Εισαγωγή – Απαιτήσεις Βασης Δεδομένων

Θα δημιουργήσουμε μια βάση δεδομένων για χρήση σε ανθοπωλείο. Η βάση δεδομένων θα πρέπει να δύναται να αποθηκεύσει τον κατάλογο των προιόντων του ανθοπωλείου με πλήρεις περιγραφές και τιμή για κάθε προιόν, κατάλογο προμηθευτών με πληροφορίες όπως ΑΦΜ, Ονομα προμηθευτή/Επειχήρησης, Τηλέφωνα (οπωσδήπωτε 1 κινητό και 1 σταθερό, με δυνατότητες για περισσότερα, και διεύθυνση. Ακόμα κατάλογο μελών, Με κωδικό μέλους, ονοματεπώνυμο, τηλέφωνα επαφής (και παλι τουλάχιστον 1 κινητό, με δυνατότητα για μεχρι 2 τηλέφωνα ακόμα), Διεύθυνση, Αριθμούς Πιστωτικών Καρτών (μεχρι 2 διαφορετικές, προεραιτικά), και E-mail και κωδικό πρόσβασης για χρήση μέσω Διαδυκτίου. Θα ύπαρχει κατάλογος με τους εργαζόμενους στην επειχήρηση που θα αποθηκεύει ονοματεπώνυμο, εργασία [ένα εκ των CEO (ανώτατος διαχειρηστής, με δυνατότητα να ελέγχει και να διαγράφει δεδομένα ελεύθερα), Manager (Θα έχει την δυνατότητα να πραγματοποιεί όλες τις λειτουγρείες που απαιτούνται απο την λειτουργία του καταστήματος, αλλα όχι να διαργάφει δεδομένα), Seller (με περιορισμένες δυνατότητες, θα μπορεί μόνο να παρακολουθεί και να κάνει πωλήσεις, οπως και κα βλέπει τον κατάλογο), Driver (χωρίς δικαιώματα πρόσβασης)] και μισθό. Θα μπορεί να αποθηκεύει παραγγελίες με πελάτη που πραγματοποίησε την παραγγελία (μόνο μέλη), το πόσο θα πληρώσει (αποφασίζεται απο τον Manager), τι θα περιέχει, ημερομηνία, ώρα και διεύθυνση παράδωσης, και τέλος χώρο για σχόλια ή αφιέρωση. Τέλος, θα διατηρεί στην μνήμη της όλες τις συναλαγές που πραγματοποιήθηκαν (αγορές και πωλήσεις) με ακριβή ημερομηνία και ώρα, και τον πωλητη (σε περίπτωση αγοράς) η τον πελάτη (σε περίπτωση πώλησης), και, φυσικά το ποσό της συναλλαγής.

Ακόμα, θα πρέπει να μπορεί να εμφανίσει τον κατάλογο των προιόντων, τις παραγγελίες (όλες, προς παραδοση μεσα στην εβδομάδα, και προς παράδωση σήμερα), το πελατολόγιο, μαζι με το πόσα χρήματα έχει ξοδέψει ο πελάτης, τις ημερίσιες πωλήσεις, τις ημερήσιες συναλαγές, τα προιόντα σε κάθε παραγγελία, και τέλος τα προιόντα στην αποθήκη. Θα πρέπει επίσης κάποιος που δεν έχει γνώσεις απο βάσεις δεδομένων (οπως για παράδειγμα πωλητές ή ακόμα και Managers) να μπορούν να προσθέτουν προιόντα, μέλη και παραλλελίες, να καταγράφουν αγορές και πωλήσεις, και να προσλαμβάνουν υπαλλήλους. Τέλος, επειδή θα απαιτείται συχνά, η βάση δεδομένων θα πρέπει να μπορει να επιστρέφει τι ποσότητα απαιτειται αγοραστει απο κάποιο αντικείμενο για να ικανοποιήθούν οι παραγγελίες (μέχρι μια συγγεκριμένη ημερομηνία), και επείσης να δίνει τον ισολογισμό της επειχήρησης απο μία συγκεκριμένη ημερομηνία και μετά.

Αναλυση Απαιτήσεων – Δημιουργία Οντοτήτων

BUYERS: <u>TSN</u>, CODE, NAME, TEL1, CELL1, TEL2, CELL2, CELL3, ADDRESS

Με το TSN (ΑΦΜ) ως κλειδί, ο πίνακας προμηθευτών Μπορεί να αποθηκεύει μέχρι 5 τηλεφωνα. Ως υποψήφιο κλειδί, το CODE (εσωτερικός κωδικός προμηθευτή) θα είναι unique. Επιλέχθηκε τελικά το TSN ως κλειδί, αλλα το CODE πρεπει να υπαρχει, ετσι ώστε να εξασφαλίζεται ευκολότερα η ακαιραιώτητα της ΒΔ (ειναι πιο “ευκολο” να θυμαται ενας υπαλληλος απλους σειριακους αριθμους παρα τα ΑΦΜ καθε προμηθευτή). Τα NAME, TEL1, CELL1, ADDRESS και το CODE θα είναι not null (υποχρεωτικά πεδία)

ITEMS: <u>KEYCODE</u>, FULL NAME, PRICE, DESCRIPTION, QUANTITY

Ο πίνακας καταλόγου (αντικειμένων) που υπάρχει στο κατάστημα. Το <u>KEYCODE</u> είναι το κλειδί. Όλα ειναι not null (υποχρεωτικά πεδία).

MEMBERS: <u>CODE</u>, FIRST NAME, LAST NAME, TEL1, CELL1, CELL2, CREDIT CARD 1, CREDIT CARD 2, EMAIL, PASS, ADDRESS

Ο πίνακας των πελατών-μελών του καταστήματος. Περιλαμβάνει τα δεδομένα που αναλύθηκαν πιο πάνω, και κλειδί είναι το <u>CODE</u>.Τα FIRST NAME, LAST NAME, CELL1 και ADDRESS ειναι υποχρεωτικά πεδία.

ORDERS: SN, PAYER, PRICE, DELIVERY DATE, DELIVERY ADDRESS, COMMENTS-CARD

Ο πίνακας που αποθηκεύει τις παραγγελίες, όπως περιγράφηκε πιο πανω. Κλειδί είναι το SN, ενώ το PAYER είναι ο κωδικός μέλους (ξένο κλειδί).Τα PAYER, PRICE, DELIVERY DATE και DELIVERY ADDRESS είναι υποχρεωτικά πεδία.

-- Ο πίνακας διασπάστητε λόγω κανονικοποίησης, δημιουργώντας και τον αμέσως επόμενο:

ORDER STUFF: SN, ITEM, QUANTITY

Ο πίνακας αυτός δημειουργήθηκε απο διάσπαση, και έτσι δανείζεται ως κλείδί τα γνωρίσματα SN (απο τον πίνακα παραγγελιων) και ITEM (απο τον πίνακα καταλόγου). Πρέπει να σημειωθεί εδώ οτι, παρόλο που αρχικά φαίνεται ο πίνακας αυτός να συνδέει τους πίνακες ORDERS και ITEMS, δεν είναι έτσι, αλλα αποτελεί οντότητα. Όλα είναι υποχρεωτικά πεδία.

TRANSACTIONS: TRANSCODE, DATE, TIME, TYPE, BUYER, SELLER, AMOUNT

Ο Πίνακας συναλαγών. 'Οπως περιγράφηκε πιο πάνω. Το TRANSCODE είναι κλειδί, και τα πεδία BUYER και SELLER είναι ξένα κλειδιά απο τους πίνακες MEMBER και BUYER αντίστοιχα. Όλα είναι υποχρεωτικά πεδία.

