Συστήματα Διαχείρισης & Ανάλυσης Δεδομένων
2η Ομάδα Ασκήσεων

Αλβιόνα Μάντσο
3200098

Άσκηση 1

(A)

Δίνονται τα εξής:

R(A,B,C,D,E,F)

T(R) = 1000000

V(R,A) = 10000 **V**(R,B) = 1000 **V**(R,F) = 10

```
SELECT * FROM R WHERE A=1652 AND B>500 AND F>2
```

Εφόσον υπάρχουν V(R,A) διακριτές τιμές στο γνώρισμα A και είναι ομοιόμορφα κατανεμημένες, η πιθανότητα εμφάνισης εγγραφών με τιμή 1652 είναι $\frac{1}{V(R,A)} = \frac{1}{10000}$. Συνεπώς:

$$\frac{1}{V(R,A)} * 1000000 = \frac{1000000}{10000} = 100$$

εγγραφές στον πίνακα R έχουν τιμή 1652 στο R.A

X

Ανεξαρτησία τιμών, αλλά T(R) = 100 < V(R,B) = 1000

Λόγω ομοιόμορφης κατανομής, η πιθανότητα εμφάνισης εγγραφών με τιμή >500 στον πίνακα R είναι $\frac{(1000-501+1)}{1000} = \frac{1}{2}$, δηλ. οι μισές εγγραφές έχουν τιμή [501..1000] στο γνώρισμα B. Λόγω ανεξαρτησίας, η πιθανότητα αυτή διατηρείται στον πίνακα X. Συνεπώς:

$$\frac{1}{2} * T(X) = \frac{1}{2} * 100 = 50$$

εγγραφές στον ενδιάμεσο πίνακα X έχουν τιμή >500 στο X.B

Εφόσον υπάρχουν V(Y,F) διακριτές τιμές στο γνώρισμα F και είναι ομοιόμορφα κατανεμημένες, η πιθανότητα εμφάνισης εγγραφών με τιμή [2..10] είναι $\frac{(10-3+1)}{V(Y,F)} = \frac{8}{10}$. Συνεπώς:

$$\frac{8}{10} * T(Y) = \frac{8}{10} * 50 = 40$$

εγγραφές στον ενδιάμεσο πίνακα Y έχουν τιμή >2 στο Y.F

Y

Ανεξαρτησία τιμών

⇨ V(Y,F) = V(X,F) = V(R,F) = 10

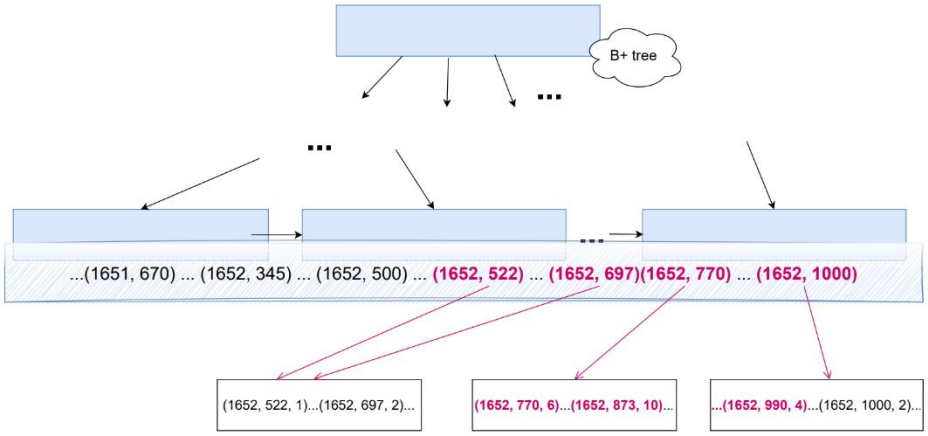
Συνεπώς **40 εγγραφές** του πίνακα R υπολογίζεται να ικανοποιούν τη συνθήκη A=1652 AND B>500 AND F>2 και επιστρέφονται στην έξοδο του επερωτήματος.

(2)

Πληροφορούμαστε ότι το ευρετήριο βρίσκεται στη μνήμη. Επομένως το κόστος σε I/O για τη διάσχιση του ευρετηρίου είναι 0. Αφού γίνει η διάσχιση του B+ δέντρου και καταλήξουμε στα

φύλλα, θα υπάρχει ένα φύλλο που θα αναφέρεται (μεταξύ άλλων) στην σελίδα της πρώτης εγγραφής με (A,B) = (1652, >500). Διατρέχοντας με τη βοήθεια των δεικτών τα φύλλα που βρίσκονται δεξιότερα από αυτό, θα εξετάσουμε όλα εκείνα που αναφέρονται σε έστω και μία εγγραφή με A=1652 και B>500, Δεν μπορούμε να γνωρίζουμε εκ των προτέρων ποια από αυτά αφορούν εγγραφές με F>2 γιατί το κλειδί αναζήτησης του ευρετηρίου είναι στα γνωρίσματα (A,B).

Παραπάνω υπολογίσαμε ότι υπάρχουν 50 εγγραφές για τις οποίες ισχύει A=1652 και B>500. Εφόσον μια σελίδα χωράει 20 εγγραφές της σχέσης R, απαιτούνται $\left\lceil \frac{50}{20} \right\rceil = 3$ σελίδες για να φιλοξενήσουν τις εγγραφές αυτές. Οι δείκτες, λοιπόν, των φύλλων που εξετάζουμε θα δείχνουν τελικά σε 3 σελίδες (εφόσον το ευρετήριο είναι συσταδοποιημένο). Συνεπώς θα χρειαστούν 3 I/Os για να ανακτηθούν οι σελίδες αυτές.



Άσκηση 2

ΟΜΑΔΑ Α	
Συχνότητα εκτέλεσης επερωτήματος	Επερώτημα
100000	E1. SELECT * FROM T WHERE B < ?
10000	E2. SELECT * FROM T WHERE C= ?

Προτείνεται η δημιουργία των εξής ευρετηρίων:

1. Συσταδοποιημένο (clustered) ευρετήριο μορφής B+ δέντρου με κλειδί αναζήτησης το B: Επιτάχυνση του E1.
   - Επιλέγεται ως κλειδί το γνώρισμα που εμφανίζεται στη συνθήκη του WHERE ώστε να επιταχυνθεί ο έλεγχος της συνθήκης, που είναι και ο βασικός φόρτος υπολογισμού του επερωτήματος.
   - Το ευρετήριο είναι συσταδοποιημένο (clustered) για να επιταχυνθεί η εκτέλεση του επερωτήματος. Εφόσον πρόκειται για range query, οι σελίδες που θα πρέπει να ανακτηθούν από τον δίσκο ενδέχεται να είναι πολλές (δεδομένου και του μεγάλου μεγέθους της σχέσης T). Συνεπώς είναι συμφέρουσα η εξασφάλιση σειριακών προσπελάσεων για την εκτέλεση τέτοιου είδους επερωτημάτων συγκριτικά με τις πιο χρονοβόρες τυχαίες προσπελάσεις στον δίσκο. Η ανάγκη αυτή εντείνεται λόγω της συχνότητας με την οποία εκτελείται το επερώτημα (100000).

