ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Project Part B΄

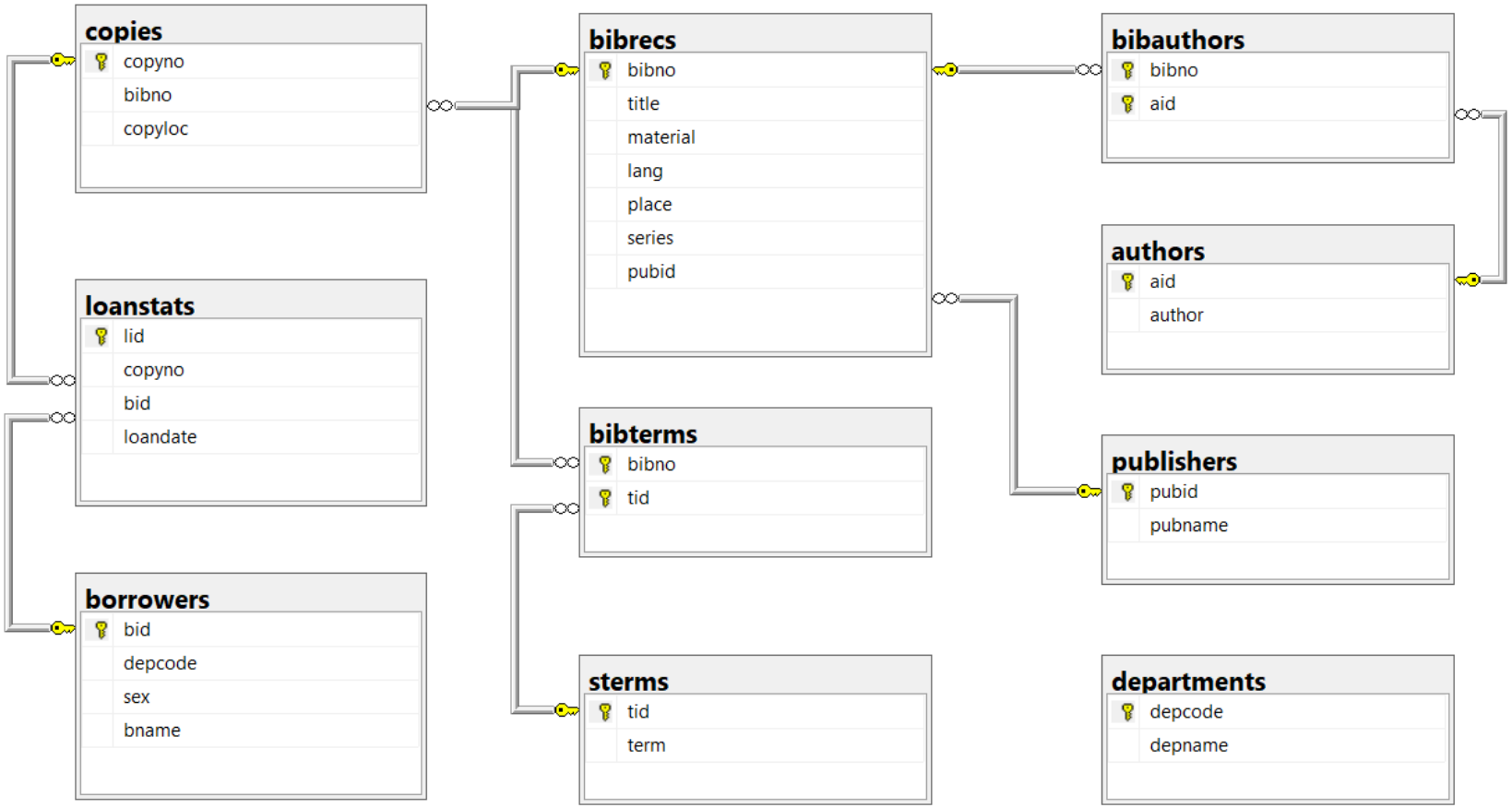
Όνομα: **Παναγιώτης Ντυμένος**

ΑΜ: **3160120**

Περιεχόμενα

Ζήτημα Πρώτο

Έχω την εξής αρχική μορφή της Βάσης Δεδομένων **LIBRARY**:



1.