WORKERS: WORKER NUM, ID_NUM, FIRST NAME, LAST NAME, SALARY, JOB DESC

Ο πίνακας εργαζομένων. Κλειδί είναι το WORKER NUM, που πρωτημήθηκε απο το ID_NUM (Αριθμός ταυτότητας) για 2 λόγους: Πρώτον, ο αριθμός ταυτότητας ενος ατομου μπορεί να αλλάξει, αν αυτός αλλάξει ταυότητα, και δεύτερον, ο Αριθμός Υπαλλήλου είναι σειριακός, διευκολύνοντας έτσι τις αναζητήσεις στον πίνακα. Όλα είναι υποχρεωτικά πεδία

Ακόμα, προκειμένου να υπάρχει συνοχή στη Βάση Δεδομένων, κρίθηκε απαραίτητη η δημιουργία των παρακάτω πινάκων:

JOB DESCRIPTIONS: Περιέχει τις εγρασίες ενός υπαλήλου, και τα δεδομένα του είναι ξενο κλειδι στον πίνακα WORKERS στο πεδίο JOB DESC. Αυτο γίνεται για να μην μπορει να προστεθεί υπαλληλος χωρίς εργασία, ή με άσχετη εργασία (οχι μια απο αυτες ου αναφέρθηκαν πιο πάνω)

KEYS: Αποθηκεύει τον αριθμο που θα χρησημοποιηθει ωσ αυτόματο κλειδί απο τις διαδικασίες για προσθήκη δεδομένων στους πίνακες.

TRANS_TYPE: Περιέχει 2 μόνο πλειάδες, “BUY” και “SELL” για το πεδίο TYPE του πίνακα TRANSACTIONS, προκειμένου να μήν μπαίνουν “ασχετες” πλειάδες και δημειουργηθεί πρόβλημα στις Όψεις και στις Διαδικασίες που χρησιμοποιούν τον πίνακα.

Τυποι Δεδομένων στους πινακες

BUYERS:

TSN, CHAR 12, NOT NULL

CODE, INTEGER, NOT NULL (DOMAIN: MEMBER_INT_KEY)

NAME, CHAR 25, NOT NULL

TEL1, CHAR 10 (DOMAIN: TEL_NUM), NOT NULL

CELL1, CHAR 10 (DOMAIN: TEL_NUM), NOT NULL

TEL2, CHAR 10 (DOMAIN: TEL_NUM)

CELL2, CHAR 10 (DOMAIN: TEL_NUM)

CELL3, CHAR 10 (DOMAIN: TEL_NUM)

ADDRESS, VARCHAR 25, NOT NULL

ITEMS:

KEYCODE, INTEGER, NOT NULL (DOMAIN: ITEM_INT_KEY)

FULL NAME, VARCHAR 25, NOT NULL

PRICE, DECIMAL 3, 2, NOT NULL

DESCRIPTION, VARCHAR 80, NOT NULL

QUANTITY, INTEGER, NOT NULL

MEMBERS:

CODE, INTEGER, NOT NULL (DOMAIN: MEMBER_INT_KEY)

FIRST NAME, CHAR 15, NOT NULL

LAST NAME, CHAR 15, NOT NULL

TEL1, CHAR 10 (DOMAIN: TEL_NUM)

CELL1, CHAR 10 (DOMAIN: TEL_NUM), NOT NULL

CELL2, CHAR 10 (DOMAIN: TEL_NUM)

CREDIT CARD 1, CHAR 16

CREDIT CARD 2, CHAR 16

EMAIL, VARCHAR 25

PASS, CHAR 12

ADDRESS, VARCHAR 50

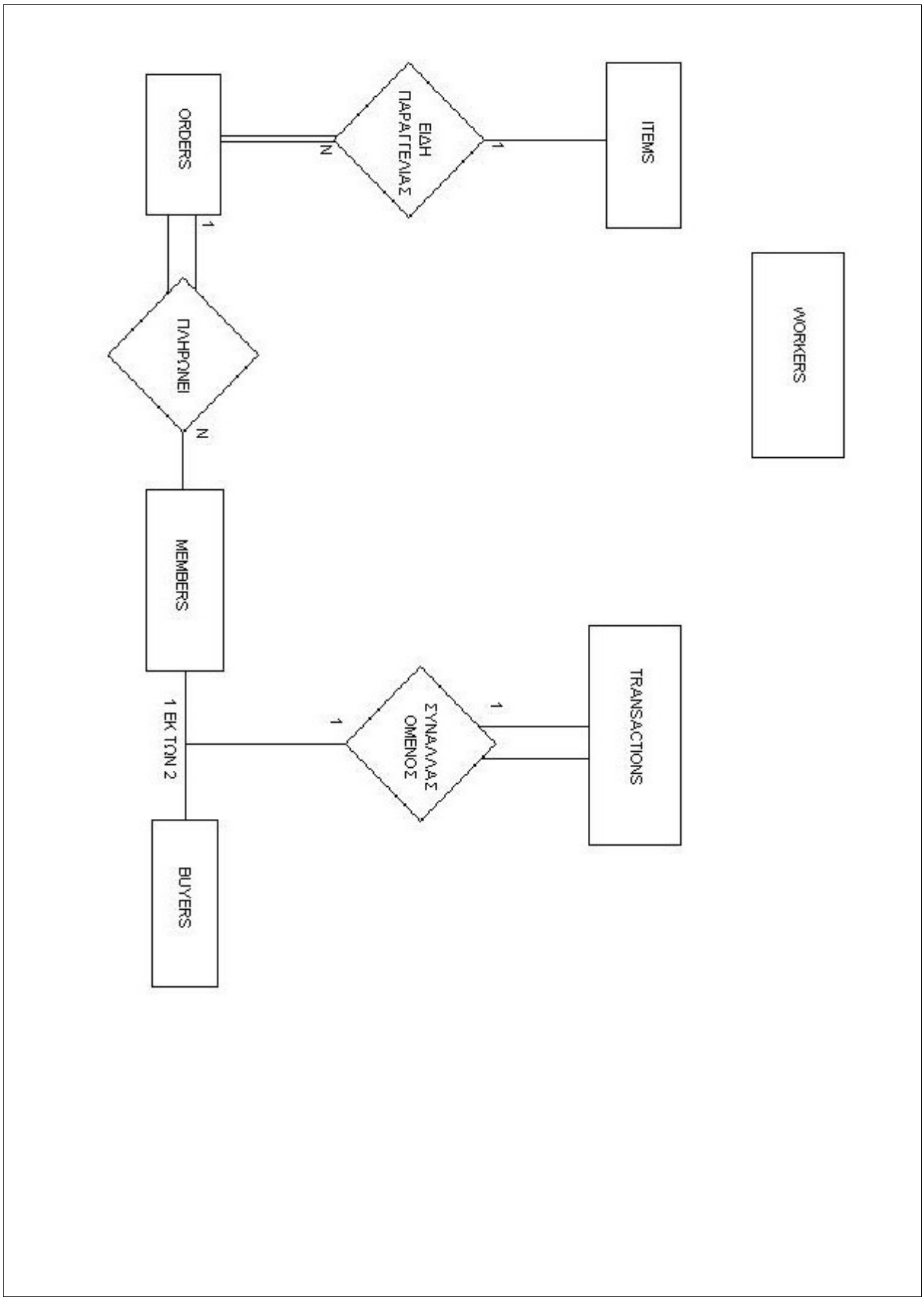
ORDERS:
SN, INTEGER, NOT NULL (DOMAIN: ORDER_INT_KEY)
PAYER, INTEGER, NOT NULL (DOMAIN: MEMBER_INT_KEY)
PRICE, DECIMAL 5, 2, NOT NULL
DELIVERY DATE, TIMESTAMP, NOT NULL
DELIVERY ADDRESS, VARCHAR 25, NOT NULL
COMMENTS-CARD, BLOB SUB_TYPE 0 SEGMENT SIZE 300

ORDER STUFF:
SN, INTEGER, NOT NULL (DOMAIN: ORDER_INT_KEY)
ITEM, INTEGER, NOT NULL (DOMAIN: ITEM_INT_KEY)
QUANTITY, INTEGER, NOT NULL

TRANSACTIONS:
TRANSCODE, CHAR 10, NOT NULL
DATE, DATE, NOT NULL
TIME, TIME, NOT NULL
TYPE, CHAR 4, NOT NULL
BUYER, INTEGER, NOT NULL (DOMAIN: MEMBER_INT_KEY)
SELLER, INTEGER, NOT NULL (DOMAIN: MEMBER_INT_KEY)
AMOUNT, DECIMAL 15, 2, NOT NULL

WORKERS:
WORKER NUM, INTEGER, NOT NULL (DOMAIN: WORKER_INT_KEY)
ID_NUM, CHAR 11, NOT NULL
FIRST NAME, CHAR 15, NOT NULL
LAST NAME, CHAR 15, NOT NULL
SALARY, DECIMAL 15, 2, NOT NULL
JOB DESC, CHAR 10, NOT NULL

Μοντελο Οντοτήτων-Συσχετίσεων



Κανονοικοποιηση

Οι σχεσεις είναι σε Boyce-Codd κανονική μορφή. Έχουμε:

TSN --> CODE, NAME, TEL1, CELL1, TEL2, CELL2, CELL3, ADDRESS

Ειναι σε BCNF, αφού (α) το κλειδι είναι μονοσύνολο, (β), το κλειδι προσδιορίζει μοναδικά καθε στοιχειο της σχέσης, και (γ) κανενα απο τα στοιχεια του δευτερου μελους της σχεσης που δεν ειναι υποψήφια κλειδια δεν μπορεί να προσδιορίσει μονοσήμαντα το κλειδί.