- Το ευρετήριο υλοποιείται ως B+ δέντρο, δεδομένου ότι πρόκειται για range query. Με την αξιοποίηση των δεικτών των φύλλων του B+ δέντρου το range query επιταχύνεται. Αντιθέτως το Hash Index δεν θα ήταν βοηθητικό εδώ εφόσον δεν είναι χρήσιμο όταν πρόκειται για range queries.

2. Μη συσταδοποιημένο (non-clustered) ευρετήριο μορφής Hash με κλειδί αναζήτησης το C: Επιτάχυνση του E2.
   - Επιλέγεται ως κλειδί το γνώρισμα που εμφανίζεται στη συνθήκη του WHERE ώστε να επιταχυνθεί ο έλεγχος της συνθήκης, που είναι και ο βασικός φόρτος υπολογισμού του επερωτήματος.
   - Το ευρετήριο είναι μη συσταδοποιημένο (non-clustered) εφόσον προηγουμένως ορίστηκε ένα clustered ευρετήριο για την ίδια σχέση Τ. (Δεν θα μπορούσαμε να έχουμε δύο διαφορετικά clustered indexes επί της ίδιας σχέσης Τ, εφόσον ένα clustered index καθορίζει τη θέση των εγγραφών της σχέσης στον δίσκο, η οποία είναι μία και μοναδική). Επιλέχθηκε clustered ευρετήριο με προτεραιότητα προηγουμένως λόγω της μεγαλύτερης συχνότητας εκτέλεσης και της φύσης (range query) του επερωτήματος.
   - Το ευρετήριο υλοποιείται ως Hash Index, που είναι προτιμότερο για point queries. Το B+ δέντρο θα μπορούσε να λειτουργήσει εδώ, αλλά το Hash Index είναι εν γένει καλύτερο (εν γένει απαιτείται 1I/O για την αναζήτηση με Hash Index (χωρίς overflow pages) έναντι h I/Os για το B+ δέντρο, όπου h το ύψος του δέντρου).

ΟΜΑΔΑ Β	
Συχνότητα εκτέλεσης επερωτήματος	Επερώτημα
100000	E1. SELECT * FROM T WHERE B < ? AND C=?
10000	E2. SELECT * FROM T WHERE D=?
1000	E3. SELECT * FROM T WHERE A=?

Προτείνεται η δημιουργία των εξής ευρετηρίων:

1. Συσταδοποιημένο (clustered) ευρετήριο μορφής B+ δέντρου με κλειδί αναζήτησης το ζεύγος (C, B): Επιτάχυνση του E1.
   - Επιλέγεται ως κλειδί το ζεύγος των γνωρισμάτων που εμφανίζεται στη συνθήκη του WHERE, όπως και προηγουμένως. Ο λόγος που το κλειδί ορίζεται στο ζεύγος (C,B) με τη συγκεκριμένη σειρά των γνωρισμάτων είναι ότι εν γένει αναμένεται η συνθήκη C = ? να είναι πιο περιοριστική από την B < ?, δηλαδή συχνότερα θα είναι λιγότερες οι εγγραφές που ικανοποιούν την C = ?. Έτσι, περιορίζοντας από νωρίς τις εγγραφές που αξίζει να εξεταστούν, επιταχύνεται η αναζήτηση στο ευρετήριο (το οποίο δεν έχουμε την εγγύηση ότι πάντοτε θα βρίσκεται στη μνήμη). Η υπόθεση ότι η συνθήκη C = ? θα είναι πιο περιοριστική δεν θα επαληθεύεται προφανώς για όλες τις τιμές που λαμβάνουν τα «?», αλλά είναι στατιστικά πιθανότερο να συμβεί αυτό.
   - Το ευρετήριο είναι συσταδοποιημένο (clustered) για να επιταχυνθεί η εκτέλεση του επερωτήματος, εφόσον πρόκειται για range query που μάλιστα εκτελείται με μεγάλη συχνότητα (συλλογισμός όμοιος με το Α.1).

- Το ευρετήριο υλοποιείται ως B+ δέντρο, δεδομένου ότι πρόκειται για range query και εδώ.

Απομένει ένα διαθέσιμο ευρετήριο να κατασκευαστεί. Το ευρετήριο αυτό δεν μπορεί να επιταχύνει με κάποιον τρόπο τα E2 και E3 (και τα δύο), οπότε η επιλογή γίνεται με κριτήριο τη συχνότητα εκτέλεσης των επερωτημάτων, και συνεπώς δίνεται προτεραιότητα στην επιτάχυνση του E2:

2. Μη συσταδοποιημένο (non-clustered) ευρετήριο μορφής Hash με κλειδί αναζήτησης το D:
   - Επιλέγεται ως κλειδί το ζεύγος των γνωρισμάτων που εμφανίζεται στη συνθήκη του WHERE, όπως και προηγουμένως.
   - Το ευρετήριο είναι μη συσταδοποιημένο (non-clustered) εφόσον προηγουμένως ορίστηκε ένα clustered ευρετήριο για την ίδια σχέση T. Επιλέχθηκε clustered ευρετήριο με προτεραιότητα προηγουμένως λόγω της μεγαλύτερης συχνότητας εκτέλεσης και της φύσης (range query) του επερωτήματος.
   - Το ευρετήριο υλοποιείται ως Hash Index, που είναι προτιμότερο για point queries συγκριτικά με τα B+ δέντρα.

ΟΜΑΔΑ Γ	
Συχνότητα εκτέλεσης επερωτήματος	Επερώτημα
100000	E1. SELECT A,C FROM T WHERE B < ?
10000	E2. SELECT * FROM T WHERE D < ?

