2η σειρά ασκήσεων, Σχεδιασμός βάσεων Δεδομένων

p3170122 | Κων/νος Νικολούτσος

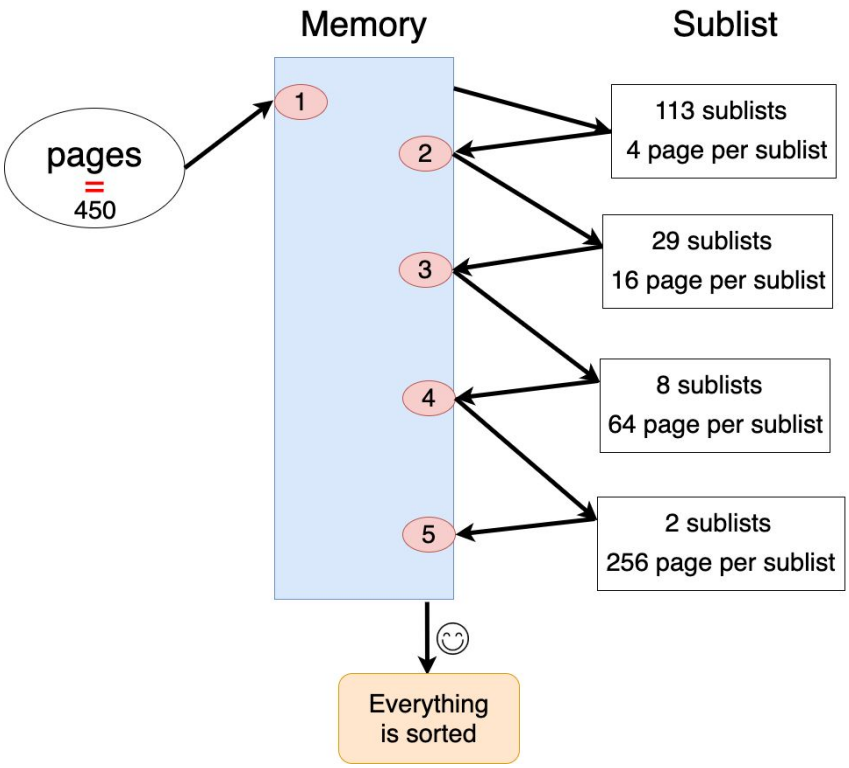
Άσκηση 1

1.
Ο αριθμός των *sorted sublist* μετά την πρώτη φάση είναι:
Pages = 45.000/100 = 450

Sorted_sublists = Pages/(buffering_size) = ceil(112,5) = 113

2.
*(Πέρασμα θεωρώ πόσες φορές φορτώνονται ολα τα δεδομένα στην RAM του DBMS)

Θα χρειαστούν **5 περάσματα**. (Δείτε την παρακάτω εικόνα για μια πιο visual απάντηση)



3.
Ειναι κατανοητό και απο την παραπάνω εικόνα οτι το κόστος είναι όσα και τα βελάκια μείον ένα:
Cost = 9*B(R) = 4050 I/Os

4.
Για να γίνουν μόνο δύο περάσματα αρκεί να ισχύει η παρακάτω συνθήκη (απο τα slides του μαθήματος).

M >= sqrt(B(R)) ⇔ M > 21.213 ⇔ **M >= 22**

Επιπλέον σε αυτην την περίπτωση που μεγάλωνε το M το κόστος θα ήταν 3*B(R)

Άσκηση 2

1.

Ο πιο αποδοτικός αλγόριθμος για το παράδειγμα είναι ο **Hash-Join** διότι έχει το μικρότερο I/O cost.

NLJ:
Το κόστος είναι: **1290 I/Os**

Επεξήγηση: Χρησιμοποιούμε ως εξωτερική σχέση εκεινη με τις περισσότερες σελίδες.Αρα ως εξωτερική σχέση έχουμε την R2
B(R2) + ceil(B(R2)/(M-1))*B(R1) = 1140 I/Os

SMJ:
Το κόστος είναι: **1200 I/Os**

Επεξήγηση: Το μέγεθος του ενταμιευτή μνήμης είναι μικρό για να εφαρμοστεί η αποδοτική μορφή του αλγορίθμου. Αυτο συμβαίνει διότι:

ceil((B(R1) + B(R2))/M) = 19 > M (Δεν χωράνε στην μνήμη)

Οπότε θα χρησιμοποιήσουμε κανονική έκδοση του αλγορίθμου και θα έχουμε κόστος:
5(B(R1) + B(R2)) = 1200 I/Os

Επίσης να τονίσουμε ότι το μεγεθος του ενταμιευτή μνήμης είναι αρκετό για την εφαρμογή της κανονικης έκδοσης του αλγορίθμου διοτι ισχύει:
M > sqrt(max(B(R1), B(R2))) ⇔ 13 > 12,24

HJ:
Το κόστος είναι: **720 I/Os**

Επεξήγηση: Το μέγεθος του ενταμιευτή μνήμης είναι μικρό για να εφαρμοστεί η βέλτιστη μορφή του αλγορίθμου(Καμία από τις δύο σχέσεις δεν χωράει ολοκληρη στην μνήμη). M < min(B(R1), B(R2))