Δημιούργησα την βάση **LIBDW** στον *SQL SERVER* επομένως ήρθε η ώρα να μετατρέψω την **LIBDW** σε μορφή ***Star Schema***.

2.

Αρχικά να δούμε ποιες διαστάσεις χρειαζόμαστε. Η εκφώνηση αναφέρει πως η διοίκηση της βιβλιοθήκης θέλει να εστιάσει σε στατιστικά που αφορούν τα πεδία ***copyloc***, ***material***, ***depcode*** και ***sex***.

Παρατηρούμε πως χρειαζόμαστε τους πίνακες **Departments**, **Copies**, **Bibrecs**, **Borrowers** οι οποίοι χρειάζονται κάποιες αλλαγές για να μπορούν να ικανοποιήσουν την υλοποίηση του ***Star Schema*** και να μπορέσω να τους ορίσω ως **Dimension Tables**. Επίσης επέλεξα να δημιουργήσω και έναν ακόμα **Dimension Table** που θα έχει την ημερομηνία πιο αναλυτικά(**Dateinfo**). Φυσικά σε κάθε διάσταση θα κρατήσω μόνο τα χαρακτηριστικά που χρειάζομαι.

Ακολουθούν οι δημιουργίες πινάκων βάση του ***Star Schema***:

```
CREATE TABLE departments (
	depcode INT PRIMARY KEY,
	depname VARCHAR(30)
);

CREATE TABLE copies (
	copyno CHAR(8) PRIMARY KEY,
	copyloc CHAR(3)
);

CREATE TABLE borrowers (
	bid INT PRIMARY KEY,
	sex CHAR(1),
);

CREATE TABLE bibrecs (
	bibno INT PRIMARY KEY,
	material VARCHAR(30)
);

CREATE TABLE dateinfo (
	loandate DATE PRIMARY KEY,
	l_year INT,
	l_month INT,
	l_dayofmonth INT,
	l_week INT,
	l_dayofyear INT,
	l_dayofweek INT
);
```

```
CREATE TABLE loans (
	lid INT,
	depcode INT,
	copyno CHAR(8),
	bid INT,
	bibno INT,
	loandate DATE,

	PRIMARY KEY (lid,depcode,copyno,bid,bibno,loandate),
	FOREIGN KEY (depcode) REFERENCES departments(depcode),
	FOREIGN KEY (copyno) REFERENCES copies(copyno),
	FOREIGN KEY (bid) REFERENCES borrowers(bid),
	FOREIGN KEY (bibno) REFERENCES bibrecs(bibno),
	FOREIGN KEY (loandate) REFERENCES dateinfo(loandate)
);
```

Όπως βλέπετε έφτιαξα έναν πίνακα **Loans** ο οποίος έχει όλα τα primary keys των Dimension Tables επομένως και γίνεται **Fact Table**.

Κράτησα μόνο τα attributes που χρειάζεται η βιβλιοθήκη για τα στατιστικά της.

Ως primary key έχω θέσει τον συνδυασμό των primary key όλων των διαστάσεων της βάσης, τα οποία είναι και οι αναφορές των foreign keys του **Loans**.

3.

Ήρθε η ώρα να γεμίσω τους πίνακες της βάσης μου με τα δεδομένα της βάσης **LIBRARY**.

DEPARTMENTS

```
INSERT INTO departments(depcode, depname) SELECT depcode, depname
FROM LIBRARY.dbo.departments;
```

COPIES

```
INSERT INTO copies(copyno, copyloc) SELECT copyno, copyloc FROM LIBRARY.dbo.copies;
```

BORROWERS

```
INSERT INTO borrowers(bid, sex) SELECT bid, sex FROM LIBRARY.dbo.borrowers;
```

BIBRECS

```
INSERT INTO bibrecs(bibno, material) SELECT bibno, material FROM LIBRARY.dbo.bibrecs;
```