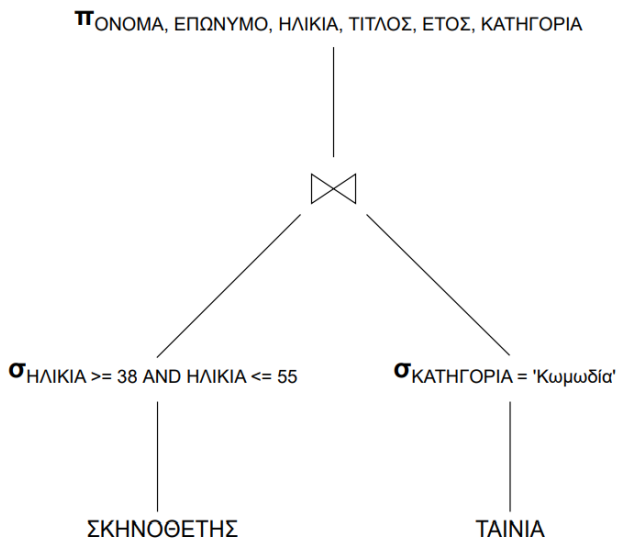
Προτείνεται η δημιουργία των εξής ευρετηρίων:

1. Συσταδοποιημένο (clustered) ευρετήριο μορφής B+ δέντρου με κλειδί αναζήτησης το B: Επιτάχυνση του E1.
   - Επιλέγεται ως κλειδί το γνώρισμα που εμφανίζεται στη συνθήκη του WHERE ώστε να επιταχυνθεί ο έλεγχος της συνθήκης, που είναι και ο βασικός φόρτος υπολογισμού του επερωτήματος. (Σημειώνεται ότι τα γνωρίσματα A,C που εμφανίζονται στο select δεν επηρεάζουν την επιλογή του κλειδιού.)
   - Το ευρετήριο είναι συσταδοποιημένο (clustered) για να επιταχυνθεί η εκτέλεση του επερωτήματος, εφόσον πρόκειται για range query που μάλιστα εκτελείται με μεγάλη συχνότητα (συλλογισμός όμοιος με το A.1).
   - Το ευρετήριο υλοποιείται ως B+ δέντρο, δεδομένου ότι πρόκειται για range query και εδώ.

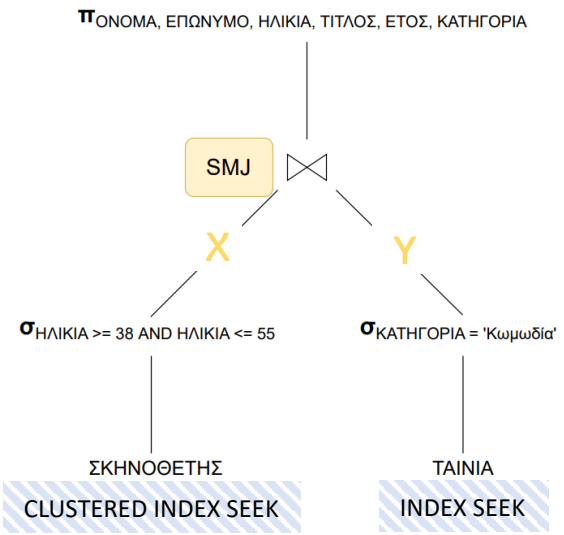
2. Μη συσταδοποιημένο (non-clustered) ευρετήριο μορφής μορφής B+ δέντρου με κλειδί αναζήτησης το D:
   - Επιλέγεται ως κλειδί το γνώρισμα που εμφανίζεται στη συνθήκη του WHERE, όπως και προηγουμένως.
   - Το ευρετήριο είναι μη συσταδοποιημένο (non-clustered) εφόσον προηγουμένως ορίστηκε ένα clustered ευρετήριο για την ίδια σχέση T. Επιλέχθηκε clustered ευρετήριο με προτεραιότητα προηγουμένως λόγω της μεγαλύτερης συχνότητας εκτέλεσης του επερωτήματος.
   - Το ευρετήριο υλοποιείται ως B+ δέντρο, δεδομένου ότι πρόκειται για range query και εδώ.

Άσκηση 3

(A)



(B) Αρχικά πρέπει να υπολογίσουμε το μέγεθος **T(.)** των ενδιάμεσων αποτελεσμάτων **X,Y**.



· **σ**ΗΛΙΚΙΑ >= 38 AND ΗΛΙΚΙΑ <= 55 (ΣΚΗΝΟΘΕΤΗΣ) = **X**

Για τον υπολογισμό του **T(X)** έχουμε στη διάθεσή μας ιστόγραμμα (για το γνώρισμα ΗΛΙΚΙΑ) το οποίο και αξιοποιούμε:

Ηλικία	Αριθμός Εγγραφών
[20..29]	500
[30..39]	1000
[40..49]	1500
[50..59]	500
[60..69]	500

$$\mathbf{T(X)} = (39-38+1)*\frac{1000}{39-30+1} + 1500 + (55-50+1)*\frac{500}{59-50+1}$$

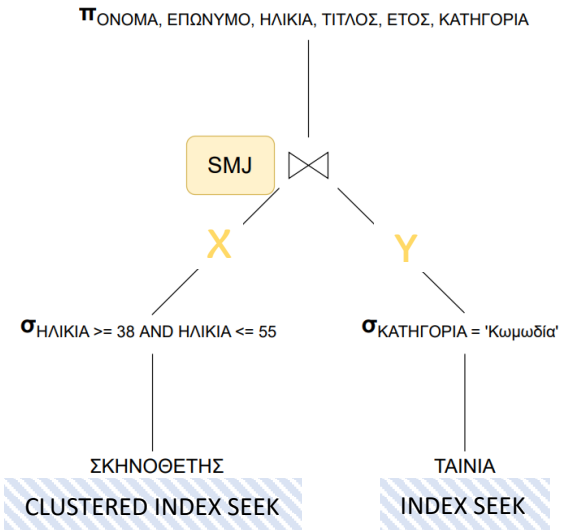
$$= 2*100 + 1500 + 6*50 = 2000$$

· **σ**ΚΑΤΗΓΟΡΙΑ = 'Κωμωδία' (ΤΑΙΝΙΑ) = **Y**

Για τον υπολογισμό του **T(Y)** έχουμε στη διάθεσή μας τις πληροφορίες **T**(ΤΑΙΝΙΑ) = 20000 **V**(ΤΑΙΝΙΑ, ΚΑΤΗΓΟΡΙΑ) = 10 τις οποίες και αξιοποιούμε:

$$\mathbf{T(Y)} = \frac{T(TAINIA)}{V(TAINIA,KATHГOPIA)} = \frac{20000}{10} = 2000$$

· Σε μία σελίδα χωράνε 40 εγγραφές της σχέσης ΣΚΗΝΟΘΕΤΗΣ, συνεπώς **B(X)** = 2000/40 = 50.
· Ομοίως, σε μια σελίδα χωράνε 20 εγγραφές της σχέσης ΤΑΙΝΙΑ, οπότε **B(Y)** = 2000/20 = 100.



