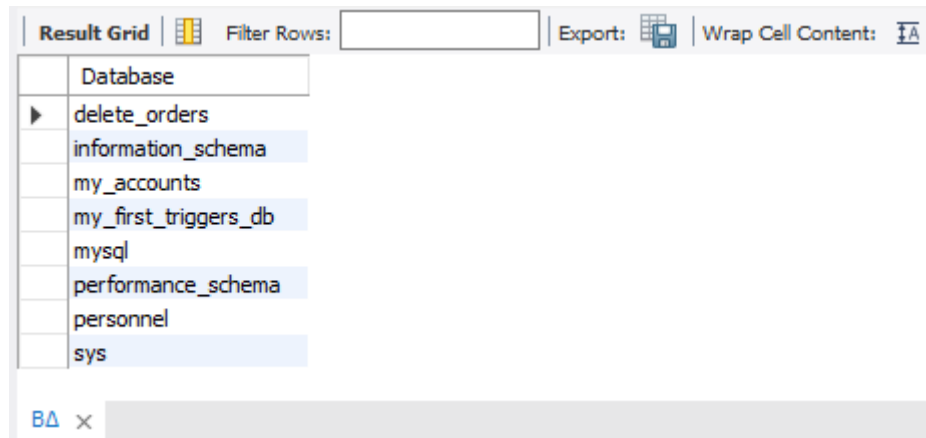
2.1. Δήλωση

Εμφάνιση όλων των ΒΔ

SHOW databases;

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

2.2. Στιγμιότυπο



3. Δημιουργήστε τη ΒΔ my_accounts, επιλέξτε την για χρήση και δημιουργήστε και πίνακα με όνομα Accounts με δομή και περιεχόμενα, όπως φαίνονται στις ακόλουθες εντολές Δείξτε το αποτέλεσμα εμφανίζοντας (α) τη λίστα πινάκων της ΒΔ, (β) τα περιεχόμενα και (γ) τη δομή του πίνακα Accounts

3.1. Δήλωση

Δημιουργία της ΒΔ my_accounts και του πίνακα Accounts

```
DROP DATABASE IF EXISTS my_accounts;
```

```
CREATE DATABASE my_accounts;
```

```
USE my_accounts;
```

```
CREATE TABLE Accounts (acctID int not null primary key,  
                        Balance int not null);
```

Δοκιμές

```
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);
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

(α) Λίστα πινάκων της ΒΔ

```
SHOW tables;
```

(β) Περιεχόμενα του πίνακα Accounts

```
SELECT * FROM Accounts;
```

(γ) Δομή του πίνακα Accounts

```
DESCRIBE Accounts;
```

3.2. Στιγμιότυπο

The screenshot displays a database management interface with two panels. The top panel shows the 'Tables_in_my_accounts' list with 'accounts' selected. The bottom panel is split into two views: a data view on the left and a structure view on the right.

Data View (Left): Shows the contents of the 'accounts' table. The columns are 'acctID' and 'Balance'. The data rows are:

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

Structure View (Right): Shows the schema of the 'accounts' table. The columns are 'Field', 'Type', 'Null', 'Key', 'Default', and 'Extra'. The structure is:

Field	Type	Null	Key	Default	Extra
acctID	int	NO	PRI	NULL	
Balance	int	NO		NULL	

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

4. Εμφανίστε τα περιεχόμενα του πίνακα Accounts, προσθέτοντας αύξουσα αρίθμηση στις εγγραφές του

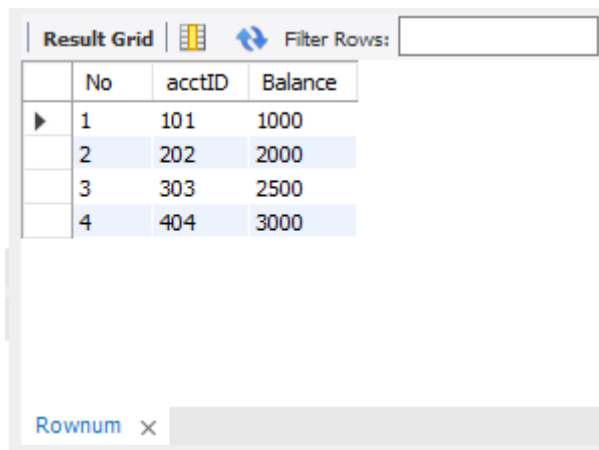
4.1. Δήλωση

Τα περιεχόμενα του πίνακα Accounts με αύξουσα αρίθμηση στις εγγραφές του