DATEINFO

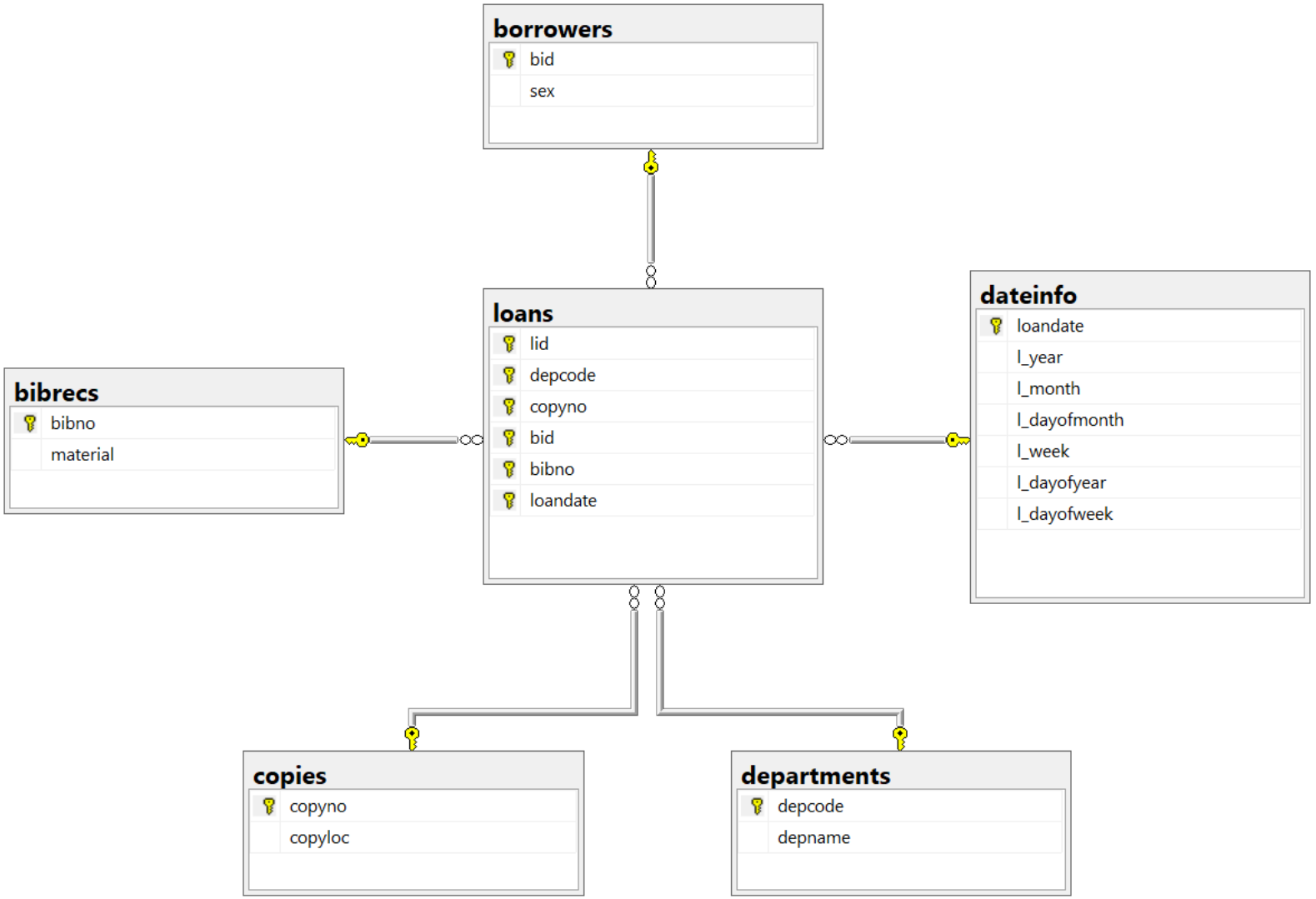
```
SET DATEFIRST 1;
INSERT INTO dateinfo
SELECT DISTINCT loandate, DATEPART(YEAR, loandate), DATEPART(MONTH, loandate),
                          DATEPART(DAY,loandate), DATEPART(WEEK,loandate),
                          DATEPART(DAYOFYEAR,loandate),DATEPART(dw,loandate)
FROM LIBRARY.dbo.loanstats;
```