KEYCODE --> FULL NAME, PRICE, DESCRIPTION, QUANTITY

Ειναι σε BCNF, αφού (α) το κλειδι είναι μονοσύνολο, (β), το κλειδι προσδιορίζει μοναδικά καθε στοιχειο της σχέσης, και (γ) κανενα απο τα στοιχεια του δευτερου μελους της σχεσης δεν μπορεί να προσδιορίσει μονοσήμαντα το κλειδί.

CODE --> FIRST NAME, LAST NAME, TEL1, CELL1, CELL2, CREDIT CARD 1, CREDIT CARD 2, EMAIL, PASS, ADDRESS

Ειναι σε BCNF, αφού (α) το κλειδι είναι μονοσύνολο, (β), το κλειδι προσδιορίζει μοναδικά καθε στοιχειο της σχέσης, και (γ) κανενα απο τα στοιχεια του δευτερου μελους της σχεσης δεν μπορεί να προσδιορίσει μονοσήμαντα το κλειδί.

SN --> PAYER, PRICE, DELIVERY DATE, DELIVERY ADDRESS, COMMENTS-CARD

Ειναι σε BCNF, αφού (α) το κλειδι είναι μονοσύνολο, (β), το κλειδι προσδιορίζει μοναδικά καθε στοιχειο της σχέσης, και (γ) κανενα απο τα στοιχεια του δευτερου μελους της σχεσης δεν μπορεί να προσδιορίσει μονοσήμαντα το κλειδί.

SN, ITEM --> QUANTITY

Ειναι σε BCNF, αφού (α) το κλειδι είναι ελαχιστικό (ούτε το SN ουτε το ITEM μόνα τους μπορούν να προσδιορίσουν το QUANTITY), (β), το κλειδι προσδιορίζει μοναδικά καθε στοιχειο της σχέσης, και (γ) το QUANTITY δεν μπορεί να προσδιορίσει μονοσήμαντα το κλειδί.

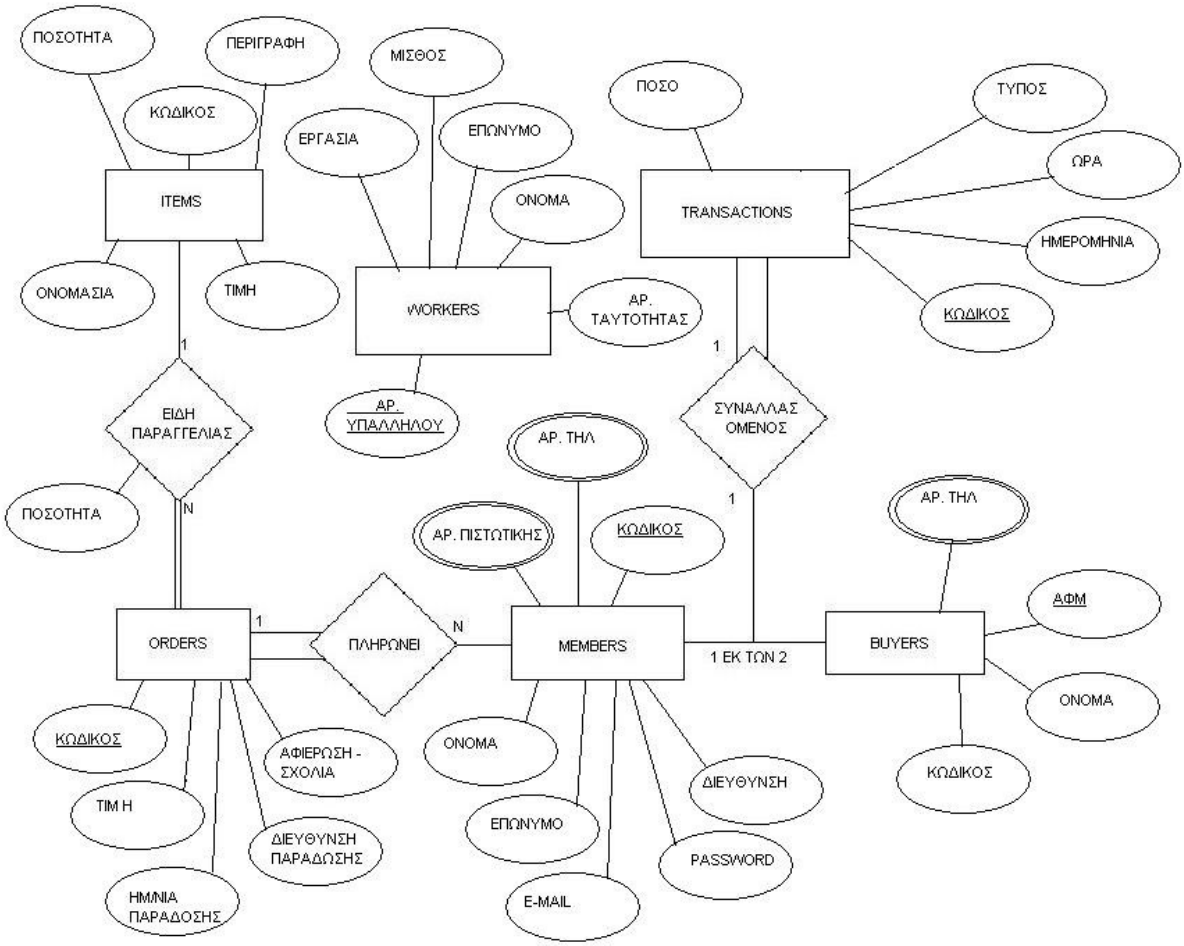
TRANSCODE --> DATE, TIME, TYPE, BUYER, SELLER, AMOUNT

Ειναι σε BCNF, αφού (α) το κλειδι είναι μονοσύνολο, (β), το κλειδι προσδιορίζει μοναδικά καθε στοιχειο της σχέσης, και (γ) κανενα απο τα στοιχεια του δευτερου μελους της σχεσης δεν μπορεί να προσδιορίσει μονοσήμαντα το κλειδί.

WORKER NUM --> ID_NUM, FIRST NAME, LAST NAME, SALARY, JOB DESC

Ειναι σε BCNF, αφού (α) το κλειδι είναι μονοσύνολο, (β), το κλειδι προσδιορίζει μοναδικά καθε στοιχειο της σχέσης, και (γ) κανενα απο τα στοιχεια του δευτερου μελους της σχεσης δεν μπορεί να προσδιορίσει μονοσήμαντα το κλειδί.