```
SET @rownum = 0;
```

```
SELECT (@rownum := @rownum + 1) AS No, acctID, Balance FROM Accounts ORDER BY acctID;
```

4.2. Στιγμιότυπο



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

5. Η αύξουσα αρίθμηση που εμφανίστηκε στην στήλη με τίτλο No του [βήματος 4](#), θα πρέπει να υπάρχει και στον πίνακα Accounts; Αιτιολογήστε την απάντηση σας.

5.1. Δήλωση

Τα περιεχόμενα του πίνακα Accounts με αύξουσα αρίθμηση στις εγγραφές του

```
SET @rownum = 0;
```

```
SELECT (@rownum := @rownum + 1) AS No, acctID, Balance FROM Accounts ORDER BY acctID;
```

```
DESCRIBE Accounts;
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

5.2. Στιγμιότυπο

Result Grid | Filter Rows: | Export:

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

Rownum x Accounts

Result Grid | Filter Rows: | Export:

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

Rownum Accounts x

5.3. Αιτιολόγηση

Η αύξουσα αρίθμηση που εμφανίστηκε στην στήλη με τίτλο No του βήματος 4, δεν θα πρέπει να υπάρχει και στον πίνακα Accounts, καθώς, η εντολή «SELECT» τυπώνει αποτελέσματα με βάση κάποια ορίσματα και συνεπώς, δεν τις αποθηκεύει κάπου. Η εντολή «CREATE» και «ALTER», αντίθετα, προσθέτουν στήλες σε πίνακες.

6. Προσθέστε τον πίνακα CUSTOMERS που εμφανίζεται στην [Εικόνα 1](#). Ορίστε τύπους δεδομένων των στηλών CUSTNO και CUST_NAME ως integer και varchar(30) αντίστοιχα. Δείξτε το αποτέλεσμα εμφανίζοντας (α) τη λίστα πινάκων της ΒΔ, (β) τα περιεχόμενα και (γ) τη δομή του πίνακα CUSTOMERS.

6.1. Δήλωση

Δημιουργία του πίνακα Customers στη ΒΔ my_accounts

USE my_accounts;

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
CREATE TABLE Customers (custno int not null, cust_name varchar(30), primary  
key(custno));
```

Δοκιμές

```
INSERT INTO Customers (custno, cust_name) VALUES (10, '101');
```

```
INSERT INTO Customers (custno, cust_name) VALUES (20, '202');
```

(α) Λίστα πινάκων της ΒΔ

