ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Β ΦΑΣΗ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2023

ΟΜΑΔΑ 92

Χαρίλαος Καπελώνης 2020030058
Χρήστος Κωσταδήμας 2020030050

Βήμα 1:

Στο παραδοτέο .zip αρχείο επισυνάπτεται το eclipse project που υλοποιεί τις ζητούμενες προδιαγραφές που περιγράφονται από την εκφώνηση.

Βήμα 2:

Στην παρούσα εργαστηριακή άσκηση μας ζητείται από την εκφώνηση να μελετήσουμε το εξής αίτημα σε SQL : Τύπωσε τα ονόματα των πόλεων με πληθυσμό μεγαλύτερο των 50.000 κατοίκων και το πλήθος των φοιτητών που προέρχονται από αυτές και οι οποίοι έχουν συμμετάσχει σε τουλάχιστον δύο (2) προγράμματα πενταετούς διάρκειας και η ημερομηνία εισαγωγής τους είναι μεταξύ 01-09-2040 και 30-09-2050 (πρέπει να εκτυπώνονται όλες οι πόλεις από τον πίνακα Cities, ο οποίος αναφέρεται παρακάτω). Όπως ζητήθηκε από την εκφώνηση στην αναφορά μας σημειώσαμε τα αποτελέσματα της EXPLAIN ANALYSE για τουλάχιστον τρεις (3) εκτελέσεις κάθε ερώτησης.

Ο SQL κώδικας που υλοποιεί το ερώτημα αυτό φαίνεται παρακάτω (έγινε χρήση view):

```
1
2  CREATE OR REPLACE VIEW city_entries AS
3      SELECT person.city_id AS id
4      FROM "Program" program
5          JOIN "Joins" joins USING ("ProgramID")
6          JOIN "Student" student ON (joins."StudentAMKA" = student.amka)
7          JOIN "Person" person USING (amka)
8      WHERE student.entry_date >= '2040-09-01'::date AND student.entry_date <= '2050-09-30'::date
9          AND program."Duration" = 5
10     GROUP BY student.amka, person.city_id
11     HAVING COUNT(student.amka) >= 2;
12
13
14
15 SELECT name, COUNT(city_entries.id) AS student_num
16 FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
17 WHERE population > 50000 ---- BTREE
18 GROUP BY id
19 ORDER BY student_num DESC
20
```

ΜΕΛΈΤΗ - ΜΕΤΡΉΣΕΙΣ ΧΡΌΝΟΥ ΕΚΤΈΛΕΣΗΣ ΕΡΩΤΗΣΗΣ

α) *"Μελετήστε το πλάνο και τους χρόνους εκτέλεσης της ερώτησης χρησιμοποιώντας την EXPLAIN ANALYSE και σημειώστε τα αποτελέσματα και τις παρατηρήσεις σας στην αναφορά σας."*

Κώδικας:

```
1  -- Βήμα 2
2  -- απενεργοποίηση της δυνατότητας δημιουργίας παράλληλων πλάνων εκτέλεσης
3  SET max_parallel_workers_per_gather = 0;
4
5  -- εκτέλεση query
6  EXPLAIN ANALYSE SELECT name, COUNT(city_entries.id) AS student_num
7  FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
8  WHERE population > 50000
9  GROUP BY id
10 ORDER BY student_num DESC
```

Αποτελέσματα:
Εκτέλεση 1 :

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=6079.857..6079.861 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=6314.977..6314.981 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=6087.077..6087.087 rows=19 loops=1)

Παρατηρήσεις:
Για την εκτέλεση του συγκεκριμένου ερωτήματος απαιτούνται γύρω στα 6.100msec και αυτά , χωρίς τη χρήση κανενός ευρετηρίου.

β) *"Στη συνέχεια δοκιμάστε διαδοχικά να δημιουργήσετε κάποιο ή κάποια κατάλληλα ευρετήρια που θεωρείτε ότι μπορεί να επιταχύνει την εκτέλεση του αιτήματος και μελετήστε εκ νέου το πλάνο εκτέλεσης. Τι ευρετήριο(α) επιλέγετε, τι τύπου και γιατί; Δοκιμάστε και σημειώστε στην αναφορά σας τις διαφορές μεταξύ διαφορετικών τύπων ευρετηρίων όπως προκύπτουν από τα αποτελέσματα που σας δίνει η EXPLAIN ANALYSE. Τι παρατηρείτε;"*

Επιλέγουμε αρχικά (γνωρίζοντας όμως από τη θεωρία ότι αυτή η επιλογή δεν είναι η καταλληλότερη) να δημιουργήσουμε ένα ευρετήριο τύπου hash για την ημερομηνία εισαγωγής των φοιτητών, που συμμετέχει σε query όπου εμφανίζονται τελεστές σύγκρισης (π.χ. > , <).