T(X) = 2000, B(X) = 50

T(Y) = 2000, B(Y) = 100

M = 16

Ο αλγόριθμος SMJ απαιτεί οι πίνακες που συμμετέχουν στη σύζευξη (εδώ οι X,Y) να είναι ταξινομημένοι. Εφόσον λοιπόν οι X,Y δεν είναι ήδη ταξινομημένοι, θα πρέπει να ταξινομηθούν. Θα εξετάσουμε εάν δεδομένων των μεγεθών (των πινάκων και της μνήμης) μπορεί να τρέξει η αποδοτική έκδοση του SMJ ή η λιγότερο αποδοτική.

1) B(X) > M, B(Y) > M, επομένως η ταξινόμηση κανενός εκ των δύο πινάκων δεν μπορεί να γίνει απευθείας στη μνήμη (με γνωστούς αλγορίθμους ταξινόμησης όπως QuickSort, MergeSort κ.λπ.).

2) Ο X θα δημιουργήσει $\left\lceil \frac{B(X)}{M} \right\rceil = \left\lceil \frac{50}{16} \right\rceil = 4$ υπολίστες.

Ο Y αντίστοιχα θα δημιουργήσει $\left\lceil \frac{B(Y)}{M} \right\rceil = \left\lceil \frac{100}{16} \right\rceil = 7$ υπολίστες.

Συνολικά λοιπόν δημιουργούνται 4+7 = 11 < 16 υπολίστες που μπορούν να συγχωνευθούν μαζί στη συνέχεια. Άρα μπορεί να τρέξει **η αποδοτική έκδοση** του SMJ που περιλαμβάνει τη δημιουργία των υπολιστών για κάθε πίνακα, και στη συνέχεια τη συγχώνευση-σύζευξη των εγγραφών τους).

· Συνεπώς, **COST** = COST(X) + 2*B(X) + COST(Y) + 2*B(Y)

δημιουργία sublists → 1 write → 1 read → συγχώνευση - σύζευξη

· *Για το COST(X):*

σΗΛΙΚΙΑ >= 38 AND ΗΛΙΚΙΑ <= 55 (ΣΚΗΝΟΘΕΤΗΣ) = **X**

Πληροφορούμαστε ότι υπάρχει ευρετήριο συστάδων (clustered index) B+ δέντρο στο γνώρισμα ΣΚΗΝΟΘΕΤΗΣ.ΗΛΙΚΙΑ, το οποίο βρίσκεται στη μνήμη. Το κόστος σε I/O για να διατρέξουμε το δέντρο μέχρι να καταλήξουμε στα φύλλα του είναι λοιπόν 0.
Εφόσον πρόκειται για εύρος τιμών, μόλις καταλήξουμε στο φύλλο που περιλαμβάνει το κλειδί 38, θα πρέπει να διατρέξουμε και τα δεξιότερα από αυτό φύλλα με τα οποία συνδέεται (ακολουθώντας τους δείκτες του B+ δέντρου), μέχρι και το πρώτο κλειδί με τιμή >55. Στη διαδρομή αυτή, για κάθε δείκτη που συναντούμε προς σελίδα στον δίσκο θα πρέπει να ανακτούμε τη σελίδα αυτή καθώς περιλαμβάνει εγγραφές που ικανοποιούν τη συνθήκη ΗΛΙΚΙΑ >= 38 AND ΗΛΙΚΙΑ <= 55 . Λόγω τώρα του ότι το ευρετήριο είναι συσταδοποιημένο (clustered), οι σελίδες που θα ανακτήσουμε είναι ουσιαστικά B(X) = 50, αφού εγγραφές με διαδοχικές τιμές στο γνώρισμα ΗΛΙΚΙΑ θα είναι γραμμένες διαδοχικά και στον δίσκο.
Άρα **COST**(X) = 50.

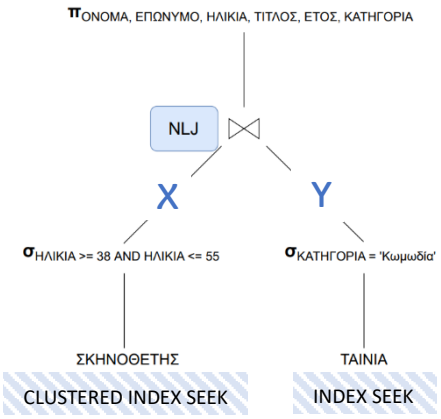
· *Για το COST(Y):*

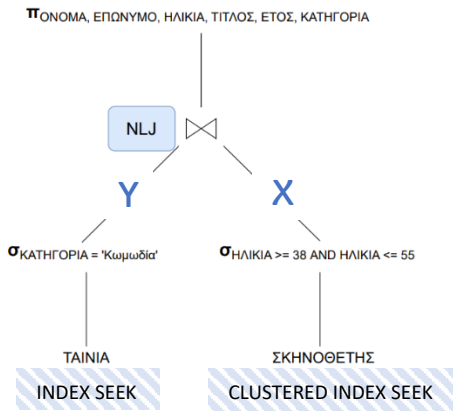
σΚΑΤΗΓΟΡΙΑ = 'Κωμωδία' (ΤΑΙΝΙΑ) = **Y**

Πληροφορούμαστε ότι υπάρχει απλό ευρετήριο (non-clustered index) B+ δέντρο στο γνώρισμα ΤΑΙΝΙΑ.ΚΑΤΗΓΟΡΙΑ, το οποίο βρίσκεται στη μνήμη. Το κόστος σε I/O για να διατρέξουμε το δέντρο μέχρι να καταλήξουμε στα φύλλα του είναι λοιπόν 0.

Αντίθετα με προηγουμένως, εδώ έχουμε μη συσταδοποιημένο ευρετήριο (non-clustered index), με αποτέλεσμα στη χειρότερη περίπτωση κάθε εγγραφή να βρίσκεται και σε διαφορετική σελίδα στο δίσκο. Συνεπώς οι σελίδες που θα ανακτήσουμε είναι MIN {T(Y), B(TAINIA) } = MIN {2000, 20000/20} = MIN{2000, 1000} = 1000
Άρα **COST**(Y) = 1000.

⇨ Τελικά **COST**(Result) = COST(X) + 2*B(X) + COST(Y) + 2*B(Y) = 50 + 2*50 + 1000 + 2*100 = 50 + 100 +1000 + 200 = **1350**





Όπως υπολογίσαμε προηγουμένως:

T(X) = 2000, **B(X)** = 50, **COST**(X) = 50

T(Y) = 2000, **B(Y)** = 100, **COST**(Y) = 1000

M = 16

Θα εξετάσουμε τόσο την πράξη X ⋈ Y όσο και την Y ⋈ X, εφόσον στον αλγόριθμο NLJ (BNLJ) η σειρά ενδέχεται να μεταβάλλει το κόστος.