```
SHOW tables;
```

(β) Περιεχόμενα του πίνακα Customers

```
SELECT * FROM Customers;
```

(γ) Δομή του πίνακα Customers

```
DESCRIBE Customers;
```

6.2. Στιγμιότυπο

The image displays three screenshots of a database management interface, likely MySQL Workbench, arranged in a grid. The top-left screenshot shows the 'Tables in my_accounts' list with 'accounts' and 'customers' visible. The top-right screenshot shows the 'Result Grid' for the 'customers' table, displaying two rows of data: (10, '101') and (20, '202'), plus a row for NULL values. The bottom screenshot shows the 'Result Grid' for the 'DESCRIBE Customers;' query, displaying the table structure with columns: Field, Type, Null, Key, Default, and Extra. The structure shows 'custno' as an integer, not null, primary key, and 'cust_name' as a varchar(30), nullable.

Field	Type	Null	Key	Default	Extra
custno	int	NO	PRI	NULL	
cust_name	varchar(30)	YES		NULL	

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

7. Στον πίνακα **Accounts** προσθέστε στήλη με όνομα **Custno**, τύπο δεδομένων **integer** και ορίστε την ως FK του πίνακα **Accounts** για τη σύνδεση των εγγραφών του με τις εγγραφές του πίνακα **CUSTOMERS**. Ενημερώστε 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**. Δείξτε το αποτέλεσμα εμφανίζοντας (α) τα περιεχόμενα και (β) τη δομή του πίνακα **Accounts**.

7.1. Δήλωση

Προσθήκη foreign key custno στον πίνακα Accounts

```
USE my_accounts;
```

```
ALTER TABLE Accounts ADD custno int;
```

```
ALTER TABLE Accounts ADD foreign key(custno) references Customers(custno);
```

Δοκιμές

```
UPDATE Accounts SET custno = 20 WHERE acctID = 202;
```

```
UPDATE Accounts SET custno = 10 WHERE acctID <> 202;
```

(α) Περιεχόμενα του πίνακα Accounts

```
SELECT * FROM Accounts;
```

(β) Δομή του πίνακα Accounts

```
DESCRIBE Accounts;
```


ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

7.2. Στιγμιότυπο

Result Grid				Result Grid						
Filter Rows:				Filter Rows: Export:						
	acctID	Balance	custno		Field	Type	Null	Key	Default	Extra
▶	101	1000	10	▶	acctID	int	NO	PRI	NULL	
	202	2000	20		Balance	int	NO		NULL	
	303	2500	10		custno	int	YES	MUL	NULL	
	404	3000	10							
*	NULL	NULL	NULL							

(α) × (β)

(α) (β) ×

8. Εκτελέστε και ερμηνεύστε τις ακόλουθες δηλώσεις SQL

8.1. Δήλωση & Ερμηνεία

Χρήση της ΒΔ my_accounts

```
USE my_accounts;
```

(α) Εμφάνισε το id, το πλήθος και το άθροισμα των υπολοίπων των πελατών

που δεν έχουν id ίσο με 20. Εμφάνισε τα αποτελέσματα ανά πελάτη

```
SELECT custno, count(*), sum(balance)
```

```
FROM Accounts
```

```
WHERE custno NOT IN (20)
```

```
GROUP BY custno;
```

```
SET @CUST_NO = 20;
```

(β) Εμφάνισε το id, το πλήθος και το άθροισμα των υπολοίπων των πελατών

που δεν έχουν id ίσο με το περιεχόμενο της μεταβλητής CUST_NO.

Εμφάνισε τα αποτελέσματα ανά πελάτη

```
SELECT custno, count(*), sum(balance)
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
FROM Accounts  
  
WHERE custno NOT IN (@CUST_NO)  
  
GROUP BY custno;
```

8.2. Στιγμιότυπο

	custno	count(*)	sum(balance)
▶	10	3	6500

9. Εκτελέστε και ερμηνεύστε τις ακόλουθες δηλώσεις SQL

9.1. Δήλωση & Ερμηνεία

Χρήση της ΒΔ my_accounts

```
USE my_accounts;
```

(α) Εμφάνισε το πλήθος και το άθροισμα των υπολοίπων των πελατών

```
SELECT count(*), sum(balance) FROM Accounts;
```

```
SET @COUNT_acctID = 0, @SUM_acctID = 0, @AVG_acctID = 0;
```

Αποθήκευσε το πλήθος των πελατών στην μεταβλητή COUNT_acctID

Αποθήκευσε το άθροισμα των υπολοίπων των πελατών στην μεταβλητή SUM_acctID

Αποθήκευσε τον μέσο όρο των υπολοίπων των πελατών στην μεταβλητή AVG_acctID