Τελικο Μοντελο Οντοτήτων-Συσχετήσεων



ΚΩΔΙΚΑΣ SQL

```
/******************************************************************************/
/****         Generated by IBExpert 2006.10.14 23/1/2008 13:49:50         ****/
/******************************************************************************/

SET SQL DIALECT 3;

SET NAMES NONE;

CREATE DATABASE 'C:\Users\Galadar\Science\University\741 - Databases\Main
Project\anthopoleio.fdb'
USER 'SYSDBA' PASSWORD 'XXXXXXXX'
PAGE_SIZE 16384
DEFAULT CHARACTER SET NONE;



/******************************************************************************/
/****                             Domains                                  ****/
/******************************************************************************/

CREATE DOMAIN ITEM_INT_KEY AS
INTEGER
NOT NULL;

CREATE DOMAIN MEMBER_INT_KEY AS
INTEGER
NOT NULL;

CREATE DOMAIN NEW_DOMAIN AS
CHAR(11)
NOT NULL;

CREATE DOMAIN ORDER_INT_KEY AS
INTEGER
NOT NULL;
```

```
CREATE DOMAIN TEL_NUM AS
CHAR(10);

CREATE DOMAIN WORKER_INT_KEY AS
INTEGER
NOT NULL;



SET TERM ^ ;



/*****************************************************************************/
/****                    Stored Procedures                    ****/
/*****************************************************************************/

CREATE PROCEDURE ADD_ITEM (
    ITEM_NAME VARCHAR(25),
    PRICE DECIMAL(15,2),
    DESCRIPTION VARCHAR(80))
AS
BEGIN
  EXIT;
END^


CREATE PROCEDURE ADD_MEMBER (
    FIRST_NAME CHAR(15),
    LAST_NAME CHAR(15),
    TEL1 CHAR(10),
    CELL1 CHAR(10),
    CELL2 CHAR(10),
    CC1 CHAR(16),
    CC2 CHAR(16),
    ADDRESS VARCHAR(50),
    EMAIL VARCHAR(25),
```

```
    PASS CHAR(12))
AS
BEGIN
  EXIT;
END^


CREATE PROCEDURE ADD_ORDER (
    MEMBER INTEGER,
    D_DATE DATE,
    ADDRESS VARCHAR(25),
    AMOUNT DECIMAL(15,2),
    COMMENTS_CARD BLOB SUB_TYPE 0 SEGMENT SIZE 300)
RETURNS (
    ORDER_NUMBER INTEGER)
AS
BEGIN
  EXIT;
END^


CREATE PROCEDURE ADD_ORDER_STUFF (
    SN INTEGER,
    ITEM INTEGER,
    QUANT INTEGER)
AS
BEGIN
  EXIT;
END^


CREATE PROCEDURE BUY (
    ITEM INTEGER,
    QUANT INTEGER,
    SELLER INTEGER,
    AMOUNT DECIMAL(15,2))
AS
```

```
BEGIN
  EXIT;
END^


CREATE PROCEDURE HIRE_EMPLOYER (
    FIRST_NAME CHAR(15),
    LAST_NAME CHAR(15),
    JOB_DESCRIPTION CHAR(10),
    SALARY DECIMAL(15,2))
AS
BEGIN
  EXIT;
END^


CREATE PROCEDURE ISOLOGISMOS (
    DATE_TO_START DATE)
RETURNS (
    TOTAL DECIMAL(15,2))
AS
BEGIN
  EXIT;
END^


CREATE PROCEDURE REQUIRED_QUANTITY (
    ITEM INTEGER,
    DATE_UNTIL DATE)
RETURNS (
    QUANTITY_REQUIRED INTEGER)
AS
BEGIN
  EXIT;
END^
```

```
CREATE PROCEDURE SELL (
   QUANT INTEGER,
   ITEM INTEGER,
   MEMBER INTEGER,
   AMOUNT DECIMAL(15,2))
AS
BEGIN
  EXIT;
END^



SET TERM ; ^


/*****************************************************************************/
/****                         Tables                                  ****/
/*****************************************************************************/



CREATE TABLE BYERS (
   TSN      CHAR(10) NOT NULL,
   CODE     MEMBER_INT_KEY,
   NAME     CHAR(25) NOT NULL,
   TEL1     TEL_NUM NOT NULL,
   CELL1    TEL_NUM NOT NULL,
   TEL2     TEL_NUM,
   CELL2    TEL_NUM,
   CELL3    TEL_NUM,
   ADDRESS  VARCHAR(25) NOT NULL
);


CREATE TABLE ITEMS (
   KEYCODE         ITEM_INT_KEY,
   FULL_NAME       VARCHAR(25) NOT NULL,
```

```
    PRICE           DECIMAL(3,2) NOT NULL,
    DESCRIPTION     VARCHAR(80) NOT NULL,
    QUANTITY_STORED  INTEGER NOT NULL
);


CREATE TABLE JOB_DESCRIPTIONS (
    JOB_NAME  CHAR(10) NOT NULL
);


CREATE TABLE KEYS (
    NAME      CHAR(10) NOT NULL,
    NEXT_KEY  INTEGER NOT NULL
);


CREATE TABLE MEMBERS (
    CODE         MEMBER_INT_KEY NOT NULL,
    FIRST_NAME    CHAR(15) NOT NULL,
    LAST_NAME     CHAR(15) NOT NULL,
    TEL1         CHAR(10),
    CELL1        CHAR(10) NOT NULL,
    CELL2        CHAR(10),
    CREDIT_CARD1  CHAR(16),
    CREDIT_CARD2  CHAR(16),
    EMAIL        VARCHAR(25),
    PASS         CHAR(12),
    ADDRESS      VARCHAR(50) NOT NULL
);


CREATE TABLE ORDER_STUFF (
    SN       ORDER_INT_KEY,
    ITEM     ITEM_INT_KEY,
    QUANTITY  INTEGER NOT NULL
);
```

```
CREATE TABLE ORDERS (
   SN              ORDER_INT_KEY NOT NULL,
   PAYER           MEMBER_INT_KEY,
   PRICE           DECIMAL(5,2) NOT NULL,
   DELIVERY_DATE    TIMESTAMP NOT NULL,
   DELIVERY_ADDRESS  VARCHAR(25) NOT NULL,
   "COMMENTS-CARD"   BLOB SUB_TYPE 0 SEGMENT SIZE 300
);


CREATE TABLE TRANS_TYPE (
   "TYPE"  CHAR(4) NOT NULL
);


CREATE TABLE TRANSACTIONS (
   TRANSCODE  CHAR(10) NOT NULL,
   "DATE"     DATE NOT NULL,
   "TIME"     TIME NOT NULL,
   "TYPE"     CHAR(4) NOT NULL,
   BUYER      INTEGER DEFAULT 100000 NOT NULL,
   SELLER     INTEGER DEFAULT 500000 NOT NULL,
   AMOUNT     DECIMAL(15,2) NOT NULL
);


CREATE TABLE WORKERS (
   WORKER_NUM  WORKER_INT_KEY NOT NULL,
   ID_NUM      CHAR(11) NOT NULL,
   FIRST_NAME  CHAR(15) NOT NULL,
   LAST_NAME   CHAR(15) NOT NULL,
   SALARY      DECIMAL(15,2) NOT NULL,
   JOB_DESC    CHAR(10) NOT NULL
);
```

```
/*****************************************************************************/
/****                        Views                          ****/
/*****************************************************************************/


/* View: ALL_ORDERS */
CREATE VIEW ALL_ORDERS(
   ORDER_NUMBER,
   PAYER_FIRST_NAME,
   PAYER_LAST_NAME,
   DELIVERY_DATE,
   ITEMS_NUMBER)
AS
select orders.sn, members.first_name, members.last_name, orders.delivery_date,
COUNT(order_stuff.sn)
from orders, members, order_stuff
where (members.code = orders.payer) AND (order_stuff.sn = orders.sn)
