Μαρίνα Σαμπροβαλάκη
3180234

Άσκηση 1

$$\frac{1000000}{20000} = 50\ εγγραφές/σελίδα$$

Έχουμε ότι V(R,a) = n και υπάρχει ευρετήριο.

α) $T(\sigma(\alpha=2)(R)) = \frac{T(R)}{V(R,a)} = \frac{1000000}{n} = \mu$

1. i) Clustered Index:

Εφόσον έχουμε clustered index τότε οι σελίδες είναι συνεχόμενες. Άρα, θα έχουμε

$$\frac{\mu}{50} = x\ I/Os$$

όπου τα I/Os αναφέρονται στον αριθμό των blocks/pages που διαβάζω.

ii) Non-Clustered index:

Εφόσον έχουμε non-clustered index τότε οι σελίδες δεν είναι συνεχόμενες. Άρα, θα έχω

μ I/Os (στην χειρότερη περίπτωση),

όπου τα I/Os αναφέρονται στον αριθμό των blocks/pages που διαβάζω.

2. Αν σαρώσουμε όλο τον πίνακα θα έχουμε I/O 20000 όσες και οι σελίδες. Για να συμφέρει να χρησιμοποιήσουμε ευρετήριο, πρέπει το I/O να είναι μικρότερο από 20000. Θα βρούμε τις τιμές του n όπου ισχύει αυτό:

$$n > 1$$

Άρα, για αυτό το ερώτημα συμφέρει πάντα να χρησιμοποιήσουμε το ευρετήριο.

b) 1. T(σ(α=2)(R)) = $\frac{\mathrm{T(R)}}{V(\mathrm{R,a})} = \frac{1000000}{n/10} = \mu 2$

i) Clustered Index: Εφόσον έχουμε clustered index τότε οι σελίδες είναι συνεχόμενες. Άρα, θα έχω

$$\frac{\mu 2}{50} = x\ I/Os$$

όπου τα I/Os αναφέρονται στον αριθμό των blocks/pages που διαβάζω.

ii) Non-Clustered index: Εφόσον έχουμε non-clustered index τότε δεν είναι συνεχόμενες οι σελίδες. Άρα, θα έχω

μ2 I/Os (στην χειρότερη περίπτωση) ,

όπου το I/O αναφέρεται στον αριθμό των blocks/pages που χρειάζεται να διαβάσω.

2. Σαρώνοντας όλο τον πίνακα θα έχουμε 20000 I/Os όσες και οι σελίδες.

Για να συμφέρει να χρησιμοποιήσουμε ευρετήριο, πρέπει τα I/Os να είνσαι μικρότερα από 20000. Οπότε ισχύει:

$$\frac{1000000}{n} = I/O$$

$$n > 50$$

Συνεπώς, συμφέρει να χρησιμοποιήσουμε το ευρετήριο εφόσον υπάρχουν πάνω από 50 εγγραφές στο α.

<u>Άσκηση 2</u>

a) Υποθέτουμε ότι στην καλύτερη περίπτωση, σε κάθε εύρος τιμών όλες οι εγγραφές είναι σε μια τιμή, δηλαδή, για το εύρος τιμών [21,40] μπορούμε να έχουμε στην σχέση R 80 εγγραφών στο 21 και στην S 100 εγγραφών στην 21. Τότε, κάνοντας το join θα έχουμε

80 * 100 = 8000 tuples.

Με την ίδια λογική θα βγάλουμε τον τύπο:

0 + (80*100) + (100*60) + (20*60) + 0 = 15200 tuples

b) Κάνουμε την παραδοχή ότι υπάρχει ομοιόμορφη κατανομή, οπότε σε κάθε τιμή θα αντιστοιχεί περίπου ο ίδιος αριθμός εγγραφών. Για παράδειγμα, στο εύρος τιμών [21,40] και στην σχέση R που έχει 80 εγγραφές, σε κάθε τιμή θα αντιστοιχούν

$$\frac{80}{20} = 4\ εγγραφές$$

Αντίστοιχα στην S στο ίδιο εύρος σε κάθε τιμή θα αντιστοιχούν

$$\frac{100}{20} = 5\ εγγραφές$$

Για να βρούμε τα tuples που θα παραχθούν μετά το join αρκεί να πολλαπλασιάσουμε τις παραπάνω τιμές με το πλήθος των διαφορετικών τιμών στο εύρος. Συνεπώς ισχύει:

$$5 * 4 * 20 = 400 \text{ tuples.}$$

Οπότε ισχύει:

$$0 + (5*4*20) + (5*3*20) + (1*3*20) + 0 = 760 \text{ tuples}$$

Άσκηση 3

1. Για τον NLJ ισχύει:

   Για να βρούμε τις σελίδες θα διαιρέσουμε το σύνολο των εγγραφών με τις εγγραφές ανά σελίδες.