1) X ⋈ Y

COST(Result) = COST(X) + $\left\lceil \frac{B(X)}{M-1} \right\rceil * \text{COST(Y)} = 50 + \left\lceil \frac{50}{15} \right\rceil * 1000 = 50 + 4*1000 =$ **4050** I/Os

2) Y ⋈ X

COST(Result) = COST(Y) + $\left\lceil \frac{B(Y)}{M-1} \right\rceil * \text{COST(X)} = 1000 + \left\lceil \frac{100}{15} \right\rceil * 50 = 1000 + 7*50 =$ **1350** I/Os

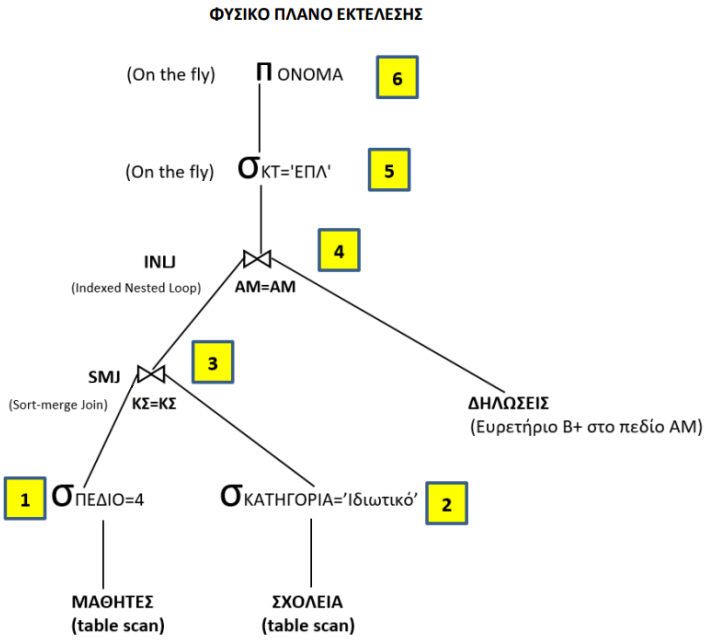
<u>Παρατηρήσεις:</u>

Μεταξύ των δύο NLJ που εξετάστηκαν (με αλλαγή στη σειρά), αποδείχθηκε καλύτερη η έκδοση όπου αριστερά του τελεστή τοποθετήθηκε το Y, το οποίο έχει μεγαλύτερο κόστος υπολογισμού. Αυτό είναι λογικό, εφόσον για όποια σχέση βρισκόταν στα δεξιά, το κόστος αυτό θα ήταν επαναλαμβανόμενο! Συνεπώς ήταν προτιμότερο η ποσότητα που θα επαναλαμβανόταν αρκετές φορές να ήταν η μικρότερη δυνατή.

Το βέλτιστο αποτέλεσμα επιτεύχθηκε είτε με χρήση του SMJ είτε του NLJ. Αυτό δεν είναι παράξενο, αφού τα ενδιάμεσα αποτελέσματα X,Y είχαν σχετικά μικρό κόστος δημιουργίας (για την ακρίβεια το ένα εκ των δύο είχε μικρό κόστος, και αυτό εκμεταλλευτήκαμε). Ο SMJ λόγω της γραμμικότητας του θα ανταποκρινόταν πιθανότατα πολύ καλύτερα από τον NLJ αν και τα δύο κόστη (**COST**(X), **COST**(Y)) ήταν σημαντικά μεγαλύτερα.

Άσκηση 4

(A)



Δίνονται τα εξής:

Τ(ΜΑΘΗΤΕΣ) = 1200 **Β**(ΜΑΘΗΤΕΣ) = 100 **V**(ΜΑΘΗΤΕΣ, ΠΕΔΙΟ) = 4

Τ(ΣΧΟΛΕΙΑ) = 600 **Β**(ΣΧΟΛΕΙΑ) = 120

Τ(ΔΗΛΩΣΕΙΣ) = 12000 **Β**(ΔΗΛΩΣΕΙΣ) = 400

Μ = 5

Έστω $\sigma_{ΠΕΔΙΟ = 4}$(ΜΑΘΗΤΕΣ) = **Χ** και $\sigma_{ΚΑΤΗΓΟΡΙΑ = 'Ιδιωτικό'}$ (ΣΧΟΛΕΙΑ) = **Υ**

1. $\sigma_{ΠΕΔΙΟ = 4}$
   Όπως δηλώνει το φυσικό πλάνο εκτέλεσης, γίνεται table scan (εφόσον δεν υπάρχει κάποιο ευρετήριο στο γνώρισμα ΜΑΘΗΤΕΣ.ΠΕΔΙΟ).

   $COST_{\sigma ΠΕΔΙΟ = 4}$ = **Β**(ΜΑΘΗΤΕΣ) = 100 I/Os

2. $\sigma_{ΚΑΤΗΓΟΡΙΑ = 'Ιδιωτικό'}$
   Όπως δηλώνει το φυσικό πλάνο εκτέλεσης, γίνεται table scan (εφόσον δεν υπάρχει κάποιο ευρετήριο στο γνώρισμα ΣΧΟΛΕΙΑ.ΚΑΤΗΓΟΡΙΑ).

   $COST_{\sigma ΚΑΤΗΓΟΡΙΑ = 'Ιδιωτικό'}$ = **Β**(ΣΧΟΛΕΙΑ) = 120 I/Os

3. SMJ ⋈
   Για να υπολογίσουμε το ζητούμενο κόστος πρέπει να εκτιμήσουμε αρχικά το μέγεθος των ενδιάμεσων αποτελεσμάτων που συμμετέχουν στη σύζευξη.

- Γνωρίζουμε ότι τα επιστημονικά πεδία είναι 4 και οι μαθητές κατανέμονται ομοιόμορφα στα 4 επιστημονικά πεδία. Συνεπώς:
  $\mathbf{T}(X) = \frac{T(ΜΑΘΗΤΕΣ)}{V(ΜΑΘΗΤΕΣ,ΠΕΔΙΟ)} = \frac{1200}{4} = 300$

  Κάθε σελίδα χωράει T(ΜΑΘΗΤΕΣ) / B(ΜΑΘΗΤΕΣ) = 1200/100 = 12 εγγραφές της σχέσης ΜΑΘΗΤΕΣ, άρα και 12 εγγραφές του X (εφόσον δεν γίνεται κάποια προβολή).
  Άρα $\mathbf{B}(X) = \frac{300}{12} = 25$

- Όσον αφορά τα σχολεία, γνωρίζουμε ότι το 5% του συνόλου των λυκείων είναι ιδιωτικά. Επομένως:
  $\mathbf{T}(Y) = \frac{5}{100} * T(ΣΧΟΛΕΙΑ) = \frac{5}{100} * 600 = 30$

  Κάθε σελίδα χωράει **T**(ΣΧΟΛΕΙΑ) / **B**(ΣΧΟΛΕΙΑ) = 600/120 = 5 εγγραφές της σχέσης ΣΧΟΛΕΙΑ, άρα και 5 εγγραφές του Y (εφόσον δεν γίνεται κάποια προβολή).
  Άρα $\mathbf{B}(Y) = \frac{30}{5} = 6$

Ο αλγόριθμος SMJ απαιτεί οι πίνακες που συμμετέχουν στη σύζευξη (εδώ οι X,Y) να είναι ταξινομημένοι. Εφόσον λοιπόν οι X,Y δεν είναι ήδη ταξινομημένοι, θα πρέπει να ταξινομηθούν. Θα εξετάσουμε εάν δεδομένων των μεγεθών (των πινάκων και της μνήμης) μπορεί να τρέξει η αποδοτική έκδοση του SMJ ή η λιγότερο αποδοτική.