```
SELECT count(*), sum(balance), avg(balance)
```

```
INTO @COUNT_acctId, @SUM_acctID, @AVG_acctID
```

```
FROM Accounts;
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
# (β) Εμφάνισε το περιεχόμενο της μεταβλητής COUNT_acctID,  
# το περιεχόμενο της μεταβλητής SUM_acctID,  
# το περιεχόμενο της μεταβλητής AVG_acctID,  
# το περιεχόμενο της μεταβλητής MY_AVG, όπου περιέχει το αποτέλεσμα  
# της διαίρεσης SUM_acctID/COUNT_acctID  
  
SELECT @COUNT_acctID, @SUM_acctID, @AVG_acctID, @MY_AVG :=  
@SUM_acctID/@COUNT_acctID;
```

9.2. Στιγμιότυπο

Result Grid

	count(*)	sum(balance)
▶	4	8500

(α) × (β)

Result Grid

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

(α) (β) ×

10. Ορίστε και χρησιμοποιήστε τη συνάρτηση factorial που υπολογίζει το $n! = 1 * 2 * \dots * n$

10.1. Δήλωση

```
DROP FUNCTION IF EXISTS factorial;  
  
DELIMITER !  
  
CREATE FUNCTION factorial(N int)  
  
RETURNS int
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

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 ;

(α) Εμφάνισε το αποτέλεσμα του 4!

SELECT factorial(4);

(β) Εμφάνισε το αποτέλεσμα του 15!

SELECT factorial(15);

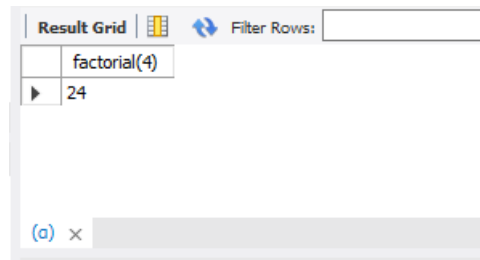
Error Code: 1264. Out of range value for column 'F' at row 1

Το αποτέλεσμα του παραγοντικού 15! ξεπερνάει την μέγιστη τιμή

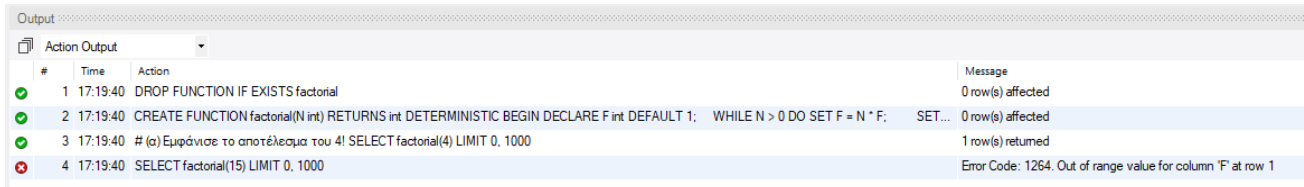
που μπορεί να αποθηκευτεί σε τύπου int

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

10.2. Στιγμιότυπο



	factorial(4)
▶	24



#	Time	Action	Message
1	17:19:40	DROP FUNCTION IF EXISTS factorial	0 row(s) affected
2	17:19:40	CREATE FUNCTION factorial(N int) RETURNS int DETERMINISTIC BEGIN DECLARE F int DEFAULT 1; WHILE N > 0 DO SET F = N * F; SET...	0 row(s) affected
3	17:19:40	# (α) Εμφάνισε το αποτέλεσμα του 4! SELECT factorial(4) LIMIT 0, 1000	1 row(s) returned
4	17:19:40	SELECT factorial(15) LIMIT 0, 1000	Error Code: 1264. Out of range value for column 'F' at row 1

11. Ορίστε και χρησιμοποιήστε την διαδικασία my_procedure_Local_Variables για υπολογισμούς με χρήση τοπικών μεταβλητών