LOANS

```
INSERT INTO loans(lid, depcode, copyno, bid, bibno, loandate)
SELECT LIBRARY.dbo.loanstats.lid,
             LIBRARY.dbo.departments.depcode,
             LIBRARY.dbo.loanstats.copyno,
             LIBRARY.dbo.borrowers.bid,
             LIBRARY.dbo.bibrecs.bibno,
             LIBRARY.dbo.loanstats.loandate
FROM LIBRARY.dbo.loanstats,
             LIBRARY.dbo.departments,
             LIBRARY.dbo.copies,
             LIBRARY.dbo.borrowers,
             LIBRARY.dbo.bibrecs
WHERE LIBRARY.dbo.loanstats.bid = LIBRARY.dbo.borrowers.bid AND
             LIBRARY.dbo.departments.depcode = LIBRARY.dbo.borrowers.depcode AND
             LIBRARY.dbo.loanstats.copyno = LIBRARY.dbo.copies.copyno AND
             LIBRARY.dbo.bibrecs.bibno = LIBRARY.dbo.copies.bibno;
```

4.

LIBDW STAR-SCHEMA



Ζήτημα Δεύτερο

1.

```
SELECT l_year, departments.depcode, COUNT(lid)
FROM loans
       JOIN dateinfo
       ON dateinfo.loandate = loans.loandate
       JOIN departments
       ON departments.depcode = loans.depcode
GROUP BY l_year, departments.depcode
ORDER BY l_year, departments.depcode
```

2.

```
SELECT copies.copyno, bibrecs.material, COUNT(lid)
FROM loans
       JOIN copies
       ON copies.copyno = loans.copyno
       JOIN bibrecs
       ON bibrecs.bibno = loans.bibno
GROUP BY copies.copyno, bibrecs.material
ORDER BY bibrecs.material
```

3.

```
SELECT l_month, borrowers.sex, COUNT(lid)
FROM loans
       JOIN dateinfo
       ON dateinfo.loandate = loans.loandate
       JOIN borrowers
       ON borrowers.bid = loans.bid
WHERE l_year = '2000'
GROUP BY l_month, borrowers.sex
ORDER BY l_month, borrowers.sex
```

4.

```
SELECT l_year, l_month, COUNT(lid)
FROM loans
       JOIN dateinfo
       ON dateinfo.loandate = loans.loandate
GROUP BY l_year, l_month
HAVING COUNT(lid) > 800
ORDER BY l_year, l_month
```

5.

```
SELECT l_year, department, sex, COUNT(lid) AS total_loans, year_loans,
       year_depcode_loans, year_depcode_sex_loans
FROM loans,
       (SELECT dateinfo.l_year AS l_year, departments.depcode AS department,
       borrowers.sex AS sex, year_loans, year_depcode_loans,
       COUNT(loans.lid) AS year_depcode_sex_loans FROM loans
              JOIN departments
              ON departments.depcode = loans.depcode
              JOIN dateinfo
              ON dateinfo.loandate = loans.loandate
              JOIN borrowers
              ON borrowers.bid = loans.bid

              RIGHT JOIN (SELECT COUNT(lid) as year_loans, l_year FROM loans
                           JOIN dateinfo
                           ON dateinfo.loandate = loans.loandate
                           GROUP BY l_year) AS YEARLY
              ON YEARLY.l_year = dateinfo.l_year

              RIGHT JOIN (SELECT COUNT(lid) as year_depcode_loans, departments.depcode,
                           l_year FROM loans
                           JOIN dateinfo
                           ON dateinfo.loandate = loans.loandate
                           JOIN departments
                           ON departments.depcode = loans.depcode
                           GROUP BY l_year, departments.depcode) AS YEARLY_DEPARTMENT
ON YEARLY_DEPARTMENT.depcode = departments.depcode AND YEARLY_DEPARTMENT.l_year =
dateinfo.l_year
GROUP BY dateinfo.l_year, departments.depcode, borrowers.sex, year_loans,
year_depcode_loans) AS YEARLY_DEPARTMENT_SEX

GROUP BY year_loans, year_depcode_loans, department, sex, l_year, year_depcode_sex_loans
ORDER BY l_year, department, sex
```