group by orders.sn, members.first_name, members.last_name, orders.delivery_date
order by orders.delivery_date
;



/* View: CATALOG */
CREATE VIEW CATALOG(
   NAME,
   PRICE,
   DESCRIPTION)
AS
select items.full_name, items.price, items.description
from items
where ITEMS.quantity_stored>0
;
```

```
/* View: CUSTOMERS */
CREATE VIEW CUSTOMERS(
    MEMBERCODE,
    FIRST_NAME,
    LAST_NAME,
    CONTACT_PHONE,
    SPEND_TOTAL,
    ADDRESS)
AS
select members.code, members.first_name, members.last_name, members.cell1,
SUM(transactions.amount), members.address
from members, transactions
where transactions.buyer = members.code
group by members.code, members.first_name, members.last_name, members.cell1,
members.address
;



/* View: DAILY_SELLS */
CREATE VIEW DAILY_SELLS(
    CODE,
    "TIME",
    BUYER,
    AMOUNT)
AS
select transactions.transcode, transactions."TIME", transactions.buyer, transactions.amount
From transactions
Where transactions."TYPE" = 'SELL' AND transactions."DATE" = current_date
;



/* View: DAILY_TRANS */
CREATE VIEW DAILY_TRANS(
```

```
   CODE,
   "TYPE",
   "TIME",
   BUYER,
   SELLER,
   AMOUNT)
AS
select transactions.transcode, transactions."TYPE", transactions."TIME", transactions.buyer, transactions.seller, transactions.amount
FROM transactions
where transactions."DATE" = current_date
;



/* View: DAY_ORDERS */
CREATE VIEW DAY_ORDERS(
   ORDER_NUMBER,
   PAYER_FIRST_NAME,
   PAYER_LAST_NAME,
   DELIVERY_TIME,
   DELIVERY_ADDRESS,
   ITEMS_NUMBER)
AS
select orders.sn, members.first_name, members.last_name, orders.delivery_date, orders.delivery_address, COUNT(order_stuff.sn)
from orders, members, order_stuff
where (members.code = orders.payer) AND (order_stuff.sn = orders.sn) AND (orders.delivery_date < current_date +1) AND (orders.delivery_date > current_date)
group by orders.sn, members.first_name, members.last_name, orders.delivery_date, orders.delivery_address
order by orders.delivery_date
;



/* View: ORDER_ITEMS */
CREATE VIEW ORDER_ITEMS(
```

```
   ORDER_NUM,
   ITEM_NAME,
   ITEM_QUANTITY,
   DELIVERY_DATE,
   DELIVERY_ADDRESS)
AS
SELECT orders.sn, items.full_name, order_stuff.quantity, orders.delivery_date,
orders.delivery_address
FROM ORDERS, items, order_stuff
where (orders.sn = order_stuff.sn) and (items.keycode = order_stuff.item)
;



/* View: STORED_ITEMS */
CREATE VIEW STORED_ITEMS(
   ITEM,
   QUANTITY)
AS
select ITEMS.full_name, items.quantity_stored
from ITEMS
;



/* View: WEEK_ORDERS */
CREATE VIEW WEEK_ORDERS(
   ORDER_NUMBER,
   PAYER_FIRST_NAME,
   PAYER_LAST_NAME,
   DELIVERY_DATE,
   ITEMS_NUMBER)
AS
select orders.sn, members.first_name, members.last_name, orders.delivery_date,
COUNT(order_stuff.sn)
from orders, members, order_stuff
where (members.code = orders.payer) AND (order_stuff.sn = orders.sn) and (orders.delivery_date <
current_date + 8)
```

```
group by orders.sn, members.first_name, members.last_name, orders.delivery_date
order by orders.delivery_date
;



/* Check constraints definition */

ALTER TABLE ORDERS ADD CONSTRAINT CHK1_ORDERS check (payer <> '100000');
ALTER TABLE TRANSACTIONS ADD CONSTRAINT CHK2_TRANSACTIONS check ((seller > 499999) OR (seller = 0));
ALTER TABLE ITEMS ADD CONSTRAINT CHK1_ITEMS check (QUANTITY_STORED>=0);
ALTER TABLE ORDER_STUFF ADD CONSTRAINT CHK1_ORDER_STUFF check (quantity >0);
ALTER TABLE TRANSACTIONS ADD CONSTRAINT CHK1_TRANSACTIONS check (buyer < 500000);



/*************************************************************************/
/****                    Unique Constraints                    ****/
/*************************************************************************/

ALTER TABLE BYERS ADD CONSTRAINT UNQ1_BYERS UNIQUE (CODE);
ALTER TABLE MEMBERS ADD CONSTRAINT UNQ1_MEMBERS UNIQUE (CREDIT_CARD1);
ALTER TABLE MEMBERS ADD CONSTRAINT UNQ2_MEMBERS UNIQUE (CREDIT_CARD2);
ALTER TABLE TRANSACTIONS ADD CONSTRAINT UNQ1_TRANSACTIONS UNIQUE ("DATE", "TIME");
ALTER TABLE WORKERS ADD CONSTRAINT UNQ1_WORKERS UNIQUE (ID_NUM);



/*************************************************************************/
/****                    Primary Keys                    ****/
/*************************************************************************/

ALTER TABLE BYERS ADD CONSTRAINT PK_BYERS PRIMARY KEY (TSN);
```

```
ALTER TABLE ITEMS ADD CONSTRAINT PK_ITEMS PRIMARY KEY (KEYCODE);

ALTER TABLE JOB_DESCRIPTIONS ADD CONSTRAINT PK_JOB_DESCRIPTIONS
PRIMARY KEY (JOB_NAME);

ALTER TABLE KEYS ADD CONSTRAINT PK_KEYS PRIMARY KEY (NAME);

ALTER TABLE MEMBERS ADD CONSTRAINT PK_MEMBERS PRIMARY KEY (CODE);

ALTER TABLE ORDERS ADD CONSTRAINT PK_ORDERS PRIMARY KEY (SN);

ALTER TABLE ORDER_STUFF ADD CONSTRAINT PK_ORDER_STUFF PRIMARY KEY
(ITEM, SN);

ALTER TABLE TRANSACTIONS ADD CONSTRAINT PK_TRANSACTIONS PRIMARY KEY
(TRANSCODE);

ALTER TABLE TRANS_TYPE ADD CONSTRAINT PK_TRANS_TYPE PRIMARY KEY
("TYPE");

ALTER TABLE WORKERS ADD CONSTRAINT PK_WORKERS PRIMARY KEY
(WORKER_NUM);



/*****************************************************************************/

/****                    Foreign Keys                    ****/

/*****************************************************************************/


ALTER TABLE ORDERS ADD CONSTRAINT FK_ORDERS_1 FOREIGN KEY (PAYER)
REFERENCES MEMBERS (CODE);

ALTER TABLE ORDER_STUFF ADD CONSTRAINT FK_ORDER_STUFF_1 FOREIGN KEY
(SN) REFERENCES ORDERS (SN);

ALTER TABLE ORDER_STUFF ADD CONSTRAINT FK_ORDER_STUFF_2 FOREIGN KEY
(ITEM) REFERENCES ITEMS (KEYCODE);

ALTER TABLE TRANSACTIONS ADD CONSTRAINT FK_TRANSACTIONS_1 FOREIGN
KEY ("TYPE") REFERENCES TRANS_TYPE ("TYPE");

ALTER TABLE TRANSACTIONS ADD CONSTRAINT FK_TRANSACTIONS_2 FOREIGN