ATTRIBUTE	INDEX TYPE
"Student".entry_date	hash

Κώδικας:

```
10  CREATE INDEX Student_entryDate_idx ON "Student" USING hash(entry_date );
11
12  SET max_parallel_workers_per_gather = 0;
13
14  EXPLAIN ANALYSE SELECT name, COUNT(city_entries.id) AS student_num
15  FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
16  WHERE population > 50000
17  GROUP BY id
18  ORDER BY student_num DESC
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=5817.223..5817.237 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=5834.081..5834.091 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=6125.725..6125.730 rows=19 loops=1)

Παρατηρήσεις:
Για την εκτέλεση αυτή, παρατηρούμε οτι ο απαιτούμενος χρόνος είναι σχεδόν 6.000msec , δηλαδή η προσθήκη του hash ευρετηρίου στο πεδίο entry_date δεν βελτιώνει την “απόδοση”. Το αποτέλεσμα είναι αναμενόμενο εφόσον γνωρίζουμε από τη θεωρία ,πως με χρήση btree θα δούμε πολύ καλύτερα αποτελέσματα εφόσον πρόκειται για query όπου εμφανίζονται τελεστές σύγκρισης (π.χ. > , <).

Έχοντας καταλήξει στο ότι η χρήση τύπου hash ευρετηρίου δεν είναι ικανοποιητική, αποφασίσαμε να χρησιμοποιήσουμε τύπου btree ευρετήριο.

ATTRIBUTE	INDEX TYPE
“Student”.entry_date	btree

Κώδικας:

CREATE INDEX Student_entryDate_idx ON "Student" USING **btree**(entry_date);

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=56506.51..56506.55 rows=17 width=34) (actual time=1995.718..1995.723 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=56506.51..56506.55 rows=17 width=34) (actual time=1893.690..1893.695 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=56506.51..56506.55 rows=17 width=34) (actual time=1991.864..1991.871 rows=19 loops=1)

Παρατηρήσεις :
Για την εκτέλεση του συγκεκριμένου ερωτήματος απαιτούνται γύρω στα 2.000msec άρα τα αποτελέσματα είναι πιο ικανοποιητικά. Χωρίς ευρετήρια είδαμε ότι απαιτούνταν 6.100msec ,άρα συνολικά ο χρόνος εκτέλεσης μειώνεται σημαντικά.

Στη συνέχεια κρατώντας το ευρετήριο για το attribute “Student”.entry_date (δηλ. δεν έγινε DROP INDEX), υλοποιήσαμε ένα ακόμη ευρετήριο για το attribute “Program”.”Duration” , τύπου hash.

ATTRIBUTE	INDEX TYPE
“Student”.entry_date	btree
“Program”."Duration"	hash

Κώδικας:

```
CREATE INDEX Program_duration_idx ON "Program" USING hash("Duration");
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=56506.51..56506.55 rows=17 width=34) (actual time=1930.900..1930.910 rows=19 loops=1)
2	Sort Key: (count(city_entries.id)) DESC

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=56506.51..56506.55 rows=17 width=34) (actual time=1953.586..1953.590 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=56506.51..56506.55 rows=17 width=34) (actual time=1966.194..1966.197 rows=19 loops=1)

Παρατηρήσεις :
Για την εκτέλεση του συγκεκριμένου ερωτήματος απαιτούνται γύρω στα 2.000msec άρα τα αποτελέσματα είναι πιο ικανοποιητικά. Βέβαια η προσθήκη του hash index δεν αλλάζει δραματικά τον απαιτούμενο χρόνο εκτέλεσης.

Έπειτα προσθέτουμε ένα ακόμα ευρετήριο για το μέλος “Cities”.population για τον λόγο ότι έχουμε ranged query.

ATTRIBUTE	INDEX TYPE
“Student”.entry_date	btree
“Program”."Duration"	hash
“Cities”.population	btree

Κώδικας:

```
CREATE INDEX Cities_population_idx ON "Cities" USING btree(population);
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=56502.52..56502.56 rows=17 width=34) (actual time=2090.773..2090.790 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=56502.52..56502.56 rows=17 width=34) (actual time=1963.326..1963.338 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=56502.52..56502.56 rows=17 width=34) (actual time=1940.532..1940.536 rows=19 loops=1)

Παρατηρήσεις :
Για την εκτέλεση του συγκεκριμένου ερωτήματος απαιτούνται γύρω στα 2.000msec άρα τα αποτελέσματα είναι πιο ικανοποιητικά. Βέβαια η επιπλέον προσθήκη του btree index δεν αλλάζει δραματικά τον απαιτούμενο χρόνο εκτέλεσης.

γ) *“Ποιο ευρετήριο(α) θεωρείτε ότι είναι καταλληλότερο;”*
Σαφώς , καταλληλότερο κρίνουμε ότι είναι το btree ευρετήριο. Η απάντηση είναι αναμενόμενη, βάσει της αντίστοιχης θεωρίας που έχουμε διδαχθεί πάνω στα ευρετήρια. Η ορθότητα της απάντησης αυτής εξασφαλίζεται , πέραν της θεωρίας, με σαφήνεια από τα παραπάνω αποτελέσματα για τους χρόνους εκτέλεσης και τις παρατηρήσεις μας.

δ) *"Δοκιμάστε επίσης να επιταχύνετε ενδεχομένως περαιτέρω το αίτημα αξιοποιώντας τη δυνατότητα ομαδοποίησης (clustering). Τι παρατηρείτε; Σημειώστε όλες τις παρατηρήσεις σας στην αναφορά σας."*

Κώδικας :