*(**ΣΗΜΕΙΩΣΗ**: Ξέρω ότι είναι παράξενο το Query αλλά τρέχει όντως.)*

ΑΚΟΛΟΥΘΕΙ ΔΙΟΡΘΩΣΗ:

Μια καλύτερη λύση:

```
SELECT dateinfo.l_year AS l_year, departments.depcode AS department, borrowers.sex AS
sex, COUNT(lid) AS loans_num FROM loans
JOIN departments
ON departments.depcode = loans.depcode
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY ROLLUP (dateinfo.l_year, departments.depcode, borrowers.sex)
ORDER BY dateinfo.l_year
```

6.

Εφόσον επιτρέπεται, θα δημιουργήσω δύο ξεχωριστούς πίνακες με τους δανεισμούς των φοιτητριών και των φοιτητών αντίστοιχα.

```
SELECT departments.depcode AS depcode, COUNT(lid) AS total_loans
INTO female_loans
FROM loans
       JOIN departments
       ON departments.depcode = loans.depcode
       JOIN borrowers
       ON loans.bid = borrowers.bid
       WHERE borrowers.sex = 'F'
       GROUP BY departments.depcode
```

```
SELECT departments.depcode AS depcode, COUNT(lid) AS total_loans
INTO male_loans
FROM loans
       JOIN departments
       ON departments.depcode = loans.depcode
       JOIN borrowers
       ON loans.bid = borrowers.bid
       WHERE borrowers.sex = 'M'
       GROUP BY departments.depcode
```

Το μόνο που χρειάζεται είναι να συγκρίνω τον αριθμό των δανεισμών για κάθε τμήμα.

```
SELECT female_loans.depcode, female_loans.total_loans
FROM female_loans
       JOIN male_loans
       ON male_loans.depcode = female_loans.depcode
WHERE male_loans.total_loans < female_loans.total_loans
GROUP BY female_loans.depcode, female_loans.total_loans
ORDER BY female_loans.depcode, female_loans.total_loans
```

Ζήτημα Τρίτο

1.

```
SELECT dateinfo.l_year, copies.copyloc, borrowers.sex, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY CUBE (dateinfo.l_year, copies.copyloc, borrowers.sex)
```

2.

Εφόσον θεωρήσουμε πως ο DBMS δεν υποστηρίζει την εντολή **CUBE** θα χρειαστεί να αναλύσω τα αποτελέσματα που πήρα όταν το χρησιμοποίησα στο 1ο ερώτημα. Επομένως για κάθε **GROUP BY** θα φτιάξω και το απαραίτητο Query.

Κανονικά περιμένω πως θα χρειαστώ σύνολο 2^3 **GROUP BY**. (έτος, τοποθεσία, φύλο)

```
1.GROUP BY dateinfo.l_year, copies.copyloc, borrowers.sex
```

```
SELECT dateinfo.l_year, copies.copyloc, borrowers.sex, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY dateinfo.l_year, copies.copyloc, borrowers.sex
```

```
2.GROUP BY dateinfo.l_year, copies.copyloc, ALL
```

```
SELECT dateinfo.l_year, copies.copyloc, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY dateinfo.l_year, copies.copyloc
```

```
3.GROUP BY dateinfo.l_year, ALL, borrowers.sex
```

```
SELECT dateinfo.l_year, borrowers.sex, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY dateinfo.l_year, borrowers.sex
```

4.GROUP BY ALL, copies.copyloc, borrowers.sex

```
SELECT copies.copyloc, borrowers.sex, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY copies.copyloc, borrowers.sex
```

5. GROUP BY dateinfo.l_year, ALL, ALL

```
SELECT dateinfo.l_year, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY dateinfo.l_year
```

6. GROUP BY ALL, copies.copyloc, ALL

```
SELECT copies.copyloc, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY copies.copyloc
```