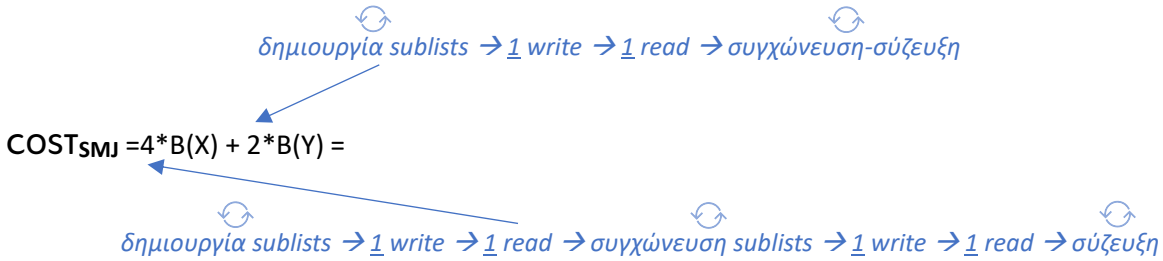
1) B(X) > M, B(Y) > M, επομένως η ταξινόμηση κανενός εκ των δύο πινάκων δεν μπορεί να γίνει απευθείας στη μνήμη (με γνωστούς αλγορίθμους ταξινόμησης όπως QuickSort, MergeSort κ.λπ.).

2) Ο X θα δημιουργήσει $\left\lceil \frac{B(X)}{M} \right\rceil = \left\lceil \frac{25}{5} \right\rceil = 5$ υπολίστες.
   Ο Y αντίστοιχα θα δημιουργήσει $\left\lceil \frac{B(Y)}{M} \right\rceil = \left\lceil \frac{6}{5} \right\rceil = 2$ υπολίστες.
   Συνολικά λοιπόν δημιουργούνται 5+2 = 7 > 5 υπολίστες. Άρα <u>δεν</u> μπορεί να τρέξει η αποδοτική έκδοση του SMJ.

3) Για τις υπολίστες των X,Y έχουμε MAX{5, 2} = 5 <= 5. Άρα μπορεί να τρέξει ο SMJ, αλλά όχι αποδοτικά. Στην περίπτωση αυτή, έχουμε:

   · *Για τον X:*
   Δημιουργούνται 5 υπολίστες οι οποίες **γράφονται**[(1)] στον δίσκο. Στη συνέχεια **ξαναδιαβάζονται**[(2)] στη μνήμη και συγχωνεύονται σε 1 (με μέγεθος 25 σελίδων). Αυτή **ξαναγράφεται**[(3)] στον δίσκο και θα **ξαναδιαβαστεί**[(4)] μια τελευταία φορά για τη σύζευξη.

   · *Για τον Y:*
   Δημιουργούνται 2 υπολίστες οι οποίες **γράφονται**[(1)] στον δίσκο. Στη συνέχεια **ξαναδιαβάζονται**[(2)] στη μνήμη για την συγχώνευση-σύζευξη με την 1 ταξινομημένη υπολίστα του X (απαιτούνται 1 + 2 = 3 σελίδες < M).



= 4*25 + 2*6 = 100 + 12 = 112

4. INLJ ⋈

Έστω ότι X ⋈ Y = Z

Για να υπολογίσουμε το ζητούμενο κόστος πρέπει να εκτιμήσουμε αρχικά το μέγεθος του Z (το μέγεθος της σχέσης ΔΗΛΩΣΕΙΣ είναι γνωστό).

Λαμβάνοντας υπόψιν ότι το 5% των σχολείων είναι ιδιωτικά και με την υπόθεση ότι οι T(ΜΑΘΗΤΕΣ) = 1200 μαθητές κατανέμονται ομοιόμορφα στα T(ΣΧΟΛΕΙΑ) = 600 σχολεία, το 5% των μαθητών ανήκει σε ιδιωτικό σχολείο. Υποθέτοντας ανεξαρτησία μεταξύ του πεδίου που επέλεξε ο μαθητής και του σχολείου (άρα και της κατηγορίας σχολείου) όπου ανήκει, οι μαθητές που ανήκουν σε ιδιωτικό σχολείο και κατά συνέπεια το πλήθος εγγραφών του X που γίνεται join με κάποια εγγραφή του Y είναι $\frac{5}{100} * 300 = 15.$

Συνεχίζοντας τον συλλογισμό, γνωρίζουμε ότι το γνώρισμα ΜΑΘΗΤΕΣ.ΚΣ (άρα και Χ.ΚΣ) είναι ξένο κλειδί το οποίο αναφέρεται (references) στο γνώρισμα ΣΧΟΛΕΙΑ.ΚΣ (άρα και Υ.ΚΣ). Επομένως συμπεραίνουμε ότι κατά τη σύζευξη (που παράγει το Z) κάθε ένας από τους 15 μαθητές που προσδιορίστηκαν παραπάνω γίνεται join με ένα ακριβώς σχολείο του Υ. Άρα το πλήθος των εγγραφών που προκύπτουν από τη σύζευξη είναι ακριβώς 15, δηλαδή T(Z) = 15.

Ως προς την εφαρμογή του αλγορίθμου INLJ τώρα, γνωρίζουμε ότι υπάρχει ένα απλό ευρετήριο (non-clustered index) B+δέντρο στο γνώρισμα ΔΗΛΩΣΕΙΣ.ΑΜ και όλες οι σελίδες του ευρετηρίου βρίσκονται στην κύρια μνήμη. Επίσης, κάθε μαθητής μπορεί να δηλώσει μέχρι και 10 τμήματα του επιστημονικού πεδίου που διαγωνίστηκε.