11.1. Δήλωση

```
DROP PROCEDURE IF EXISTS my_procedure_Local_Variables;

DELIMITER $$

CREATE PROCEDURE my_procedure_Local_Variables(IN x int, IN y int)
BEGIN
    SET @X = x;
    SET @Y = y;
    SELECT @X, @Y, @X*@Y;
END $$

DELIMITER ;
```

```
# (α) Υπολογισμός 25*10
CALL my_procedure_Local_Variables(25, 10);

# (β) Υπολογισμός 50*10
CALL my_procedure_Local_Variables(50, 10);
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

(γ) Υπολογισμός $3*5$

```
CALL my_procedure_Local_Variables(3, 5);
```

11.2. Στιγμιότυπο

Result Grid

Filter Rows:

	@X	@Y	@X*@Y
▶	25	10	250

(a) × (b) (c)

Result Grid

Filter Rows:

	@X	@Y	@X*@Y
▶	50	10	500

(a) (b) × (c)

Result Grid

Filter Rows:

	@X	@Y	@X*@Y
▶	3	5	15

(a) (b) (c) ×

12. Ακολουθήστε τα ακόλουθα για τη δημιουργία μιας αποθηκευμένης διαδικασίας και χρήση των commit/rollback. Εξηγείστε τι κάνει η διαδικασία myProc

12.1. Δήλωση

Δοκιμές χρήσης της συνάρτησης MOD

```
SET @p_no = 3;
```

(α) Εμφάνισε το αποτέλεσμα της πράξης $3 \bmod 2$

```
SELECT MOD(@p_no, 2);
```

```
SET @p_no = 8;
```

(β) Εμφάνισε το αποτέλεσμα της πράξης $8 \bmod 2$

```
SELECT MOD(@p_no, 2);
```

Δημιουργία βάσης και πίνακα

```
DROP DATABASE IF EXISTS trace;
```

```
CREATE DATABASE trace;
```

```
USE trace;
```

```
DROP TABLE IF EXISTS myTrace;
```

```
CREATE TABLE myTrace (t_no INT,
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
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);
```

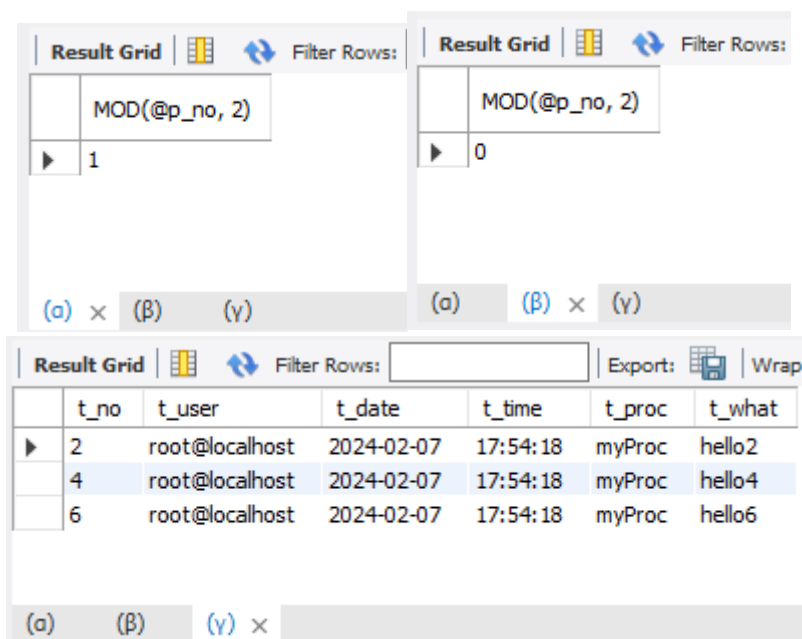
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
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);
```