Ωστόσο το μέγεθος του ενταμιευτή είναι αρκετό για να εφαρμόσουμε hash-join στην μνήμη. M > sqrt(min(B(R1), B(R2)))

Οπότε το κόστος θα είναι:
3(B(R1) + B(R2)) = 720 I/Os

2.
Αν αυξήσουμε το M σε 100 σελίδες.
Ακολουθώντας την ίδια λογική προκύπτει ότι:

NLJ:

#pages	**M = 13**	**M = 100**
#I/Os	1140	240

Επεξήγηση: Η έκδοση δεν αλλάζει αλλά κερδίζουμε από το γεγονός ότι φορτώνουμε περισσοτερα pages απο το R1. Το οποιο σημαινει οτι θα φορτώσουμε λιγότερες φορές σελίδες της σχέσης R2.

SMJ:

#pages	**M = 13**	**M = 100**
#I/Os	1200	720

Επεξήγηση: Χρησιμοποιούμε την καλύτερη έκδοση του SMJ διότι:
$M > sqrt(B(R1) + B(R2)) \Leftrightarrow 100 > 15.49$

HJ:

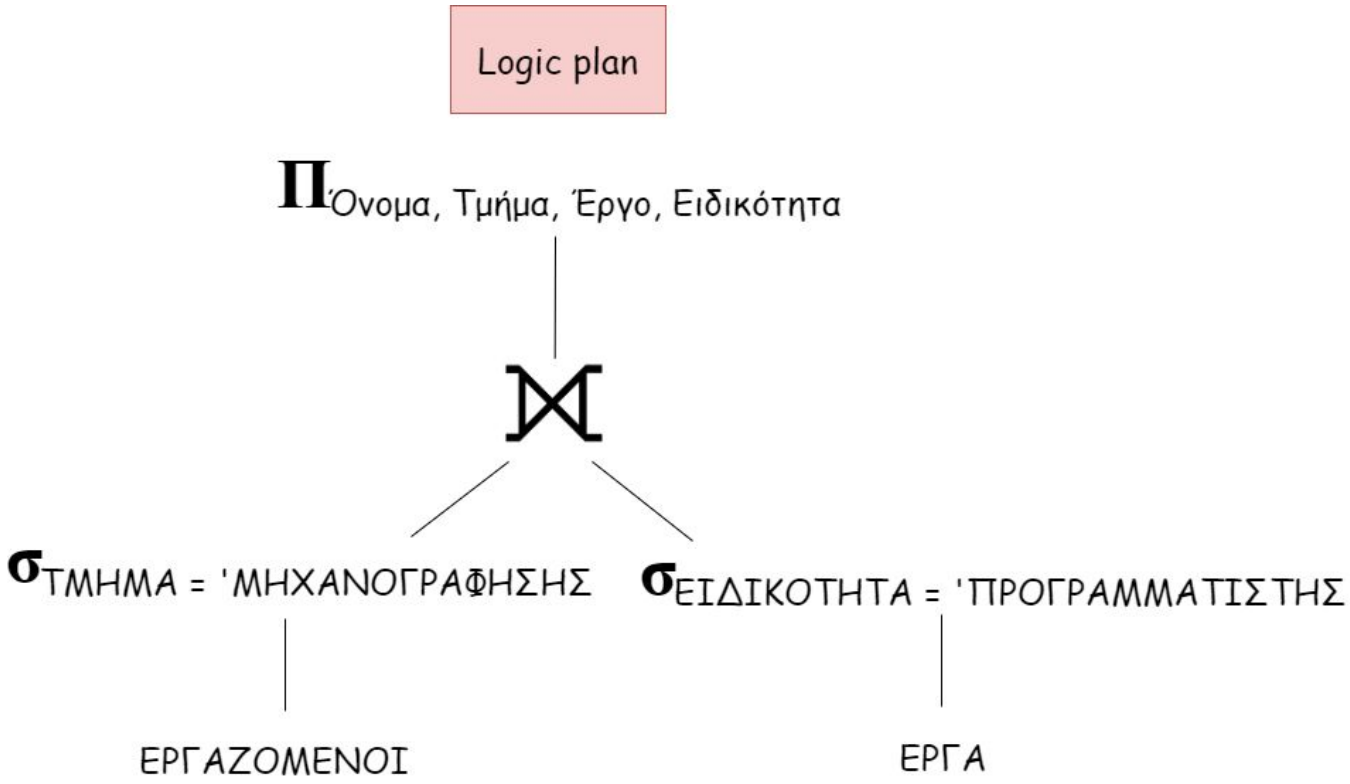
#pages	**M = 13**	**M = 100**
#I/Os	720	240

Επεξήγηση: Το μέγεθος του ενταμιευτή μνήμης είναι αρκετό για να χωρέσει η σχέση R2 άρα το κόστος ανέρχεται στο: B(R1) + B(R2)

Άσκηση 3

1.

