Τζένη Μπολένα
3170117

Δεύτερη Σειρά Ασκήσεων

Άσκηση 1

Το **V(R,a) = n** δηλώνει ότι το **a** έχει **n** διακριτές τιμές. Άρα κάθε τιμή θα συναντάτε σε **1.000.000/n εγγραφές**.

1.000.000/20.000 -> 50 εγγραφές ανά σελίδα

1.

i.Για επερώτημα a:

Θα έχουμε 1.000.000/n εγγραφές /50 αφού χωράνε 50 εγγραφές ανα σελίδα το το index είναι clustered**.**

I/O: 1.000.000/n/50

Ομοίως για το επερώτημα b:

I/O: 10.000.000/n/50

ii. Στις περιπτώσεις που το ερετήριο είναι non-clustered αλλάζουν τα δεδομένα. Πλέον θα πρέπει να ακολουθείτε pointer για κάθε μια εγγραφή στα φύλλα.

Για επερώτημα a:

Θα γίνουν select 1.000.000/n rows άρα θα ακολουθήσουμε τόσους pointers.

I/O: 1.000.000/n

Για επερώτημα b:

Θα γίνουν select 1.000.000/(n/10)= 10.000.000/n rows άρα θα ακολουθήσουμε τόσους pointers.

I/O: 10.000.000/n

2.

Θα θεωρήσουμε ότι το ευρετήριο θα είναι αποδοτικό όταν θα διαβάζουμε λιγότερες σελίδες από ότι με table scan(δλδ. 20.000).

Για επερώτημα a:

Για να είναι πιο αποδοτικο το index θα πρέπει να ισχύει

1.000.000/n < 20.000 ⇔ **n > 50**

Για επερώτημα b:

Για να είναι πιο αποδοτικο το index θα πρέπει να ισχύει

10.000.000/n < 20.000 ⇔ **n > 500**

Άσκηση 2

a. Ας δώσουμε πρώτα το σκεπτικό με το οποίο το υπολογίζουμε. Ας πάρουμε για παράδειγμα την τέταρτη γραμμή.

[61..80]	20	60

Στον πίνακα S στο διάστημα [61...80] θα υπάρχουν 60/20 = 3 rows ανα τιμή. Συνολικά rows (για αυτά μόνο τα στοιχεία) από join θα είναι 20(όσες οι εγγραφές/rows του R στο διάστημα αυτό) * 3 = 60 rows απο join στο διάστημα [61...80]. Με το ίδιο σκεπτικό βρίσκουμε και τα υπόλοιπα.
Άρα το πλήθος των πλειάδων που θα εμφανιστουν συνολικά είναι:

0*(10/20) + 80*(100/20) + 100*(60/20) + 20*(60/20) + 30*(0/20) =
80*5 + 100*3 + 20*3 = **760 πλειάδες/rows**

b. Αν λάβουμε υπόψη την ομοιόμορφη κατανομή τότε έχουμε 100 διαφορετικές τιμές και σε κάθε μια θα αντιστοιχούν 230/100 = 2.3 rows. Άρα 230 rows του R θα γίνουν το καθένα join με 2.3 rows του S. 230*2.3 = **529 πλειάδες/rows**

Άσκηση 3

Η σχέση R χωράει σε 20.000/25 = 800 blocks
Η σχέση S χωράει σε 45.000/30 = 1500 blocks
M = 41

1.

- **NLJ:** θα επιλέξω για εξωτερική την R που είναι και η μικρότερη σχέση. Κρατάω 1 απο τις 41 σελίδες στην μνήμη για την R και τις άλλες 40 για την S. Οπότε προκύπτει ότι το κόστος σε I/O είναι:
  B(R) +[B(R)/40] * B(S) = 800 + [800/40] * 1500 = 800 + 30.000 = **30.800 I/O**
- **SMJ:** έχουμε μέγεθος μνήμης 41 blocks που δεν αρκεί για να τρέξουμε την αποδοτικη έκδοση του αλγορίθμου αφού 800/41 + 1500/41 = 20 + 37 = 57 > 41. Άρα θα τρέξουμε την μη αποδοτική. 5(B(R) + B(S)) = 5(800 + 1500) = **11.500 I/O**
- **Hash Join:** καμία από τις σχέσει δεν χωράει ολόκληρη στην μνήμη οπότε θα έχουμε read->write->read άρα 3(B(R) + B(S)) = 3(800 + 1500)= **6.900 I/O**

2.