(γ) Εμφάνισε τα περιεχόμενα του πίνακα myTrace

```
SELECT * FROM myTrace;
```

12.2. Στιγμιότυπο



	t_no	t_user	t_date	t_time	t_proc	t_what
▶	2	root@localhost	2024-02-07	17:54:18	myProc	hello2
	4	root@localhost	2024-02-07	17:54:18	myProc	hello4
	6	root@localhost	2024-02-07	17:54:18	myProc	hello6

12.3. Αιτιολόγηση

Η διαδικασία myProc παίρνει για είσοδο το id (p_no) και το μήνυμα (p_in) της ιχνηλάτησης (εγγραφή του πίνακα myTrace της ΒΔ trace) και παράγει για έξοδο το μήνυμα της ιχνηλάτησης (p_out). Αφού, εκχωρήσει το μήνυμα της ιχνηλάτησης στην έξοδο σε μία μεταβλητή (SET p_out = p_in), η διαδικασία πραγματοποιεί μία εγγραφή INSERT στον πίνακα myTrace με δεδομένα τα εξής:

- Το id της ιχνηλάτησης (p_no) στο πεδίο t_no
- Τον τρέχον χρήστη (current_user) στο πεδίο t_user

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

- Την τρέχουσα ημερομηνία (current_date) στο πεδίο t_date
- Την τρέχουσα ώρα (current_time) στο πεδίο t_time
- Το όνομα της διαδικασίας ('myProc') στο πεδίο t_proc
- Το μήνυμα ιχνηλάτησης (p_in) στο πεδίο t_what

Στην συνέχεια, πραγματοποιεί έναν έλεγχο όσον αφορά τα id (p_no) των ιχνηλατήσεων που έχουν εγγραφεί με επιτυχία στον πίνακα myTrace. Πιο συγκεκριμένα, με την ρουτίνα MOD εκτελείται η πράξη $p_no \bmod 2$ και εφόσον, το αποτέλεσμα είναι ίσο με 0, δηλαδή, πρόκειται για άρτιο id ιχνηλάτησης, τότε εκτελείται η λειτουργία COMMIT, διαφορετικά εκτελείται η λειτουργία ROLLBACK. Διεξοδικά, αν πρόκειται για άρτιο id ιχνηλάτησης, τότε αποθηκεύεται η αλλαγή που έγινε στον πίνακα myTrace (COMMIT), διαφορετικά αν πρόκειται για περιττό id ιχνηλάτησης η αλλαγή που έγινε στον πίνακα myTrace αναιρείται.

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.

13.1. Δήλωση

Δημιουργία πίνακα Accounts με 2 εγγραφές

```
USE my_accounts;
```

```
DROP TABLE IF EXISTS Accounts;
```

```
CREATE TABLE Accounts (acctID int not null primary key,  
    balance int not null,  
    CONSTRAINT unloanable_account CHECK (balance >= 0));
```

Δοκιμές

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

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

```
COMMIT;
```

(α) Εμφάνισε τα περιεχόμενα του πίνακα Accounts πριν την συναλλαγή

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
SELECT * FROM Accounts;
```

Συναλλαγή

```
BEGIN;
```

```
UPDATE Accounts SET balance = balance - 100
```

```
WHERE acctID = 101;
```

```
UPDATE Accounts SET balance = balance + 100
```

```
WHERE acctID = 202;
```



```
COMMIT;
```

(β) Εμφάνισε τα περιεχόμενα του πίνακα Accounts μετά την συναλλαγή

```
SELECT * FROM Accounts;
```

13.2. Στιγμιότυπο



Result Grid



Filter Rows:

	acctID	balance
▶	101	1000
	202	2000
*	NULL	NULL

(α) × (β)

Result Grid



Filter Rows:

	acctID	balance
▶	101	900
	202	2100
*	NULL	NULL

(α) (β) ×

14. Ακολουθεί τη λύση του προβλήματος στο [βήμα 15](#) με χρήση της procedure **BankTransfer**

14.1. Δήλωση

Δημιουργία procedure BankTransfer