Σύμφωνα με τον INLJ, για κάθε εγγραφή του Z (T(Z) εγγραφές συνολικά) γίνεται αναζήτηση μέσω του ευρετηρίου για εγγραφές της σχέσης ΔΗΛΩΣΕΙΣ που ικανοποιούν τη συνθήκη Z.ΑΜ = ΔΗΛΩΣΕΙΣ.ΑΜ και από τα παραπάνω μπορούμε να συμπεράνουμε ότι κάθε αναζήτηση μέσω του ευρετηρίου επιστρέφει το πολύ 10 αποτελέσματα (worst case). Το κόστος για τη διάσχιση του ευρετηρίου είναι 0 (είναι ήδη στη μνήμη), αλλά για την ανάκτηση των αποτελεσμάτων από τον δίσκο απαιτούνται κάθε φορά 10 I/Os αφού στη χειρότερη περίπτωση οι εγγραφές αυτές θα βρίσκονται σε διαφορετικές σελίδες (το ευρετήριο είναι non-clustered).

$COST_{INLJ}$ = T(Z) * 10 = 15*10 = 150 I/Os

5. $\boldsymbol{\sigma}_{\text{ΚΤ = 'ΕΠΛ'}}$
   Δεν υπάρχει επιπλέον κόστος σε I/Os

6. $\boldsymbol{\pi}_{\text{ONOMA}}$
   Δεν υπάρχει επιπλέον κόστος σε I/Os

$$\textit{Τελικό Κόστος}:$$
$$\text{COST}_{\boldsymbol{\sigma}_{\text{ΠΕΔΙΟ=4}}} + \text{COST}_{\boldsymbol{\sigma}_{\text{ΚΑΤΗΓΟΡΙΑ= 'Ιδιωτικό'}}} + \text{COST}_{SMJ} + \text{COST}_{INLJ} =$$
$$= 100 + 120 + 112 + 150 = 482\ I/Os$$

(B)

- Έστω ότι το μέγεθος της διαθέσιμης μνήμης είναι M=32 (αντί για M=5).
  Αυτό ενδέχεται να επηρεάσει το κόστος του SMJ, καθώς είναι πιθανόν να μπορεί πλέον να εκτελεστεί κάποια πιο αποδοτική του έκδοση. Εξετάζουμε εκ νέου το SMJ:

  Υπενθύμιση: **T**(X) = 300 **B**(X) = 25
  T(Y) = 30 **B**(Y) = 6

  Εξετάζουμε και πάλι τις επιλογές του SMJ:
  1) B(X) < M, B(Y) < M, επομένως η ταξινόμηση και των δύο πινάκων μπορεί να γίνει απευθείας στη μνήμη (με γνωστούς αλγορίθμους ταξινόμησης όπως QuickSort, MergeSort κ.λπ.). Επίσης B(X) + B(Y) = 25 + 6 = 31 < M, που σημαίνει ότι οι δύο σχέσεις μπορούν να συνυπάρχουν στη μνήμη, να ταξινομηθούν όπως ειπώθηκε και στη συνέχεια να συγχωνευτούν, με το επιπλέον κόστος σε I/Os να είναι 0.

     Άρα **COST$_{\text{SMJ'}}$** = 0 I/Os

- Έστω, επίσης, ότι το ευρετήριο που υπάρχει στο πεδίο ΔΗΛΩΣΕΙΣ.ΑΜ είναι ευρετήριο συστάδων (clustered index) και όχι απλό ευρετήριο.
  Αυτό επηρεάζει το κόστος του INLJ. Το κόστος για τη διάσχιση του ευρετηρίου είναι και πάλι 0 (είναι ήδη στη μνήμη), αλλά για την ανάκτηση των αποτελεσμάτων από τον δίσκο απαιτείται πλέον κάθε φορά 1 I/O αφού οι εγγραφές αυτές θα βρίσκονται στην ίδια* σελίδα (το ευρετήριο είναι clustered).

  *κάθε σελίδα χωράει **T**(ΔΗΛΩΣΕΙΣ) / **B**(ΔΗΛΩΣΕΙΣ) = 12000/400 = 30 εγγραφές της σχέσης ΔΗΛΩΣΕΙΣ

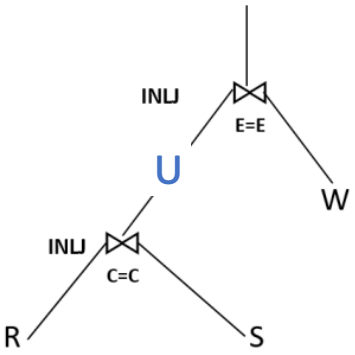
  Άρα **COST$_{\text{INLJ'}}$** = T(Z) * 1 = 15*1 = 15 I/Os

$$\textit{Τελικό Κόστος}:$$
$$\text{COST}_{\boldsymbol{\sigma}_{\text{ΠΕΔΙΟ=4}}} + \text{COST}_{\boldsymbol{\sigma}_{\text{ΚΑΤΗΓΟΡΙΑ= 'Ιδιωτικό'}}} + \text{COST}_{SMJ'} + \text{COST}_{INLJ'} =$$
$$= 100 + 120 + 0 + 15 = 235\ I/Os$$

Παρατηρήσεις:

Με τα νέα δεδομένα το τελικό κόστος μειώθηκε σημαντικά. Παρότι το μέγεθος της μνήμης δεν θα ήταν εύκολο να μεταβληθεί, η δημιουργία του clustered index (αντί non-clustered) είναι θέμα σχεδίασης και είναι και η αλλαγή που συνέβαλλε περισσότερο στη μείωση του κόστους.

Άσκηση 5



Δίνονται τα εξής:

T(R)=1000 **B**(R)=100 **V**(R.C)=125

T(S)=1500

T(W)=750

(A) Έστω ότι R ⋈ S = U

Θα υπολογίσουμε πρώτα το μέγεθος **T**(U) του ενδιάμεσου αποτελέσματος U.
Γνωρίζουμε ότι το γνώρισμα R.C είναι ξένο κλειδί το οποίο αναφέρεται (references) στο γνώρισμα S.C. Επομένως συμπεραίνουμε ότι κατά τη σύζευξη (που παράγει το U) κάθε εγγραφή του R γίνεται join με ακριβώς μια εγγραφή του S. Άρα το πλήθος των εγγραφών που προκύπτουν από τη σύζευξη είναι ακριβώς T(R), δηλαδή **T(U) = T(R) = 1000.**

Έστω ότι U ⋈ W = Result
Μένει να υπολογίσουμε το μέγεθος T(Result) του τελικού αποτελέσματος.
Γνωρίζουμε ότι το γνώρισμα S.E (άρα και το U.E) είναι ξένο κλειδί το οποίο αναφέρεται (references) στο γνώρισμα W.E. Επομένως συμπεραίνουμε ότι κατά τη σύζευξη (που παράγει το Result) κάθε εγγραφή του U γίνεται join με ακριβώς μια εγγραφή του W. Άρα το πλήθος των εγγραφών που προκύπτουν από τη σύζευξη είναι ακριβώς T(U), δηλαδή **T(Result) = T(U) = 1000.**

(B) 1ο (χρονικά) INLJ

Γνωρίζουμε ότι υπάρχει ένα απλό ευρετήριο (non-clustered index) στο γνώρισμα S.C. Επίσης, το γνώρισμα R.C είναι ξένο κλειδί το οποίο αναφέρεται στο S.C.