KEY (BUYER) REFERENCES MEMBERS (CODE) ON DELETE SET DEFAULT ON UPDATE
CASCADE;

ALTER TABLE TRANSACTIONS ADD CONSTRAINT FK_TRANSACTIONS_3 FOREIGN
KEY (SELLER) REFERENCES BYERS (CODE) ON DELETE SET DEFAULT ON UPDATE
CASCADE;

ALTER TABLE WORKERS ADD CONSTRAINT FK_WORKERS_1 FOREIGN KEY
(JOB_DESC) REFERENCES JOB_DESCRIPTIONS (JOB_NAME);



/*****************************************************************************/

/****                    Indices                    ****/

/*****************************************************************************/
```

```
CREATE INDEX BYERS_IDX1 ON BYERS (NAME);
CREATE INDEX ITEMS_IDX1 ON ITEMS (FULL_NAME);
CREATE INDEX MEMBERS_IDX1 ON MEMBERS (LAST_NAME);
CREATE INDEX MEMBERS_IDX2 ON MEMBERS (EMAIL);
CREATE INDEX ORDERS_IDX1 ON ORDERS (PRICE);
CREATE INDEX ORDERS_IDX2 ON ORDERS (DELIVERY_DATE);
CREATE INDEX ORDER_STUFF_IDX1 ON ORDER_STUFF (QUANTITY);
CREATE INDEX TRANSACTIONS_IDX1 ON TRANSACTIONS (AMOUNT);
CREATE INDEX WORKERS_IDX1 ON WORKERS (JOB_DESC);
CREATE INDEX WORKERS_IDX2 ON WORKERS (SALARY);


/*****************************************************************************/
/****                    Stored Procedures                    ****/
/*****************************************************************************/


SET TERM ^ ;

ALTER PROCEDURE ADD_ITEM (
    ITEM_NAME VARCHAR(25),
    PRICE DECIMAL(15,2),
    DESCRIPTION VARCHAR(80))
AS
declare variable it_nu integer;
begin
  SELECT KEYS.next_key FROM KEYS WHERE KEYS.name = 'ITEM_KEY' INTO :it_nu;
  insert into items values (:IT_NU, :item_name, :price, :description, 0);
  update keys set keys.next_key = keys.next_key+1 where keys.name = 'ITEM_KEY';
end
^

ALTER PROCEDURE ADD_MEMBER (
    FIRST_NAME CHAR(15),
    LAST_NAME CHAR(15),
    TEL1 CHAR(10),
```

```
   CELL1 CHAR(10),
   CELL2 CHAR(10),
   CC1 CHAR(16),
   CC2 CHAR(16),
   ADDRESS VARCHAR(50),
   EMAIL VARCHAR(25),
   PASS CHAR(12))
AS
declare variable member_nu integer;
begin
  SELECT KEYS.next_key FROM KEYS WHERE KEYS.name = 'MEMBER_KEY' INTO
:MEMBER_NU;
  insert into MEMBERS values (:MEMBER_NU, :first_name, :last_name, :TEL1, :CELL1,
:CELL2, :CC1, :CC2, :EMAIL, :PASS, :ADDRESS);
  update keys set keys.next_key = keys.next_key+1 where keys.name = 'MEMBER_KEY';
end
^


ALTER PROCEDURE ADD_ORDER (
   MEMBER INTEGER,
   D_DATE DATE,
   ADDRESS VARCHAR(25),
   AMOUNT DECIMAL(15,2),
   COMMENTS_CARD BLOB SUB_TYPE 0 SEGMENT SIZE 300)
RETURNS (
   ORDER_NUMBER INTEGER)
AS
declare variable ord_nu integer;
begin
  SELECT KEYS.next_key FROM KEYS WHERE KEYS.name = 'ORDER_KEY' INTO :ord_nu;
  insert into orders values (:ord_nu, :MEMBER, :AMOUNT, :d_date, :ADDRESS,
:COMMENTS_CARD);
  update keys set keys.next_key = keys.next_key+1 where keys.name = 'ORDER_KEY';
  order_number = :ord_nu;
end
^


ALTER PROCEDURE ADD_ORDER_STUFF (
```

```
    SN INTEGER,
    ITEM INTEGER,
    QUANT INTEGER)
AS
begin
   INSERT into order_stuff VALUES (:SN, :ITEM, :quant);
end
^

ALTER PROCEDURE BUY (
    ITEM INTEGER,
    QUANT INTEGER,
    SELLER INTEGER,
    AMOUNT DECIMAL(15,2))
AS
declare variable trans integer;
begin
  SELECT KEYS.next_key FROM KEYS WHERE KEYS.name = 'TRANS_KEY' INTO :TRANS;
  insert into transactions values (:TRANS, current_date, current_time, 'BUY', 0, :SELLER,
:AMOUNT);
  UPDATE ITEMS SET ITEMS.quantity_stored = ITEMS.quantity_stored + :quant WHERE
ITEMS.keycode = :ITEM;
  update keys set keys.next_key = keys.next_key+1 where keys.name = 'TRANS_KEY';

end
^

ALTER PROCEDURE HIRE_EMPLOYER (
    FIRST_NAME CHAR(15),
    LAST_NAME CHAR(15),
    JOB_DESCRIPTION CHAR(10),
    SALARY DECIMAL(15,2))
AS
declare variable worker_nu integer;
begin
  SELECT KEYS.next_key FROM KEYS WHERE KEYS.name = 'WORKE_KEY' INTO
:WORKER_NU;
  insert into workers values (:worker_nu, :first_name, :last_name, :salary, :job_description);
```

```
  update keys set keys.next_key = keys.next_key+1 where keys.name = 'WORKE_KEY';
end
^

ALTER PROCEDURE ISOLOGISMOS (
    DATE_TO_START DATE)
RETURNS (
    TOTAL DECIMAL(15,2))
AS
declare variable spend decimal(15,2);
declare variable taken decimal(15,2);
begin
   SELECT SUM(transactions.amount) FROM transactions where (transactions."TYPE" = 'SELL')
AND (transactions."DATE" > :date_to_start -1) into :taken;
   SELECT SUM(transactions.amount) FROM transactions where (transactions."TYPE" = 'BUY')
AND (transactions."DATE" > :date_to_start -1) into :spend;


   TOTAL = :taken - :spend;
end
^

ALTER PROCEDURE REQUIRED_QUANTITY (
    ITEM INTEGER,
    DATE_UNTIL DATE)
RETURNS (
    QUANTITY_REQUIRED INTEGER)
AS
declare variable stored integer;
declare variable ordered integer;
begin
   SELECT SUM(order_stuff.quantity) FROM order_stuff
   where (order_stuff.item = :item) AND (order_stuff.sn) in (select ORDERS.sn FROM orders
where orders.delivery_date between current_date AND :date_until)
   INTO :ordered;
   SELECT SUM(items.quantity_stored) FROM ITEMS where (ITEMS.keycode = :item) INTO
:stored;


   quantity_required = :ordered - :stored;
```

```
end
^

ALTER PROCEDURE SELL (
    QUANT INTEGER,
    ITEM INTEGER,
    MEMBER INTEGER,
    AMOUNT DECIMAL(15,2))
AS
declare variable trans integer;
begin
  SELECT KEYS.next_key FROM KEYS WHERE KEYS.name = 'TRANS_KEY' INTO :TRANS;
  UPDATE ITEMS SET ITEMS.quantity_stored = ITEMS.quantity_stored - :quant WHERE
ITEMS.keycode = :ITEM;
  insert into transactions values (:TRANS, current_date, current_time, 'SELL', :MEMBER, 0,
:AMOUNT);
  update keys set keys.next_key = keys.next_key+1 where keys.name = 'TRANS_KEY';
end
^



SET TERM ; ^



/* Fields descriptions */

DESCRIBE FIELD BUYER TABLE TRANSACTIONS
'MEMBER CODE';
```