```
USE my_accounts;
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

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 row_s INT;

DECLARE newbalance INT;

SELECT count(*) INTO row_s FROM Accounts WHERE acctID = fromAcct;

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

SELECT balance INTO newbalance FROM Accounts WHERE acctID = fromAcct;

IF row_s = 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 row_s FROM Accounts WHERE acctID = toAcct;

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

IF row_s = 0 THEN

ROLLBACK;

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

ELSE

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
        COMMIT;

        SET msg = 'committed';

    END IF;

END IF;

END P1 //

DELIMITER ;
```

(α) 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;
```

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

```
SET AUTOCOMMIT = 0;

SET @out = ' ';

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

SELECT @out;

SELECT * FROM Accounts;

COMMIT;
```

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

```
SET AUTOCOMMIT = 0;

SET @out = ' ';

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

SELECT @out;
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
SELECT * FROM Accounts;
```

```
COMMIT;
```

```
# (δ) 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;
```

14.2. Στιγμιότυπο

The screenshot displays three sequential result grids from a database management tool, illustrating a transaction rollback scenario.

Result Grid 1: Shows the value of the variable `@out` as `committed`.

Result Grid 2: Shows the state of the `Accounts` table. The table has two columns: `acctID` and `balance`. The data is as follows:

acctID	balance
101	800
202	2200
*	NULL

Result Grid 3: Shows the value of the variable `@out` as `rolled back because of missing account 201`.

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

Result Grid	Filter Rows:	Edit:
acctID	balance	
101	800	
202	2200	
NULL	NULL	

(a) (a) (β) (β) × (γ) (γ) (δ) (δ)

Result Grid	Filter Rows:	Export:
@out		
rolled back because of missing account 100		

(a) (a) (β) (β) (γ) × (γ) (δ) (δ)

Result Grid	Filter Rows:	Edit:
acctID	balance	
101	800	
202	2200	
NULL	NULL	

(a) (a) (β) (β) (γ) (γ) × (δ) (δ)

15. Ακολουθεί δεύτερη λύση του προβλήματος

15.1. Δήλωση

Δημιουργία πίνακα Accounts

```
USE my_accounts;
```

```
DROP TABLE IF EXISTS Accounts;
```

```
CREATE TABLE Accounts (acctID int not null primary key,  
    balance int not null);
```

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

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

```
COMMIT;
```

(α) Εμφάνισε τα περιεχόμενα του πίνακα Accounts

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
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 ;
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

Δημιουργία procedure BankTransfer

DELIMITER !

DROP PROCEDURE IF EXISTS BankTransfer;

```
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

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

```
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 ;
```

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

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

```
SELECT @msg;
```

```
# (γ) Δοκιμή μεταφοράς 100 από acctID = 100 (ανύπαρκτος) σε acctID = 202
```

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

```
SELECT @msg;
```

```
# (δ) Δοκιμή μεταφοράς 100 από acctID = 101 σε acctID = 202
```

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

```
SELECT @msg;
```

```
# (ε) Δοκιμή μεταφοράς 2000 από acctID = 101 (ανεπαρκές) σε acctID = 202
```

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

```
SELECT @msg;
```

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

15.2. Στιγμιότυπο

Result Grid

Filter Rows:

	acctID	balance
▶	101	1000
	202	2000
*	NULL	NULL

(α) × (β) (γ) (δ) (ε)

Result Grid

Filter Rows:

	@msg
▶	missing account 201

(α) (β) × (γ) (δ) (ε)

Result Grid

Filter Rows:

	@msg
▶	missing account 100

(α) (β) (γ) × (δ) (ε)

Result Grid

Filter Rows:

	@msg
▶	committed

(α) (β) (γ) (δ) × (ε)

Result Grid

Filter Rows:

	@msg
▶	negative balance (?) in 101

(α) (β) (γ) (δ) (ε) ×

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II



Σας ευχαριστώ για την προσοχή σας.