   Οπότε για την R ισχύει

   $$\frac{20000}{25} = 800\ σελίδες$$

   Για την S ισχύει

   $$\frac{45000}{30} = 1500\ σελίδες$$

   Θα χρησιμοποιήσουμε ως εξωτερική σχέση εκείνη με τις περισσότερες σελίδες. Συνεπώς στην άσκηση μας την R.

   Χρησιμοποιώντας τον τύπο των διαφανειών καταλήγουμε στο εξής:

   $$B(S) + ceil(B(S)/(M-1)) * B(R) =$$

   $$1500 + ceil\ (1500/40) * 800 = 31900 \text{ I/Os}$$

   Για τον SMJ ισχύει:
   Σύμφωνα με τον τύπο

   $$M > \sqrt{\max(B(R), B(S)}$$

   και αφού $41 > 38.7298$

   καταλήγουμε ότι το μέγεθος του ενταμιευτή είναι αρκετό για να χρησιμοποιήσουμε την αποδοτική έκδοση του αλγορίθμου. Συνεπώς το κόστος θα είναι

$$5*(B(S)+B(R)) = 11500 \text{ I/Os}$$

Για τον Hash Join ισχύει:
Σύμφωνα με τον παρακάτω τύπο, όντως ισχύει

$$M > \sqrt{\min(B(R), B(S)}$$

και αφού $41 > 28,28$

Τότε το μέγεθος του ενταμιευτή είναι αρκετό για να επιτρέψει να εφαρμόσουμε Hash Join

Οπότε το κόστος θα είναι

$$3*(B(S) + B(R)) = 6900 \text{ I/Os}$$

2. Το ελάχιστο κόστος του SMJ είναι το άθροισμα των σελίδων των σχέσεων. Δηλαδή

$$B(S) + B(R) =$$
$$1500 + 800 = 2300 \text{ I/Os}$$

Ο πρώτος τρόπος για να πετύχουμε αυτό το κόστος είναι αν ταξινομήσουμε και τις 2 σχέσεις. Ένας εναλλακτικός τρόπος είναι να παρατηρήσουμε ότι και οι 2 σχέσεις χωράνε στη μνήμη, συνεπώς το join μπορεί να εκτελεστεί σε ένα πέρασμα. Κάνοντας hash στην S που έχει το μεγαλύτερο μέγεθος και διαβάζοντας μία μία τις σελίδες της R στη μνήμη έχουμε το ίδιο κόστος με το επιθυμητό.

Άσκηση 4

1η λειτουργία: Γίνεται Index Scan καθώς υπάρχει non clustered index στο γνώρισμα Εκδότης. Στο non clustered index το κόστος είναι ίσο με τον αριθμό των εγγραφών που ικανοποιούν την συνθήκη. Εφόσον η σχέση ΒΙΒΛΙΑ έχει 50.000 εγγραφές και οι διαφορετικές τιμές του πεδίου Εκδότες είναι 500, κάνοντας την παραδοχή ότι η τιμή υπάρχει και η κατανομή είναι ομοιόμορφη, κάθε τιμή από τις 500 διαφορετικές θα εμφανίζεται 100 φορές. Συνεπώς το κόστος της λειτουργίας είναι 100 I/Os. Σε μία σελίδα περιέχονται 10 εγγραφές βιβλίων, άρα αυτό σημαίνει ότι χωράνε σε 10 σελίδες.