ΔΕΔΟΜΕΝΑ ΠΙΝΑΚΩΝ

```
INSERT INTO BYERS (TSN, CODE, NAME, TEL1, CELL1, TEL2, CELL2, CELL3, ADDRESS)
VALUES ('0000000000', 500000, 'DELETED                ', '0000000000', '0000000000', NULL,
NULL, NULL, 'NONE');

INSERT INTO BYERS (TSN, CODE, NAME, TEL1, CELL1, TEL2, CELL2, CELL3, ADDRESS)
VALUES ('5631355648', 500001, 'Flower Seller 1        ', '2321365320', '6521254862', NULL,
NULL, NULL, 'FS1, Athens');

INSERT INTO BYERS (TSN, CODE, NAME, TEL1, CELL1, TEL2, CELL2, CELL3, ADDRESS)
VALUES ('3631356321', 500002, 'Flower Seller 2        ', '6221354863', '3133654005',
'5686332188', NULL, NULL, 'FS2, Iwannina');

INSERT INTO BYERS (TSN, CODE, NAME, TEL1, CELL1, TEL2, CELL2, CELL3, ADDRESS)
VALUES ('0000000001', 0, 'NOT APPLICABLE          ', '0000000000', '0000000000', NULL,
NULL, NULL, 'NONE');


COMMIT WORK;


INSERT INTO ITEMS (KEYCODE, FULL_NAME, PRICE, DESCRIPTION,
QUANTITY_STORED) VALUES (1, 'Kokkina Triadafylla 30p', 1.19999999999999996, 'Kokkina
Triadaffyla me mikos 30cm', 20);

INSERT INTO ITEMS (KEYCODE, FULL_NAME, PRICE, DESCRIPTION,
QUANTITY_STORED) VALUES (2, 'Kokkina Triadafylla 15p', 0.900000000000000022, 'Kokkina
Triadaffyla me mikos 15cm', 32);

INSERT INTO ITEMS (KEYCODE, FULL_NAME, PRICE, DESCRIPTION,
QUANTITY_STORED) VALUES (3, 'Lefka Triadafylla 30p', 1.19999999999999996, 'Lefka
Triadaffyla me mikos 30cm', 27);

INSERT INTO ITEMS (KEYCODE, FULL_NAME, PRICE, DESCRIPTION,
QUANTITY_STORED) VALUES (4, 'Lefka Triadafylla 15p', 0.900000000000000022, 'Lefka
Triadaffyla me mikos 15cm', 12);

INSERT INTO ITEMS (KEYCODE, FULL_NAME, PRICE, DESCRIPTION,
QUANTITY_STORED) VALUES (5, 'Xrysanthema', 1.19999999999999996, 'Lefka Xrysanthema
me mikos peripou 40cm', 30);

INSERT INTO ITEMS (KEYCODE, FULL_NAME, PRICE, DESCRIPTION,
QUANTITY_STORED) VALUES (6, 'Vasilikos', 5, 'Vasilikos se maisaia glastra diametrou peripou
13cm', 7);


COMMIT WORK;


INSERT INTO JOB_DESCRIPTIONS (JOB_NAME) VALUES ('CEO      ');

INSERT INTO JOB_DESCRIPTIONS (JOB_NAME) VALUES ('MANAGER  ');

INSERT INTO JOB_DESCRIPTIONS (JOB_NAME) VALUES ('SELLER   ');

INSERT INTO JOB_DESCRIPTIONS (JOB_NAME) VALUES ('DRIVER   ');


COMMIT WORK;
```

```
INSERT INTO KEYS (NAME, NEXT_KEY) VALUES ('MEMBER_KEY', 100007);
INSERT INTO KEYS (NAME, NEXT_KEY) VALUES ('ITEM_KEY  ', 7);
INSERT INTO KEYS (NAME, NEXT_KEY) VALUES ('ORDER_KEY ', 3);
INSERT INTO KEYS (NAME, NEXT_KEY) VALUES ('WORKE_KEY ', 10000006);
INSERT INTO KEYS (NAME, NEXT_KEY) VALUES ('TRANS_KEY ', 15);
INSERT INTO KEYS (NAME, NEXT_KEY) VALUES ('BUYER_KEY ', 500003);

COMMIT WORK;

INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (100000, 'GENERAL
', 'CUSTOMER       ', NULL, '0000000000', NULL, NULL, NULL, NULL, NULL, 'NONE');
INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (100001, 'George
', 'Bushed        ', '5806995216', '0343168597', NULL, '6482165487301091', '3519762248513364',
NULL, NULL, 'GB17, Athens');
INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (100002, 'Bill         ',
'Cilndon       ', '5246813216', '8543162917', NULL, '6482161679515891', NULL, NULL, NULL,
'BC23, Athens');
INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (100003, 'William
', 'McCoy         ', '5281195216', '7163168557', '5893147268', '6482128190315891', NULL, NULL,
NULL, 'WM39, Athens');
INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (100004, 'Tony
', 'Rebl         ', '8439285216', '8543915597', NULL, '2795878531528901', '5873165429982014',
NULL, NULL, 'TR14, Iwannina');
INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (100005, 'John
', 'Rabmo        ', '5489935216', '8543641297', '2548264879', '8652921459834191',
'9314658726514839', NULL, NULL, 'JR32, Salonika');
INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (100006, 'Michael
', 'Jonshon       ', NULL, '3254981369', NULL, '2345903249802168', NULL, NULL, NULL, 'MJ52,
Patra');
INSERT INTO MEMBERS (CODE, FIRST_NAME, LAST_NAME, TEL1, CELL1, CELL2,
CREDIT_CARD1, CREDIT_CARD2, EMAIL, PASS, ADDRESS) VALUES (0, 'NOT          ',
'APPLICABLE    ', NULL, '0000000000', NULL, NULL, NULL, NULL, NULL, 'NONE');

COMMIT WORK;
```

```
INSERT INTO ORDERS (SN, PAYER, PRICE, DELIVERY_DATE, DELIVERY_ADDRESS)
VALUES (1, 100001, 52, '2008-01-25 00:00:00', 'GS13, Athens');

INSERT INTO ORDERS (SN, PAYER, PRICE, DELIVERY_DATE, DELIVERY_ADDRESS)
VALUES (2, 100002, 43, '2008-02-03 00:00:00', 'FF7, Salonika');


COMMIT WORK;


INSERT INTO ORDER_STUFF (SN, ITEM, QUANTITY) VALUES (1, 1, 5);

INSERT INTO ORDER_STUFF (SN, ITEM, QUANTITY) VALUES (1, 3, 5);

INSERT INTO ORDER_STUFF (SN, ITEM, QUANTITY) VALUES (1, 6, 1);

INSERT INTO ORDER_STUFF (SN, ITEM, QUANTITY) VALUES (2, 6, 2);

INSERT INTO ORDER_STUFF (SN, ITEM, QUANTITY) VALUES (2, 2, 15);

INSERT INTO ORDER_STUFF (SN, ITEM, QUANTITY) VALUES (2, 4, 15);


COMMIT WORK;


INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('1         ', '2008-01-23', '01:34:19', 'BUY ', 100000, 500000, 23);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('2         ', '2008-01-23', '01:35:02', 'BUY ', 100000, 500000, 230);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('3         ', '2008-01-23', '01:35:21', 'BUY ', 100000, 500000, 130);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('4         ', '2008-01-23', '01:35:52', 'SELL', 100000, 500000, 180);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('5         ', '2008-01-23', '01:36:08', 'SELL', 100000, 500000, 180);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('6         ', '2008-01-23', '01:36:18', 'SELL', 100000, 500000, 160);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('7         ', '2008-01-23', '01:36:32', 'SELL', 100000, 500000, 160);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('8         ', '2008-01-23', '01:36:47', 'SELL', 100000, 500000, 460);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('9         ', '2008-01-23', '01:38:27', 'SELL', 100000, 500000, 460);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('10        ', '2008-01-23', '01:38:47', 'SELL', 100000, 500000, 460);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('11        ', '2008-01-23', '01:39:41', 'SELL', 100000, 500000, 23);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('12        ', '2008-01-23', '02:44:21', 'SELL', 100002, 500000, 6);

INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
```

```
SELLER, AMOUNT) VALUES ('13        ', '2008-01-23', '10:23:03', 'SELL', 100002, 0, 13.2);
INSERT INTO TRANSACTIONS (TRANSCODE, "DATE", "TIME", "TYPE", BUYER,
SELLER, AMOUNT) VALUES ('14        ', '2008-01-23', '10:23:40', 'BUY ', 0, 500002, 110);

COMMIT WORK;

INSERT INTO TRANS_TYPE ("TYPE") VALUES ('BUY ');
INSERT INTO TRANS_TYPE ("TYPE") VALUES ('SELL');

COMMIT WORK;

INSERT INTO WORKERS (WORKER_NUM, ID_NUM, FIRST_NAME, LAST_NAME,
SALARY, JOB_DESC) VALUES (10000000, NULL, 'Iwannis        ', 'Stamatiou      ', 170, 'CEO
');
INSERT INTO WORKERS (WORKER_NUM, ID_NUM, FIRST_NAME, LAST_NAME,
SALARY, JOB_DESC) VALUES (10000002, NULL, 'Steven         ', 'Lord           ', 120, 'SELLER
');
INSERT INTO WORKERS (WORKER_NUM, ID_NUM, FIRST_NAME, LAST_NAME,
SALARY, JOB_DESC) VALUES (10000003, NULL, 'William        ', 'Forester       ', 120, 'DRIVER
');
INSERT INTO WORKERS (WORKER_NUM, ID_NUM, FIRST_NAME, LAST_NAME,
SALARY, JOB_DESC) VALUES (10000001, NULL, 'Jane           ', 'Rosser         ', 120, 'SELLER
');
INSERT INTO WORKERS (WORKER_NUM, ID_NUM, FIRST_NAME, LAST_NAME,
SALARY, JOB_DESC) VALUES (10000005, NULL, 'Johan          ', 'Fishered       ', 150,
'MANAGER   ');

COMMIT WORK;
```