Σύμφωνα με τον INLJ, *για κάθε εγγραφή του R* (**T**(R) εγγραφές συνολικά) *γίνεται αναζήτηση μέσω του ευρετηρίου* για εγγραφές της σχέσης S που ικανοποιούν τη συνθήκη S.C = R.C.

Κάθε αναζήτηση μέσω του ευρετηρίου επιστρέφει το πολύ 1 αποτέλεσμα (το S.C είναι κλειδί) και τουλάχιστον 1 αποτέλεσμα (το R.C είναι ξένο κλειδί που αναφέρεται στο S.C). Δηλαδή τελικά *η αναζήτηση επιστρέφει ακριβώς 1 αποτέλεσμα*. Το κόστος για τη διάσχιση του ευρετηρίου είναι 0 (βρίσκεται ήδη στη μνήμη), αλλά για την ανάκτηση του αποτελέσματος από τον δίσκο απαιτείται κάθε φορά 1 I/O.

⇨ Έτσι, **COST**(U) = B(R) + *T(R)* * *1* = 100 + 1000 = 1100 I/Os

2ο (χρονικά) INLJ

Γνωρίζουμε ότι υπάρχει ένα απλό ευρετήριο (non-clustered index) στο γνώρισμα W.E. Επίσης, το γνώρισμα U.E είναι ξένο κλειδί το οποίο αναφέρεται στο W.E.

Για κάθε εγγραφή του U (**T**(U) εγγραφές συνολικά) γίνεται αναζήτηση μέσω του ευρετηρίου για εγγραφές της σχέσης W που ικανοποιούν τη συνθήκη W.E = U.E.

Κάθε αναζήτηση μέσω του ευρετηρίου επιστρέφει το πολύ 1 αποτέλεσμα (το W.E είναι κλειδί) και τουλάχιστον 1 αποτέλεσμα (το U.E είναι ξένο κλειδί που αναφέρεται στο W.E). Δηλαδή τελικά *η αναζήτηση επιστρέφει ακριβώς 1 αποτέλεσμα*. Το κόστος για τη διάσχιση του ευρετηρίου είναι 0 (είναι ήδη στη μνήμη), αλλά για την ανάκτηση του αποτελέσματος από τον δίσκο απαιτείται κάθε φορά 1 I/O.

⇨ Έτσι, **COST** = COST(U) + *T(U)* * *1* = 1100 + 1000 = 2100 I/Os

$\boxed{Τελικό\ Κόστος\ 2100\ I/Os}$

(C) Έστω ότι δημιουργούμε ένα ευρετήριο συστάδων (clustered) στο πεδίο R.C, εν δυνάμει χρήσιμο για τον υπολογισμό του πρώτου (χρονικά) INLJ. Με το υπάρχον πλάνο εκτέλεσης το ευρετήριο αυτό δεν θα μπορούσε να αξιοποιηθεί για αναζήτηση, αφού σύμφωνα με τον αλγόριθμο μπορούμε να επικαλεστούμε τυχόν ευρετήρια μόνο για την σχέση που εμφανίζεται δεξιά του τελεστή (την «εσωτερική»).

Ωστόσο, λόγω της ύπαρξης του ευρετηρίου συστάδων, καθορίζεται η φυσική διάταξη των εγγραφών της R στον δίσκο, ώστε εγγραφές με ίδια τιμή στο γνώρισμα C να βρίσκονται διαδοχικά γραμμένες στον δίσκο. Κατά συνέπεια, όταν διαβάζονται οι εγγραφές (μέσω της κλήσης του getNext() στο μοντέλο του επαναλήπτη) υπάρχει πλήρης εικόνα του πλήθους των εγγραφών που έχουν π.χ. C = 7. Για αυτές τις εγγραφές προηγουμένως γίνονταν πολλαπλές αναζητήσεις μέσω του ευρετηρίου στο S.C ενώ τώρα αρκεί μία αναζήτηση για κάθε batch εγγραφών της R που έχουν κοινή τιμή στο C.

Ο αλγόριθμος δηλαδή έχει ως εξής:

- Έστω prev η τιμή του γνωρίσματος C της εγγραφής της R που διαβάστηκε τελευταία.
- Έστω ότι η κλήση της getNext() επιστρέφει μια εγγραφή της R με C = x.
- Αν x = prev εκτελούμε νέο getNext().
- Αν x ≠ prev εκτελούμε seek στο ευρετήριο που υπάρχει στο S.C για την αναζήτηση της τιμής prev.
- Το 1 και μοναδικό αποτέλεσμα που επιστρέφει η αναζήτηση αυτή, γίνεται join με το batch εγγρφών της R που έχει διαβαστεί και έχει C = prev.

Για τον υπολογισμό του (μειωμένου) κόστους που προκύπτει από τη διαδικασία αυτή παρατηρούμε:

”~~*Για κάθε εγγραφή του R* (**T**(R) εγγραφές συνολικά)~~ *Για κάθε δέσμη εγγραφών του R γίνεται αναζήτηση μέσω του ευρετηρίου* για εγγραφές της σχέσης S που ικανοποιούν τη συνθήκη S.C = R.C. Λόγω του ότι **V**(R.C) = 125, θα έχουμε *125 τέτοιες δέσμες*. Με την υπόθεση ομοιόμορφης κατανομής, κάθε δέσμη θα έχει μέγεθος **T**(R) / **V**(R.C) = 1000 / 125 = 4 εγγραφές.

Κάθε αναζήτηση μέσω του ευρετηρίου επιστρέφει το πολύ 1 αποτέλεσμα (το S.C είναι κλειδί) και τουλάχιστον 1 αποτέλεσμα (το R.C είναι ξένο κλειδί που αναφέρεται στο S.C). Δηλαδή τελικά *η αναζήτηση επιστρέφει ακριβώς 1 αποτέλεσμα*. Το κόστος για τη διάσχιση του ευρετηρίου είναι 0 (βρίσκεται ήδη στη μνήμη), αλλά για την ανάκτηση του αποτελέσματος από τον δίσκο απαιτείται κάθε φορά 1 I/O.

⇨ Έτσι, **COST**(U) = B(R) + *125* * *1* = 100 + 125 =225 I/Os”

⇨ **COST** = COST(U) + *T(U)* * *1* = 225 + 1000 = 1225 I/Os

$\boxed{Τελικό\ Κόστος\ 1225\ I/Os}$