```
CLUSTER "Student" USING Student_entryDate_idx;
CLUSTER "Program" USING Program_duration_idx;
CLUSTER "Cities" USING Cities_population_idx;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=56502.52..56502.56 rows=17 width=34) (actual time=1768.839..1768.846 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=56502.52..56502.56 rows=17 width=34) (actual time=1849.283..1849.287 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=56502.52..56502.56 rows=17 width=34) (actual time=1926.844..1926.848 rows=19 loops=1)

Παρατηρήσεις:
Στο συγκεκριμένο ερώτημα παρατηρούμε ότι από τις 3 εκτελέσεις του κώδικα, ο ελάχιστος χρόνος εκτέλεσης είναι 1768 msec που είναι σίγουρα μικρότερος από τον ελάχιστο χρόνο εκτέλεσης του παραπάνω ερωτήματος (στο παραπάνω ερώτημα κάναμε όπως φαίνεται 3 εκτελέσεις και ο ελάχιστος χρόνος βρέθηκε ίσος με 1940 msec). Με μια γρήγορη ματιά παρατηρούμε το ίδιο και για τον μέγιστο χρόνο.
Συνεπώς ο χρόνος εκτέλεσης θα μπορούσαμε να αποφανθούμε ότι γενικά μειώνεται, άρα η χρήση των clusters επιταχύνει περεταίρω το αίτημα που εξετάζουμε. Βέβαια για τον χρήστη (εμάς) η βελτίωση αυτού του χρόνου δεν γίνεται αισθητή, όμως υπάρχει, έτσι καταλαβαίνουμε τη χρησιμότητα της εντολής explain analyse.

ε) *"Επιπλέον δοκιμάστε να απενεργοποιήσετε επιλεκτικά αλγορίθμους υπολογισμού συνδέσεων (εφόσον χρησιμοποιούνται) και καταγράψτε στην αναφορά σας τις παρατηρήσεις σας όσον αφορά στην αλλαγή των πλάνων εκτέλεσης και της απόδοσής τους. Αυτό να το κάνετε τόσο πριν όσο και μετά τη δημιουργία ευρετηρίων που θα αποφασίσετε."*

Πρώτα παρουσιάζουμε τα αποτελέσματα ΜΕ ευρετήρια και έπειτα τα αποτελέσματα χωρίς ευρετήρια.

***ΜΕ* ΕΥΡΕΤΗΡΙΑ:**

1η περίπτωση:

Κώδικας:

```
10  SET max_parallel_workers_per_gather = 0;
11  SET enable_hashjoin=off;
12  SET enable_mergejoin=on;
13
14
15  EXPLAIN ANALYSE SELECT name, COUNT(city_entries.id) AS student_num
16  FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
17  WHERE population > 50000
18  GROUP BY id
19  ORDER BY student_num DESC
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=70933.16..70933.20 rows=17 width=34) (actual time=4003.006..4003.009 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=70933.16..70933.20 rows=17 width=34) (actual time=3936.164..3936.187 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=70933.16..70933.20 rows=17 width=34) (actual time=3895.175..3895.180 rows=19 loops=1)

2η περίπτωση:

Εκτέλεση 1:

```
10  SET max_parallel_workers_per_gather = 0;
11  SET enable_hashjoin=off;
12  SET enable_mergejoin=off;
13
14
15  EXPLAIN ANALYSE SELECT name, COUNT(city_entries.id) AS student_num
16  FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
17  WHERE population > 50000
18  GROUP BY id
19  ORDER BY student_num DESC
20
21
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=79969.73..79969.77 rows=17 width=34) (actual time=6094.658..6094.662 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79969.73..79969.77 rows=17 width=34) (actual time=6165.480..6165.484 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79969.73..79969.77 rows=17 width=34) (actual time=6115.898..6115.902 rows=19 loops=1)

***ΧΩΡΙΣ* ΕΥΡΕΤΗΡΙΑ:**

1η περίπτωση:

Εκτέλεση 1:

```
10  SET max_parallel_workers_per_gather = 0;
11  SET enable_hashjoin=off;
12  SET enable_mergejoin=on;
13
14
15  EXPLAIN ANALYSE SELECT name, COUNT(city_entries.id) AS student_num
16  FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
17  WHERE population > 50000
18  GROUP BY id
19  ORDER BY student_num DESC
20
21
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=6112.341..6112.347 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=6182.262..6182.274 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79765.72..79765.76 rows=17 width=34) (actual time=6186.777..6186.780 rows=19 loops=1)

2η περίπτωση

Εκτέλεση 1:

```
10  SET max_parallel_workers_per_gather = 0;
11  SET enable_hashjoin=off;
12  SET enable_mergejoin=off;
13
14
15  EXPLAIN ANALYSE SELECT name, COUNT(city_entries.id) AS student_num
16  FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
17  WHERE population > 50000
18  GROUP BY id
19  ORDER BY student_num DESC
20
21
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=79969.73..79969.77 rows=17 width=34) (actual time=6200.065..6200.073 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79969.73..79969.77 rows=17 width=34) (actual time=6184.416..6184.421 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79969.73..79969.77 rows=17 width=34) (actual time=6153.444..6153.449 rows=19 loops=1)

Παρατηρήσεις :
Στους παρακάτω πίνακες βλέπουμε συγκεντρωμένα τις περιπτώσεις μας και τους χρόνους της κάθε εκτέλεσης που κάναμε για την λήψη μετρήσεων.

ΜΕ ΕΥΡΕΤΗΡΙΑ :

1η περίπτωση enable_hashjoin = off enable_mergejoin = on	2η περίπτωση enable_hashjoin = off enable_mergejoin = off
4003ms	6094ms
3936ms	6165ms
3895ms	6115ms

Άρα συνολικά με τη χρήση ευρρετηρίων, για την περίπτωση enable_hashjoin = off , enable_mergejoin = on απαιτούνται περίπου 4000msec ενώ γιατην περίπτωση enable_hashjoin = off enable_mergejoin = off απαιτούνται περίπου 6.100msec.

ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ :

1η περίπτωση enable_hashjoin = off enable_mergejoin = on	2η περίπτωση enable_hashjoin = off enable_mergejoin = off
6112ms	6200ms
6182ms	6184ms
6186ms	6153ms

Άρα συνολικά χωρίς τη χρήση ευρρετηρίων, για την περίπτωση enable_hashjoin = off , enable_mergejoin = on απαιτούνται περίπου 6.100msec ενώ για την περίπτωση enable_hashjoin = off enable_mergejoin = off απαιτούνται επίσης περίπου 6.100msec. Οι χρόνοι αυτοί μπορούν να θεωρηθούν ίσοι.

Συμπεραίνουμε λοιπόν ότι η επιλογή on/off δεν επηρεάζει σημαντικά το χρόνο εκτέλεσης του ερωτήματος.

στ) *"Τέλος δοκιμάστε, τόσο πριν όσο και μετά τη δημιουργία ευρετηρίων, να αλλάξετε τη σειρά των συνδέσεων και αναφέρατε τις αλλαγές που παρατηρείτε στα πλάνα εκτέλεσης και τους χρόνους."*

Αρχικά ενεργοποιούμε τους αλγόριθμους συνδέσεων με τους οποίους πειραματιστήκαμε στο προηγούμενο ερώτημα:

```
SET enable_hashjoin = ON;
SET enable_mergejoin = ON;
```

ΕΚΔΟΣΗ 1:

```
CREATE OR REPLACE VIEW city_entries AS
    SELECT person.city_id AS id
    FROM "Program" program
        JOIN "Joins" joins USING ("ProgramID")
        JOIN "Student" student ON (joins."StudentAMKA" = student.amka)
        JOIN "Person" person USING (amka)
    WHERE student.entry_date >= '2040-09-01'::date AND student.entry_date <= '2050-09-30'::date
        AND program."Duration" = 5
    GROUP BY student.amka, person.city_id
    HAVING COUNT(student.amka) >= 2;
```

ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=6114.316..6114.319 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=5990.123..5990.126 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=6134.644..6134.648 rows=19 loops=1)

ΜΕ ΕΥΡΕΤΗΡΙΑ :

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1827.376..1827.380 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1968.099..1968.104 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1844.143..1844.150 rows=19 loops=1)

ΕΚΔΟΣΗ 2:

```
CREATE OR REPLACE VIEW city_entries AS
    SELECT person.city_id AS id
    FROM "Joins" joins
        JOIN "Program" program USING ("ProgramID")
        JOIN "Person" person ON (joins."StudentAMKA" = person.amka)
        JOIN "Student" student USING (amka)
            WHERE student.entry_date >= '2040-09-01'::date AND student.entry_date <= '2050-09-30'::date
        AND program."Duration" = 5
    GROUP BY student.amka, person.city_id
    HAVING COUNT(student.amka) >= 2;

```

ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=5888.531..5888.534 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=6010.946..6010.950 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=5998.388..5998.392 rows=19 loops=1)

ΜΕ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1865.236..1865.244 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1750.750..1750.754 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1779.524..1779.534 rows=19 loops=1)

ΕΚΔΟΣΗ 3:

ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ

```
1  CREATE OR REPLACE VIEW city_entries AS
2      SELECT person.city_id AS id
3      FROM (SELECT *
4       FROM "Student"
5       WHERE entry_date >= '2040-09-01'::date AND entry_date <= '2050-09-30'::date
6       ) AS student
7       JOIN "Joins" joins ON (joins."StudentAMKA" = student.amka)
8       JOIN "Program" program ON (program."ProgramID" = joins."ProgramID"
9           AND program."Duration" = 5
10          )
11      JOIN "Person" person USING (amka)
12     GROUP BY student.amka, person.city_id
13     HAVING COUNT(student.amka) >= 2;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=5979.005..5979.009 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=5996.843..5996.847 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=79731.94..79731.99 rows=17 width=34) (actual time=5963.311..5963.315 rows=19 loops=1)

ΜΕ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1813.444..1813.449 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=1939.890..1939.896 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=56468.74..56468.79 rows=17 width=34) (actual time=2015.870..2015.878 rows=19 loops=1)

Συνοψίζουμε τα αποτελέσματα των παραπάνω πειραματισμών στον ακόλουθο πίνακα.

ΕΚΔΟΣΗ	ΕΥΡΕΤΗΡΙΑ (ΜΕ/ΧΩΡΙΣ)	Χρόνος εκτέλεσης 1 (msec)	Χρόνος εκτέλεσης 2 (msec)	Χρόνος εκτέλεσης 3 (msec)	Περίπου συνολικός χρόνος: (msec)
1	ΧΩΡΙΣ	6114	5990	6134	6100
1	ΜΕ	1827	1968	1844	1800
2	ΧΩΡΙΣ	5888	6010	5998	6000
2	ΜΕ	1865	1750	1779	1800
3	ΧΩΡΙΣ	5979	5996	5963	6000
3	ΜΕ	1813	1939	2015	1900

Οι χρόνοι εκτέλεσης δεν αλλάζουν σημαντικά, με την αλλαγή της σειράς των JOIN (π.χ σύγκριση γραμμής 1 με 3 και γραμμής 2 με 4).

Βήμα 3:

ΜΕΛΈΤΗ - ΜΕΤΡΉΣΕΙΣ ΧΡΌΝΟΥ ΕΚΤΈΛΕΣΗΣ ΕΡΩΤΗΣΗΣ

Αφού κατασκευάσαμε κατάλληλες συναρτήσεις δημιουργίας δεδομένων για τους πίνακες Person, Student , Joins και Program , τις χρησιμοποιήσαμε για να αυξήσουμε το πλήθος δεδομένων της βάσης. Σβήσαμε τυχόν ευρετήρια μπορεί να δημιουργήσαμε στο βήμα 2 και επαναλάβαμε τα βήματα του παραπάνω ερωτήματος.

α) *"Μελετήστε το πλάνο και τους χρόνους εκτέλεσης της ερώτησης χρησιμοποιώντας την EXPLAIN ANALYSE και σημειώστε τα αποτελέσματα και τις παρατηρήσεις σας στην αναφορά σας."*

Κώδικας:

```sql
-- Βήμα 3
-- απενεργοποίηση της δυνατότητας δημιουργίας παράλληλων πλάνων εκτέλεσης
SET max_parallel_workers_per_gather = 0;

-- εκτέλεση query
EXPLAIN ANALYSE SELECT name, COUNT(city_entries.id) AS student_num
FROM "Cities" LEFT OUTER JOIN city_entries USING (id)
WHERE population > 50000
GROUP BY id
ORDER BY student_num DESC;
```

Αποτελέσματα:
Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=397225.34..397225.39 rows=17 width=34) (actual time=7592.066..7592.074 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=365325.08..365325.12 rows=17 width=34) (actual time=9650.737..9650.741 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=365325.08..365325.12 rows=17 width=34) (actual time=8582.720..8582.726 rows=19 loops=1)

Παρατηρήσεις:
Για την εκτέλεση του συγκεκριμένου ερωτήματος απαιτούνται περίπου 8.500msec και αυτά , χωρίς τη χρήση κανενός ευρετηρίου. Αυτό το αποτέλεσμα είναι αναμενόμενο εφόσον η βάση περιέχει περισσότερα δεδομένα σε σχέση με αυτά που περιείχε στο βήμα 2,

β) *"Στη συνέχεια δοκιμάστε διαδοχικά να δημιουργήσετε κάποιο ή κάποια κατάλληλα ευρετήρια που θεωρείτε ότι μπορεί να επιταχύνει την εκτέλεση του αιτήματος και μελετήστε εκ νέου το πλάνο εκτέλεσης. Τι ευρετήριο(α) επιλέγετε, τι τύπου και γιατί; Δοκιμάστε και σημειώστε στην αναφορά σας τις διαφορές μεταξύ διαφορετικών τύπων ευρετηρίων όπως προκύπτουν από τα αποτελέσματα που σας δίνει η EXPLAIN ANALYSE. Τι παρατηρείτε;"*

Όπως και στο βήμα 2 επιλέγουμε αρχικά (γνωρίζοντας όμως από τη θεωρία ότι αυτή η επιλογή δεν είναι η καταλληλότερη) να δημιουργήσουμε ένα ευρετήριο τύπου hash για την ημερομηνία εισαγωγής των φοιτητών, που συμμετέχει σε query όπου εμφανίζονται τελεστές σύγκρισης (π.χ. > , <).

ATTRIBUTE	INDEX TYPE
"Student".entry_date	hash

Κώδικας:

```
1  CREATE INDEX Student_entryDate_idx ON "Student" USING hash(entry_date);
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=397050.11..397050.15 rows=17 width=34) (actual time=9755.925..9755.929 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=397050.11..397050.15 rows=17 width=34) (actual time=7211.945..7211.951 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=397050.11..397050.15 rows=17 width=34) (actual time=7464.512..7464.516 rows=19 loops=1)

Παρατηρήσεις:
Όπως φάνηκε και στο βήμα 2 η χρήση hash ευρετηρίου δεν βελτιώνει ιδιαίτερα τα αποτελέσματα.

Έχοντας καταλήξει στο ότι η χρήση τύπου hash ευρετηρίου δεν είναι ικανοποιητική, αποφασίσαμε να χρησιμοποιήσουμε τύπου btree ευρετήριο.

```
1  DROP INDEX Student_entryDate_idx;
```

ATTRIBUTE	INDEX TYPE
“Student”.entry_date	btree

Κώδικας:

```
1  CREATE INDEX Student_entryDate_idx ON "Student" USING btree(entry_date);
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=389718.10..389718.15 rows=17 width=34) (actual time=16388.619..16388.634 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=389718.10..389718.15 rows=17 width=34) (actual time=15343.268..15343.272 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=389718.10..389718.15 rows=17 width=34) (actual time=16322.531..16322.536 rows=19 loops=1)

Παρατηρήσεις :
Για την εκτέλεση του συγκεκριμένου ερωτήματος απαιτείται περισσότερος χρόνος,αυτό κάνει τα αποτελέσματα είναι μην είναι πιο ικανοποιητικά. Αυτό σύμφωνα με τη θεωρία είναι παράλογο. Βέβαια τα δεδομένα που προσθέσαμε είναι πάρα πολλά. Εάν το ευρετήριο B-tree είναι πολύ μεγάλο και δεν χωράει εξ ολοκλήρου στη μνήμη, η πρόσθετη είσοδος/εξόδου του δίσκου που απαιτείται για την πρόσβαση στους κόμβους ευρετηρίου και την ανάκτηση σελίδων δεδομένων ενδέχεται να υπερτερεί των πλεονεκτημάτων από τη χρήση του ευρετηρίου.

Στη συνέχεια κρατώντας το ευρετήριο για το attribute “Student”.entry_date (δηλ. δεν έγινε DROP INDEX), υλοποιήσαμε ένα ακόμη ευρετήριο για το attribute “Program”.”Duration” , τύπου hash.

ATTRIBUTE	INDEX TYPE
“Student”.entry_date	btree
“Program”."Duration"	hash

Κώδικας:

```
1  CREATE INDEX Program_duration_idx ON "Program" USING hash("Duration");
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=388815.10..388815.14 rows=17 width=34) (actual time=15441.041..15441.046 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=388815.10..388815.14 rows=17 width=34) (actual time=15782.823..15782.828 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=388815.10..388815.14 rows=17 width=34) (actual time=15610.998..15611.003 rows=19 loops=1)

Έπειτα προσθέτουμε ένα ακόμα ευρετήριο για το μέλος “Cities”.population για τον λόγο ότι έχουμε ranged query.

ATTRIBUTE	INDEX TYPE
“Student”.entry_date	btree
“Program”."Duration"	hash
“Cities”.population	btree

Κώδικας:

```
1  CREATE INDEX Cities_population_idx ON "Cities" USING btree(population);
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=388811.11..388811.15 rows=17 width=34) (actual time=15731.064..15731.069 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=388811.11..388811.15 rows=17 width=34) (actual time=15556.383..15556.388 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=388811.11..388811.15 rows=17 width=34) (actual time=16172.395..16172.401 rows=19 loops=1)

δ) *"Δοκιμάστε επίσης να επιταχύνετε ενδεχομένως περαιτέρω το αίτημα αξιοποιώντας τη δυνατότητα ομαδοποίησης (clustering). Τι παρατηρείτε; Σημειώστε όλες τις παρατηρήσεις σας στην αναφορά σας."*

Κώδικας :

```
1  CLUSTER "Student" USING Student_entryDate_idx;
2  CLUSTER "Program" USING Program_duration_idx;
3  CLUSTER "Cities" USING Cities_population_idx;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=390475.23..390475.28 rows=17 width=34) (actual time=2757.190..2757.194 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=390475.23..390475.28 rows=17 width=34) (actual time=2845.841..2845.846 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=390475.23..390475.28 rows=17 width=34) (actual time=2941.657..2941.662 rows=19 loops=1)