SELECT QUERIES

Statement #1

Πόσες αγορες εχουν γινει ανα πελατη

```
select transactions.buyer , count(transactions.transcode)
from transactions
where type = 'SELL'
group by transactions.buyer;

INSERT INTO STATEMENT1 (BUYER, "COUNT") VALUES (100000, 8);
INSERT INTO STATEMENT1 (BUYER, "COUNT") VALUES (100002, 2);

COMMIT WORK;
```

Statement #2

Πόσες πωλησεις εχουν γινει ανα προμηθευτη

```
select transactions.seller , count(transactions.transcode)
from transactions
where type = 'BUY'
group by transactions.seller;

INSERT INTO STATEMENT2 (SELLER, "COUNT") VALUES (500000, 3);
INSERT INTO STATEMENT2 (SELLER, "COUNT") VALUES (500002, 1);

COMMIT WORK;
```

Statement #3

Πόσες συναλαγες εχουν γινει ανα ημερομηνία

```
select transactions."DATE" , count(transactions.transcode)
from transactions
where type = 'SELL'
group by transactions."DATE";

INSERT INTO STATEMENT3 ("DATE", "COUNT") VALUES ('2008-01-23', 10);

COMMIT WORK;
```

Statement #4

Ποσες συναλλαγες εχοθν γινει ανα τυπο (αγορά ή πώληση)

```
select transactions."TYPE" , count(transactions.transcode)
from transactions
group by transactions."TYPE";

INSERT INTO STATEMENT4 ("TYPE", "COUNT") VALUES ('BUY', 4);
INSERT INTO STATEMENT4 ("TYPE", "COUNT") VALUES ('SELL', 10);

COMMIT WORK;
```

Statement #5

Πόσες παραγγελίες εχουν γινει ανα πελατη

```
select orders.payer, count(orders.sn)
from orders
group by orders.payer;

INSERT INTO STATEMENT5 (PAYER, "COUNT") VALUES (100001, 1);
INSERT INTO STATEMENT5 (PAYER, "COUNT") VALUES (100002, 1);

COMMIT WORK;
```

Statement #6

Πόσες παραγγελιες υπαρχουν ανα ημερομηνια

```
select orders.delivery_date, count(orders.sn)
from orders
group by orders.delivery_date;


INSERT INTO STATEMENT3 (DELIVERY_DATE, "COUNT") VALUES ('2008-01-25 00:00:00',
1);
INSERT INTO STATEMENT3 (DELIVERY_DATE, "COUNT") VALUES ('2008-02-03 00:00:00',
1);

COMMIT WORK;
```

Statement #7

Τι ποσοτητα απαιτηται απο καθε αντικειμενο για
τις υπαρχουσες παραγγελιες, ανα αντικειμενο

```
select order_stuff.item, sum(order_stuff.quantity)
from order_stuff, orders
where order_stuff.sn in (select orders.sn from orders where orders.delivery_date > current_date)
group by order_stuff.item;
```

```
INSERT INTO STATEMENT7 (ITEM, "SUM") VALUES (1, 10);
INSERT INTO STATEMENT7 (ITEM, "SUM") VALUES (2, 30);
INSERT INTO STATEMENT7 (ITEM, "SUM") VALUES (3, 10);
INSERT INTO STATEMENT7 (ITEM, "SUM") VALUES (4, 30);
INSERT INTO STATEMENT7 (ITEM, "SUM") VALUES (6, 6);

COMMIT WORK;
```

Statement #8

Πληροφορίες τακτικών μελων (ονοματεπόνυμο
και τηλ. επικοινονιας)

```
select members.first_name, members.last_name, members.cell1
from members
where members.code > 100000
```

```
INSERT INTO STATEMENT8 (FIRST_NAME, LAST_NAME, CELL1) VALUES ('George',
'Bushed', '0343168597');
INSERT INTO STATEMENT8 (FIRST_NAME, LAST_NAME, CELL1) VALUES ('Bill',
'Cilndon', '8543162917');
INSERT INTO STATEMENT8 (FIRST_NAME, LAST_NAME, CELL1) VALUES ('William',
'McCoy', '7163168557');
INSERT INTO STATEMENT8 (FIRST_NAME, LAST_NAME, CELL1) VALUES ('Tony', 'Rebl',
'8543915597');
INSERT INTO STATEMENT8 (FIRST_NAME, LAST_NAME, CELL1) VALUES ('John',
'Rabmo', '8543641297');
INSERT INTO STATEMENT8 (FIRST_NAME, LAST_NAME, CELL1) VALUES ('Michael',
'Jonshon', '3254981369');

COMMIT WORK;
```

Statement #9

Πληροφορίες τακτικών μελων (ονοματεπόνυμο,
τηλ. Επικοινονιας και διευθυνση) [χρηση της order by]

```
select members.last_name, members.first_name, members.cell1, members.address
from members
where members.code > 100000
order by members.last_name

INSERT INTO STATEMENT9 (LAST_NAME, FIRST_NAME, CELL1, ADDRESS) VALUES
('Bushed', 'George', '0343168597', 'GB17, Athens');
INSERT INTO STATEMENT9 (LAST_NAME, FIRST_NAME, CELL1, ADDRESS) VALUES
('Cilndon', 'Bill', '8543162917', 'BC23, Athens');
INSERT INTO STATEMENT9 (LAST_NAME, FIRST_NAME, CELL1, ADDRESS) VALUES
('Jonshon', 'Michael', '3254981369', 'MJ52, Patra');
INSERT INTO STATEMENT9 (LAST_NAME, FIRST_NAME, CELL1, ADDRESS) VALUES
('McCoy', 'William', '7163168557', 'WM39, Athens');
INSERT INTO STATEMENT9 (LAST_NAME, FIRST_NAME, CELL1, ADDRESS) VALUES
('Rabmo', 'John', '8543641297', 'JR32, Salonika');
INSERT INTO STATEMENT9 (LAST_NAME, FIRST_NAME, CELL1, ADDRESS) VALUES
('Rebl', 'Tony', '8543915597', 'TR14, Iwannina');

COMMIT WORK;
```

Statement #10

Πληροφορίες εγαζομενων (ονοματεπόνυμο,
εργασια και μισθος) [χρηση της order by]

```
select workers.last_name, workers.first_name, workers.job_desc, workers.salary
from workers
order by workers.id_num;

INSERT INTO STATEMENT10 (LAST_NAME, FIRST_NAME, JOB_DESC, SALARY)
VALUES ('Stamatiou', 'Iwannis', 'CEO', 170);
INSERT INTO STATEMENT10 (LAST_NAME, FIRST_NAME, JOB_DESC, SALARY)
VALUES ('Lord', 'Steven', 'SELLER', 120);
INSERT INTO STATEMENT10 (LAST_NAME, FIRST_NAME, JOB_DESC, SALARY)
VALUES ('Forester', 'William', 'DRIVER', 120);
INSERT INTO STATEMENT10 (LAST_NAME, FIRST_NAME, JOB_DESC, SALARY)
VALUES ('Rosser', 'Jane', 'SELLER', 120);
INSERT INTO STATEMENT10 (LAST_NAME, FIRST_NAME, JOB_DESC, SALARY)
VALUES ('Fishered', 'Johan', 'MANAGER', 150);

COMMIT WORK;
```

Statement #11

Μεσος ορος μισθων των υπαλληλων

```
select AVG(workers.salary)
from workers

INSERT INTO STATEMENT11 ("AVG") VALUES (136);

COMMIT WORK;
```

Τα δυο επομενα δινουν τον μεσο όρο των τιμών των αντικοιμένων που εχουν τιμες μικρότερες και μεγαλύτερες απο ενα συγκεκριμένο ποσό (εδω 4 και 3 ευρώ αντίστοιχα)

Statement #12

```
select avg(items.price)
from items
where items.keycode in (select items.keycode from items where items.price < 4.00);

INSERT INTO STATEMENT12 ("AVG") VALUES (1.08);

COMMIT WORK;
```

Statement #13

```
select avg(items.price)
from items
where items.keycode in (select items.keycode from items where items.price > 3.00);

INSERT INTO STATEMENT13 ("AVG") VALUES (5);

COMMIT WORK;
```

READ-ME FILE

Στην εργασία αυτη υπαρχει περιορισμος δικαιωμάτων. Για την ακρίβια υπάρχουν 4 χρήστες:

SYSDBA, STAMATIOU, MANAGER, SELLER. Για τους 3 τελευταιους λογαριασμους ο κωδικος ειναι masterkey. Αν εχει γίνει σωστα, θα πρέπει να πληρούνται τα δικαιώματα που αναφέρθηκαν στην Εισαγωγή (Στον υπολογιστή μου τουλαχιστον ετσι δούλευει). Εδωσα δικαιώματα χρησιμοποιώντας τη GRAND στο SQL Editor.

Ακόμα, θα ήθελα να απολογηθώ τόσο για την καθηστέρηση, όσο και για την προχειρότητα της εργασίας (ιδιεταιρα σε αυτό το γραπτό κομμάτι της), αλλα δυστυχώς δεν την δουλεψα καθόλου κατα την διάρκια των διακοπών των Χριστουγέννων, και μάλιστα έχασα και σχεδόν όλη τη δουλειά που είχα κάνει πριν, λογο μιας πολυ σοβαρής βλάβης στο παλιό μου laptop. Τελικα πήρα καινούργιο και ξανάρχισα να την δουλεύω στις 5 Ιανουαρίου.

Τέλος, σε περίπτωση που χρειαστεί (αν και δεν νομίζω, αλλα ποτε δεν ξερει κανεις ί) ο κωδικός για τον SYSDBA είναι 38491024.