7. GROUP BY ALL, ALL, borrowers.sex

```
SELECT borrowers.sex, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY borrowers.sex
```

```
8. GROUP BY ALL, ALL, ALL
```

```
SELECT COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
```

Επομένως όντως θα χρειαστώ σύνολο **8** **GROUP BY**.(**7** θεωρητικά αν εξαιρέσουμε το τελευταίο (***none***))

```
Οι παραπάνω 8 εντολές μπορούν να συγχωνευτούν σε ένα Query:
```

```
SELECT dateinfo.l_year, copies.copyloc, borrowers.sex, COUNT(lid)
FROM loans
JOIN dateinfo
ON dateinfo.loandate = loans.loandate
JOIN copies
ON copies.copyno = loans.copyno
JOIN borrowers
ON borrowers.bid = loans.bid
GROUP BY
	GROUPING SETS
	(
		(dateinfo.l_year, copies.copyloc, borrowers.sex),
		(dateinfo.l_year, copies.copyloc),
		(dateinfo.l_year, borrowers.sex),
		(copies.copyloc, borrowers.sex),
		(dateinfo.l_year),
		(copies.copyloc),
		(borrowers.sex),
		()
	)
```

(***ΣΗΜΕΙΩΣΗ***: *Η ανάλυση σε 8 διαφορετικά Queries έγινε μόνο και μόνο για δική μου εξακρίβωση της παραπάνω εντολής, επομένως είπα να τα προσθέσω και στην απάντηση της εργασίας μου από την στιγμή που τα έκανα*)

Ζήτημα Τέταρτο

Η αλήθεια είναι πως υπάρχουν αρκετοί τρόποι να δημιουργηθεί ένα **CUBE** όπως **ROLLUP** (*περίπου*), **UNION ALL** μεταξύ των διαφορετικών **GROUP BY** ή ακόμα και ο τρόπος που έδειξα στο προηγούμενο ερώτημα **GROUPING SETS**. Εγώ πιάστηκα από αυτό που τονίσατε σε ένα φροντιστήριο μετά το μάθημα σχετικά με το ότι καλό θα ήταν να εκμεταλλευτούμε την δημιουργία πινάκων.

Σκεφτόμουν να δημιουργήσω έναν καινούριο **Fact Table** που θα περιέχει ΜΟΝΟ τα πεδία που με ενδιαφέρουν όσον αφορά την δημιουργία του **CUBE**. Επομένως:

```
CREATE TABLE loans_for_cube (
	lid INT,
	copyno CHAR(8),
	bid INT,
	loandate DATE,

	PRIMARY KEY (lid, copyno, bid, loandate),
	FOREIGN KEY (copyno) REFERENCES copies(copyno),
	FOREIGN KEY (bid) REFERENCES borrowers(bid),
	FOREIGN KEY (loandate) REFERENCES dateinfo(loandate)
);


INSERT INTO loans_for_cube SELECT lid, copyno, bid, loandate FROM loans
```

Το μόνο που χρειάζεται είναι να ξανά δημιουργήσω το CUBE με τον καινούριο πίνακα:

```
SELECT dateinfo.l_year, copies.copyloc, borrowers.sex, COUNT(lid)
FROM loans_for_cube
JOIN dateinfo
ON dateinfo.loandate = loans_for_cube.loandate
JOIN copies
ON copies.copyno = loans_for_cube.copyno
JOIN borrowers
ON borrowers.bid = loans_for_cube.bid
GROUP BY
	GROUPING SETS
	(
		(dateinfo.l_year, copies.copyloc, borrowers.sex),
		(dateinfo.l_year, copies.copyloc),
		(dateinfo.l_year, borrowers.sex),
		(copies.copyloc, borrowers.sex),
		(dateinfo.l_year),
		(copies.copyloc),
		(borrowers.sex),
		()
	)
```

(***ΣΗΜΕΙΩΣΗ***: *Θα παρατηρήσετε πως όπου λέτε «δημιουργία» κύβου εγώ απλά τον εμφανίζω και δεν κάνω κάποια δημιουργία πίνακα. Σε περίπτωση που θέλω να δημιουργηθεί θα έκανα την γνωστή διαδικασία με το INSERT INTO*)