2η λειτουργία: Εφαρμόζεται Indexed Nested Loop Join, οπότε πρέπει να ψάξουμε για κάθε εγγραφή αν υπάρχει στο index. Αν υπάρχει, το I/O κόστος θα αυξάνεται κατά 1. Στην σχέση ΔΑΝΕΙΣΜΟΙ υπάρχει ένα clustered index και οι εγγραφές είναι 300.000. Τα βιβλία ωστόσο είναι 50.000. Διαιρώντας αυτές τις τιμές καταλήγουμε πως κάθε βιβλίο αντιστοιχεί σε 6 δανεισμούς. Το παραπάνω επιχείρημα έχει ως βάση το γεγονός ότι όλα τα δεδομένα ακολουθούν την κανονική κατανομή. Άρα αφού κάθε ΚΒ του αποτελέσματος από το στάδιο 1 έχει 6 κοινά ΚΒ στον πίνακα δανεισμός και έχουμε 100 εγγραφές προκύπτει ότι από το 2ο στάδιο θα επιστραφούν 600 εγγραφές. Στα clustering indexes, το κόστος είναι ίσο με τον αριθμό των σελίδων που

ικανοποιούν την συνθήκη. Άρα έχουμε κόστος ίσο με B(R) + T(R)*1 (επί 1 γιατί το index θα βρεί ένα KB μέσα στον πίνακα ΔΑΝΕΙΣΜΟΣ). Το κόστος B(R) είναι 100 και είναι αυτό που υπολογίσαμε στο πρώτο στάδιο και το T(R) είναι 100 διότι έχουμε 1 I/O για κάθε εγγραφή του σταδίου 1 που εντοπίζεται στον πίνακα ΔΑΝΕΙΣΜΟΣ. Άρα το συνολικό κόστος μέχρι τώρα είναι 200 I/O's..

3η λειτουργία: Στο προηγούμενο στάδιο επιστράφηκαν 600 εγγραφές. Από τον τύπο

B(R) + B(R)*B(S)

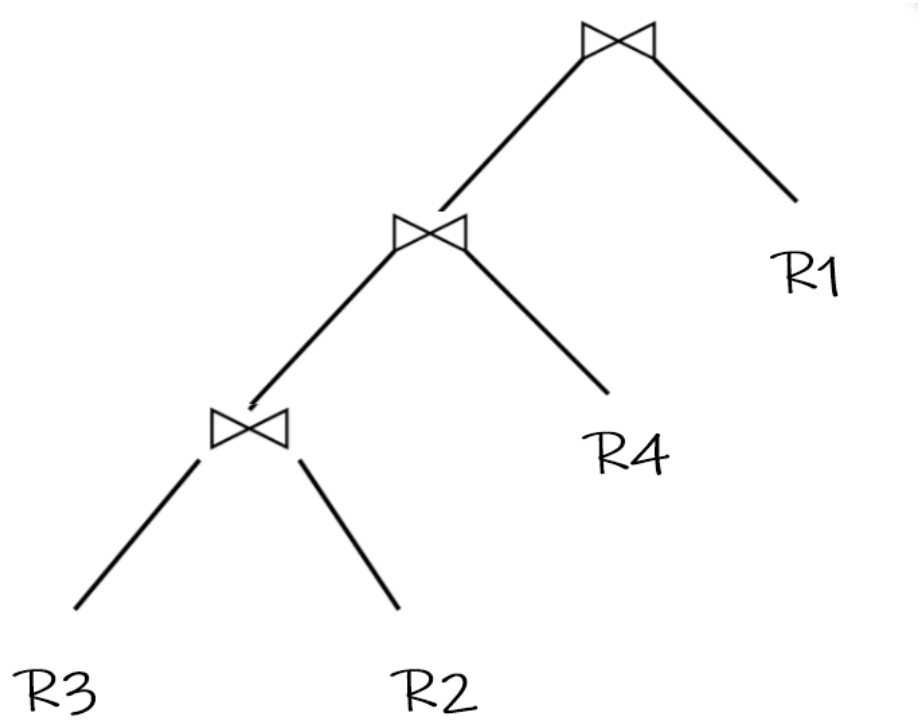
καταλήγουμε ότι τα I/Os είναι 2200.
Το κόστος είναι 0 καθώς οι προηγούμενες 2 λειτουργίες έφεραν στην κύρια μνήμη τις 10 + 4 = 14 σελίδες που αφορά το NLJ. Ένας ακόμα λόγος που το κόστος είναι μηδενικό είναι γιατί έχουμε για κάθε εγγραφή ένα ακριβώς αποτέλεσμα στον πίνακα ΔΑΝΕΙΖΟΜΕΝΟΙ (αφού το ΚΔ είναι PK) και έχουμε 600 εγγραφές. Άρα θα επιστραφούν και πάλι 600 εγγραφές.