Ένα γενικός τρόπος όπου ο SMJ είναι αποδοτικότερος είναι όταν κάνουμε το sorting στην μνήμη, με την υπόθεση ότι M>=sqrt(B(R) + B(S)). Τότε τα I/O είναι:

3(B(R) + B(S))

(στην περίπτωση μας θα ισχύει sqrt(800+1500) = sqrt(2300) = 48.Άρα πρέπει M >= 48)

Άλλη γενική λύση είναι να έχουμε Indexes και στις δύο σχέσεις. Έτσι θα αποφύγουμε το sort και απομένει μόνο το merge-join. Στην περίπτωση που τα index είναι clustered και μία από τις σχέσεις χωράει ολόκληρη στην μνήμη και μένει μια θέση μνήμης για την άλλη σχέση θα έχουμε **B(R) + B(S) I/O**

Άσκηση 4

1. **σΕκδόσης='Σαββάλας' -> θα φέρει 50.000/500 =**
   100 βιβλία/εγγραφές
2. **INLJ(Βιβλία.ΚΒ = Δανεισμοί.ΚΒ) -> 100*(300.000/50.000) = 100*6 = 600 εγγραφές**
3. **NLJ(temp.ΚΔ = Δανειζόμενοι.ΚΔ) -> 600*(10.000/10.000) = 600 εγγραφές**
4. **σ12<Ηλικία<20 -> (600/18)*7 = 33,3*7 = 34*7 = 238 εγγραφές**

I/O

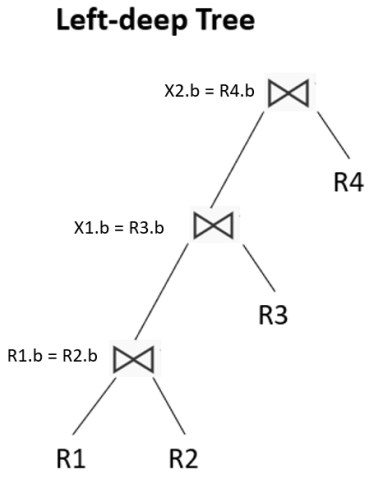
1. **100 I/O** αφού επιλέγει μόνο αυτά που έχουν τιμή Σαββάλας. Άρα θα ακολουθήσει μόνο 100 pointers στην μνήμη.
2. 300.000/15.000 = 20 εγγραφές ανα σελίδα(δανεισμοι).
   300.000/50.000 = 6 εγγραφές ανα βιβλιο
   100*upper_bound(6/20) = 100*upper_bound(0.3) = **100 I/O** συν τα 100 στο πρώτο βήμα, αρα **200 I/O**
3. Γενικός τύπος NLJ -> B(R) + (B(R)/(M-1))*B(S)
   Από πάνω έρχονται 600 στοιχεία/εγραφές με προβολή του ΚΔ από Δανεισμούς. Άρα θα κάνουμε την υπόθεση ότι κάθε εγγραφή της προβολής πιάνει χώρο όσο μια του δανεισμού. Με αυτην παραδοχή/υπόθεση το "B(R)" θα είναι 600/20(όπου 20 εγγραφές ανά σελίδα στους δανεισμους). Άρα θα είναι 30 σελίδες.
   Επομένως σε αυτήν την φάση θα διαβάσουμε upper_bound(30/19)*1000 = upper_bound(1.57)*1000 =
   2*1000 = **2000 I/O** στα οποία προσθέτουμε και τα 200 των προηγούμενων βημάτων άρα πάμε στα **2.200 I/O**

4. Σε αυτό το στάδιο γίνεται απλά ένα select που δεν απαιτεί I/O άρα **0 I/O**. Συνολικά σε όλο το query έχουν γίνει **2.200 I/O**

Άσκηση 5

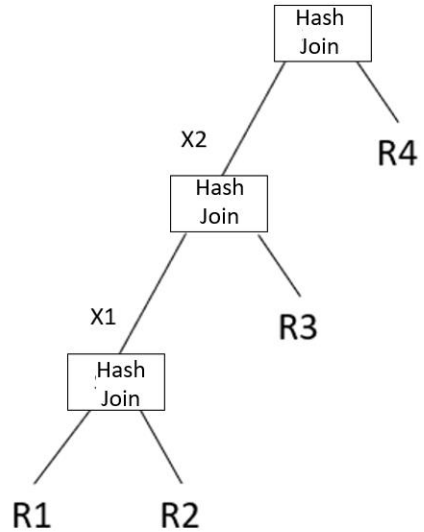
R1(a,b), R2(b,c), R3(b,e), R4(b,f)

1.



2.

Επειδή κανένα ενδιάμεσο αποτέλεσμα δεν χωράει στην μνήμη θα πάμε με το Hash Join.



3.

Ο κλασικός τύπος στον Hash Join είναι 3(B(R) +B(S)) :

Cost I/O: 3(B(R1) + B(R2) + B(R3) + B(R4))