Παρατηρήσεις:
Όπως αναφέρθηκε και νωρίτερα τα δεδομένα που εισήχθησαν στη βάση ήταν πάρα πολλά. Έτσι
Το clustering αναδιατάσσει τη φυσική σειρά των δεδομένων στο δίσκο με βάση το κλειδί ευρετηρίου, το οποίο μπορεί να βοηθήσει στη βελτίωση της τοποθεσίας δεδομένων και στη μείωση της εισόδου/εξόδου του δίσκου.
Ομαδοποιώντας τον πίνακα με βάση τη στήλη ερωτήματος εύρους, η βάση δεδομένων μπορεί να επιτύχει πιο αποτελεσματική πρόσβαση στα δεδομένα που εμπίπτουν στο καθορισμένο εύρος. Αυτό μπορεί να οδηγήσει σε ταχύτερη ανάκτηση των απαιτούμενων σειρών, με αποτέλεσμα βελτιωμένο χρόνο εκτέλεσης ερωτήματος.

Συνεπώς ο χρόνος εκτέλεσης γενικά μειώνεται, άρα η χρήση των clusters επιταχύνει περεταίρω το αίτημα που εξετάζουμε.

ε) *"Επιπλέον δοκιμάστε να απενεργοποιήσετε επιλεκτικά αλγορίθμους υπολογισμού συνδέσεων (εφόσον χρησιμοποιούνται) και καταγράψτε στην αναφορά σας τις παρατηρήσεις σας όσον αφορά στην αλλαγή των πλάνων εκτέλεσης και της απόδοσής τους. Αυτό να το κάνετε τόσο πριν όσο και μετά τη δημιουργία ευρετηρίων που θα αποφασίσετε."*

Πρώτα παρουσιαζουμε τα αποτελέσματα ΜΕ ευρετήρια και έπειτα τα αποτελέσματα χωρίς ευρετήρια.

ΜΕ **ΕΥΡΕΤΗΡΙΑ:**

1η περίπτωση:

Κώδικας:

```
6  SET enable_hashjoin = OFF;
7  SET enable_mergejoin = ON;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=801902.23..801902.27 rows=17 width=34) (actual time=51101.126..51101.130 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=801902.23..801902.27 rows=17 width=34) (actual time=53192.878..53192.882 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=801902.23..801902.27 rows=17 width=34) (actual time=52031.493..52031.497 rows=19 loops=1)

2η περίπτωση:
Κώδικας:

```
6  SET enable_hashjoin = OFF;
7  SET enable_mergejoin = OFF;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=1016996.37..1016996.41 rows=17 width=34) (actual time=5061.108..5061.113 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=1016996.37..1016996.41 rows=17 width=34) (actual time=4982.828..4982.833 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=1016996.37..1016996.41 rows=17 width=34) (actual time=5061.916..5061.934 rows=19 loops=1)

***ΧΩΡΙΣ* ΕΥΡΕΤΗΡΙΑ:**

1η περίπτωση:
Κώδικας:

```
6  SET enable_hashjoin = OFF;
7  SET enable_mergejoin = ON;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=809039.10..809039.15 rows=17 width=34) (actual time=72540.964..72540.968 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=809039.10..809039.15 rows=17 width=34) (actual time=72554.059..72554.062 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=809039.10..809039.15 rows=17 width=34) (actual time=69839.371..69839.375 rows=19 loops=1)

2η περίπτωση

1η περίπτωση:
Κώδικας:

```
6  SET enable_hashjoin = OFF;
7  SET enable_mergejoin = OFF;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=1024129.25..1024129.30 rows=17 width=34) (actual time=23426.849..23426.877 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
	Sort (cost=1024129.25..1024129.30 rows=17 width=34) (actual time=23183.688..23183.692 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=1024129.25..1024129.30 rows=17 width=34) (actual time=25553.882..25553.885 rows=19 loops=1)

Παρατηρήσεις :
Στους παρακάτω πίνακες βλέπουμε συγκεντρωμένα τις περιπτώσεις μας και τους χρόνους της κάθε εκτέλεσης που κάναμε για την λήψη μετρήσεων.

ΜΕ ΕΥΡΕΤΗΡΙΑ :

1η περίπτωση enable_hashjoin = OFF enable_mergejoin = ON	2η περίπτωση enable_hashjoin = OFF enable_mergejoin = OFF
51101ms	5061ms
53192ms	4982ms
52031ms	5061ms

Άρα συνολικά με τη χρήση ευρετηρίων, για την περίπτωση enable_hashjoin = off , enable_mergejoin = on απαιτούνται περίπου 51.000msec ενώ για την περίπτωση enable_hashjoin = off enable_mergejoin = off απαιτούνται περίπου 5.000msec.

ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ :

1η περίπτωση enable_hashjoin = OFF enable_mergejoin = ON	2η περίπτωση enable_hashjoin = OFF enable_mergejoin = OFF
72540ms	23426ms
72554ms	23183ms
69839ms	25553ms

Άρα συνολικά χωρίς τη χρήση ευρετηρίων, για την περίπτωση enable_hashjoin = off , enable_mergejoin = on απαιτούνται περίπου 71.000msec ενώ για την περίπτωση enable_hashjoin = off enable_mergejoin = off απαιτούνται επίσης περίπου 23.000msec.