4η λειτουργία: Το κόστος εδώ είναι μηδενικό καθώς οι σελίδες είναι ήδη στη μνήμη και γίνονται project μόνο συγκεκριμένες.

Συνεπώς, το συνολικό I/O κόστος είναι 2200 και το query επέστρεψε 233 εγγραφές.

Άσκηση 5

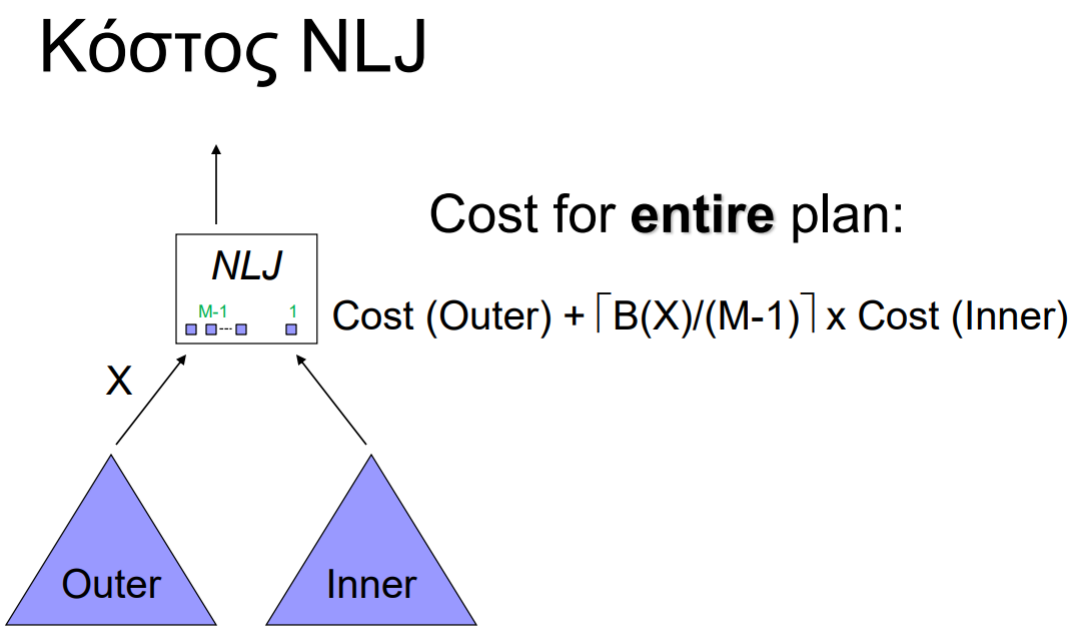
1. Το left deep λογικό πλάνο είναι το παρακάτω:



Μαρίνα Σαμπροβαλάκη
3180234

2. Στο προηγούμενο ερώτημα φτιάξαμε ένα left deep λογικό πλάνο. Αυτό θα αποτελέσει την ευρετική μας. Λόγω του περιορισμού που μας λέει η άσκηση, δεν μπορούμε να χρησιμοποιήσουμε Hash Join καθώς τότε θα χρειαζόμασταν αρκετό χώρο. Επίσης ο Sort Merge Join είναι blocking αλγόριθμος, δηλαδή πρέπει να περιμένει να τελειώσει η πρώτη φάση για να προχωρήσει στη δεύτερη, κάτι το οποίο σημαίνει ότι θα χρειαστεί κάποιο ενδιάμεσο αποτέλεσμα το οποίο δεν μπορούμε να αποθηκεύσουμε λόγω περιορισμών της μνήμης. Ωστόσο υπάρχει και ο NLJ ο οποίος χρησιμοποιεί pipelined execution και είναι αποδοτικότερος.

3. Κάνουμε την παραδοχή ότι όλοι οι πίνακες έχουν ίσες σελίδες. Εφόσον χρησιμοποιούμε NLJ, σύμφωνα με τις διαφάνειες του κ.Κωτίδη, ο τύπος για το κόστος του Plan X είναι



Ως outer επιλέγω την σχέση R1 γιατί έχει λιγότερες σελίδες από τις άλλες 3 αθροιστικά και ως εσωτερική τις 3 άλλες σχέσεις αθροιστικά.

$$Total\ Cost = B(R1) + \ ceiling(\frac{B(R1)}{M-1}) * [B(R2) + B(R3) + B(R4)]$$