Για την παραγωγή ενός καλού λογικού πλάνου έγινε χρήση της push down selection καλής πρακτικής



2.

α)

*Χρησιμοποιήσαμε οτι τα δεδομένα είναι ομοιόμορφα άρα η ειδικοτητα = ‘προγραμματιστής’ θα είναι 10 σελίδες.

*Χρησιμοποιήσαμε οτι τα δεδομένα είναι ομοιόμορφα άρα η τμημα = μηχανογραφησης θα είναι 25 σελίδες.

Ας υπολογίσουμε το κόστος ακολουθώντας bottom-up approach.

Πρεπει να γίνουν τα εξής βήματα:

- Θα κάνουμε two-phase-shorting και Θα κανουμε SMJ
- Γραφουμε τα αποτελέσματα

Τα κόστη των παραπάνω είναι:

- 3(B(Εργαζομενοι) + B(ΕΡΓΑ)) = 105 I/Os
- 0 (Μπορει να μην βγάζει κανενα αποτέλεσμα στην καλυτερη περίπτωση)

Αρα το ελάχιστο κόστος (lower bound) του παραπάνου φυσικού πλάνου ειναι **105 I/Os**

β) Στην άλλη περίπτωση το θα πρέπει να κάνουμε τα βήματα:
Θα εφαρμόσουμε την καλύτερη μορφή του αλγορίθμου αφήνοντας μονο μια θέση για την εσωτερική σχέση.

$$B(S)+\lceil B(S)/(M-1)\rceil*B(R)$$

Εφαρμόζοντας λοιπον το κόστος απο τα slides του μαθήματος προκύπτει ότι:
Το κόστος ειναι: **60 I/Os**

Παρατήρηση: Από την άλλη πλευρά όμως ο SMJ θα καταναλώνει λιγότερη ισχύ στο cpu.

Άσκηση 4

α)

1 κόστος:

Εφόσον γίνεται το table scan το DBMS θα διαβάσει ολα τα blocks. Αρα θα έχει κόστος **100 I/Os.** Όπου υποθέτοντας ομοιομορφία δεδομένων θα κρατήσει 100 εγγραφές δηλαδή θα βρίσκονται σε 5 pages(5*20 = 100).

2 κόστος:

Ομοίως, γίνεται το table scan και το DBMS θα διαβάσει ολα τα blocks.Αρα θα έχει κόστος **10 I/Os.** Υποθέτοντα πάλι ομοιομορφία δεδομένων το DBMS θα κρατήσει μόνο 9 εγγραφές, άρα 1 σελίδα.

3 κόστος:

Το κόστος εδω ειναι 0.
Τα προηγούμε operators φέραν στην κύρια μνήμη 5+1 = 6 σελίδες που χωράνε. Άρα το κόστος αυτου του τελεστή είναι 0 για το input.Αφου θα κάνει sort και join απευθείας **δίχως κάποιο I/O κόστος αφου αυτα χωρανε στην μνημη με την μια**. Απο αυτον τον operator στην καλύτερη περίπτωση δεν θα βγάλει κανένα αποτέλεσμα(0 εγγραφές) το join και στην χειρότερη θα γίνουν ολα με όλα (9 εγγραφες).
Με απλή αναλογία βγαίνει οτι οι σελίδες που θα αποθηκευτούν αυτες οι 9 εγγραφες δεν θα υπερβαινουν το μεγεθος της μνημης.

4 κόστος:
Εφόσον λοιπον τωρα εφαρμόζουμε **indexed** nested loop join για καθε εγγραφή του προηγούμενο 3ού κόστους θα ψαχνουμε στο index αν υπάρχει. Αν ναι τοτε το I/O cost θα αυξανεται κατα 1 μονάδα. Υποθέτοντας τώρα οτι απο τις 9 εγγραφες οι μίσες κατα μέσο ορο θα υπάρχουν. Έχουμε κόστος 4.5 δηλαδή **5 I/Os.**
Να τονίσουμε εδω μάλιστα ότι οι σελίδες του 3ου κόστους υπάρχουν ηδη στην μνήμη.

5 κόστος:
Το κόστος εδω σε Input είναι μηδενικό αφου εφόσον χωράνε ολα τα blocks στην μνήμη αρκει ο επεξεργαστής να κανει show μονο συγκεκριμένα.
Φυσικά τώρα αρκεί να κανει output τα δεδομένα το οποιο ισούται με τον αριθμό των σελίδων που έμειναν μετα απο ολα αυτα τα φιλτραρισμα στην μνήμη.

6 κόστος:
Μηδενικό κόστος οπως στο 5ο κόστος.

Αρα το συνολικό κόστος θα είναι: **115 I/Os**

Αξίζει να σημειώσουμε εδώ μάλιστα ότι τωρα λόγω ομοιομορφίας οι σελίδες μπορούσαν να φορτώνονται ολες στην κύρια μνήμη. Σε κανονικές συνθήκες ομως αυτο ειναι πιθανο να μην συμβαίνει και εκει θα έχουμε μεγαλύτερο I/O κόστος ωστε να έχουμε το αποτέλεσμα που θέλουμε!