στ) *"Τέλος δοκιμάστε, τόσο πριν όσο και μετά τη δημιουργία ευρετηρίων, να αλλάξετε τη σειρά των συνδέσεων και αναφέρατε τις αλλαγές που παρατηρείτε στα πλάνα εκτέλεσης και τους χρόνους."*

Αρχικά ενεργοποιούμε τους αλγόριθμους συνδέσεων με τους οποίους πειραματιστήκαμε στο προηγούμενο ερώτημα:

```
SET enable_hashjoin = ON;
SET enable_mergejoin = ON;
```

ΕΚΔΟΣΗ 1:

```
CREATE OR REPLACE VIEW city_entries AS
    SELECT person.city_id AS id
    FROM "Program" program
        JOIN "Joins" joins USING ("ProgramID")
        JOIN "Student" student ON (joins."StudentAMKA" = student.amka)
        JOIN "Person" person USING (amka)
    WHERE student.entry_date >= '2040-09-01'::date
        AND student.entry_date <= '2050-09-30'::date
        AND program."Duration" = 5
    GROUP BY student.amka, person.city_id
    HAVING COUNT(student.amka) >= 2;
```

ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=7216.630..7216.637 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=7418.150..7418.156 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=7821.690..7821.700 rows=19 loops=1)

ΜΕ ΕΥΡΕΤΗΡΙΑ :

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3008.596..3008.601 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=2888.990..2888.995 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3279.432..3279.438 rows=19 loops=1)

ΕΚΔΟΣΗ 2:

```
CREATE OR REPLACE VIEW city_entries AS
    SELECT person.city_id AS id
    FROM "Joins" joins
        JOIN "Program" program USING ("ProgramID")
        JOIN "Person" person ON (joins."StudentAMKA" = person.amka)
        JOIN "Student" student USING (amka)
    WHERE student.entry_date >= '2040-09-01'::date
        AND student.entry_date <= '2050-09-30'::date
        AND program."Duration" = 5
    GROUP BY student.amka, person.city_id
    HAVING COUNT(student.amka) >= 2;
```

ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=7908.215..7908.222 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=8200.281..8200.288 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=8101.233..8101.237 rows=19 loops=1)

ΜΕ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3222.369..3222.374 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3242.277..3242.282 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3279.406..3279.421 rows=19 loops=1)

ΕΚΔΟΣΗ 3:
ΧΩΡΙΣ ΕΥΡΕΤΗΡΙΑ

```
CREATE OR REPLACE VIEW city_entries AS
    SELECT person.city_id AS id
    FROM (
        SELECT *
        FROM "Student"
        WHERE entry_date >= '2040-09-01'::date AND entry_date <= '2050-09-30'::date
    ) AS student
        JOIN "Joins" joins ON (joins."StudentAMKA" = student.amka)
        JOIN "Program" program ON (
            program."ProgramID" = joins."ProgramID"
            AND program."Duration" = 5
        )
        JOIN "Person" person USING (amka)
    GROUP BY student.amka, person.city_id
    HAVING COUNT(student.amka) >= 2;
```

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=7421.941..7421.945 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=7499.346..7499.351 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=397280.01..397280.05 rows=17 width=34) (actual time=7361.474..7361.479 rows=19 loops=1)

ΜΕ ΕΥΡΕΤΗΡΙΑ:

Εκτέλεση 1:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3203.132..3203.137 rows=19 loops=1)

Εκτέλεση 2:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3423.879..3423.884 rows=19 loops=1)

Εκτέλεση 3:

	QUERY PLAN text
1	Sort (cost=389240.13..389240.18 rows=17 width=34) (actual time=3246.389..3246.422 rows=19 loops=1)

Συνοψίζουμε τα αποτελέσματα των παραπάνω πειραματισμών στον ακόλουθο πίνακα.

ΕΚΔΟΣΗ	ΕΥΡΕΤΗΡΙΑ (ΜΕ/ΧΩΡΙΣ)	Χρόνος εκτέλεσης 1 (msec)	Χρόνος εκτέλεσης 2 (msec)	Χρόνος εκτέλεσης 3 (msec)	Περίπου συνολικός χρόνος: (msec)
1	ΧΩΡΙΣ	7216	7418	7821	7400
1	ΜΕ	3008	2888	3279	3000
2	ΧΩΡΙΣ	7908	8200	8101	8000
2	ΜΕ	3222	3242	3279	3200
3	ΧΩΡΙΣ	7421	7499	7361	7400
3	ΜΕ	3203	3423	3246	3200

Οι χρόνοι εκτέλεσης δεν αλλάζουν σημαντικά, με την αλλαγή της σειράς των JOIN (π.χ σύγκριση γραμμής 1 με 3 και γραμμής 2 με 4).