# ΔΕΥΤΕΡΗ ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

ΚΑΡΑΓΕΩΡΓΗΣ ΠΛΑΤΩΝ 3180068

## 1Η Άσκηση

1) Η R αποθηκεύει 1,000,000/20,000 = 50 εγγραφές ανά σελίδα. Έχω:

$T(\sigma_{\alpha=2}(R))=T(R)/V(R,B)=1{,}000{,}000/n$

$T(\sigma_{K<=a<=L}(R))=T(R)/V(R,B)= 1{,}000{,}000 * 10/n$

i) Αν το ευρετήριο είναι clustered τότε για το πρώτο query έχω

$1{,}000{,}000/n * 50 = 200{,}000/n$ I/O's

Για το δεύτερο query έχω

$1{,}000{,}000 * 10 / n * 50 = 2{,}000{,}000/n$ I/O's

ii) Παίρνω τη χειρότερη περίπτωση όπου κάθε τιμή είναι σε διαφορετικό μπλοκ. Τότε το κόστος για το πρώτο query είναι:

$1{,}000{,}000/n$ I/O's

Για το δεύτερο:

$10{,}000{,}000/n$ I/O's

2) Το κόστος σαρώματος όλου του πίνακα είναι 20,000 αφού τόσες σελίδες χρησιμοποιεί συνολικά η σχέση R. Για να συμφέρει να χρησιμοποιήσω ευρετήριο, πρέπει το I/O να είναι μικρότερο από 20000. Το ευρετήριο είναι non-clustered επομένως χρησιμοποιώ τις τιμές που βρήκα στο 1ii.

Για το πρώτο query έχω: $1{,}000{,}000/n < 20{,}000 \Leftrightarrow n>50$

Για το δεύτερο query έχω $10{,}000{,}000/n <20{,}000 \Leftrightarrow n>500$

Άρα όσο το n είναι μεγαλύτερο αυτών των τιμών τότε συμφέρει να χρησιμοποιήσω ευρετήριο.

## 2Η Άσκηση

Α) Αρχικά στο πρώτο πεδίο του πίνακα δεν θα επιστραφούν tuples διότι η R δεν έχει καμία τιμή σε αυτό το διάστημα. Στο δεύτερο πεδίο μεταξύ 21-40 η R έχει 80 τιμές και η S έχει 100 άρα στην καλύτερη περίπτωση σε μία τιμή πχ 30 αν οι R,S έχουν 80 κοινές τιμές τότε θα επιστραφούν 80*100 = 8000 πλειάδες. Αντίστοιχα στο τρίτο μεταξύ 41-60 η R έχει 100 τιμές και η S έχει 60 άρα στην καλύτερη περίπτωση σε μία τιμή πχ 50 αν οι R,S έχουν 60 κοινές τιμές

τότε θα επιστραφούν 60*100=6000 πλειάδες. Στη συνέχεια στο τέταρτο μεταξύ 61-80 η R έχει 20 τιμές και η S έχει 60 άρα στην καλύτερη περίπτωση σε μία τιμή πχ 70 αν οι R,S έχουν 20 κοινές τιμές τότε θα επιστραφούν 20*60=1200 πλειάδες. Τέλος στο πέμπτο πεδίο επειδή η S έχει 0 τιμές δεν θα επιστραφεί καμία πλειάδα. Άρα συνολικά με βάση το ιστόγραμμα υπολογίζουμε ότι στην καλύτερη περίπτωση θα επιστραφούν 15,200 πλειάδες.

Β) Υποθέτουμε ότι οι τιμές είναι καταχωρημένες στα διαστήματα ομοιόμορφα. Στο πρώτο διάστημα όπως και πριν δεν έχει νόημα να κάνουμε υπολογισμούς γιατί η η R δεν έχει καμία τιμή. Στο δεύτερο λοιπόν, έχουμε ένα διάστημα 20 τιμών (από 21-40) και υποθέτουμε ότι κάθε τιμή από αυτές τις 20 έχει ίδιο αριθμό πλειάδων. Άρα για την R έχουμε 80/20 = 4 πλειάδες σε κάθε τιμή και αντίστοιχα για την S έχουμε 100/20 = 5 πλειάδες σε κάθε τιμή. Επομένως έχουμε 4*5=20 πλειάδες σε κάθε τιμή και συνολικά το διάστημα έχει 20 τιμές άρα 20*20 = 400 πλειάδες θα επιστραφούν από το διάστημα 21-40. Αντίστοιχα στο τρίτο διάστημα με την ίδια λογική έχουμε 100/20 = 5 για την R και 60/20 = 3 για την S επομένως έχουμε 3*5*20=300 πλειάδες. Ακολούθως για το τέταρτο διάστημα έχουμε 20/20 = 1 και 60/20 = 3 άρα 3*1*20=60 πλειάδες. Τέλος για το πέμπτο επειδή η S έχει 0 τιμές δεν θα επιστραφεί καμία πλειάδα. Συνολικά θα επιστραφούν λοιπόν 760 τιμές εάν η κατανομή των τιμών είναι ομοιόμορφη.

## 3$^{Η}$ Άσκηση

### 1$^{ο}$ Ερώτημα

Α) Για τον αλγόριθμο NLJ:

Πρέπει να υπολογίσουμε τις σελίδες για την S και για την R αντίστοιχα. Για την S έχουμε 45,000/30 = 1500 σελίδες και για την R έχουμε 20,000/25 = 800 σελίδες. Για τον αλγόριθμο NLJ πρέπει να εκτελέσουμε το διπλό for loop για να βρούμε τα συνολικά I/O's. Με βάση αυτά που είπαμε στο μάθημα και με βάση τις διαφάνειες είναι καλύτερο να έχουμε στο εσωτερικό loop τη σχέση με τις περισσότερες σελίδες και άρα θα βάλουμε την S. Εφαρμόζουμε λοιπόν τον τύπο B(R)+⌈B(R)/(M-1)⌉*B(S) από τις διαφάνειες. 800 + (800/40)*1500 και άρα 30,800 I/O's.

Β) Για τον αλγόριθμο SMJ:

Επιλέγουμε να χρησιμοποιήσουμε την παλιά έκδοση του SMJ.

Πρώτα πρέπει να υπολογίσουμε εάν μπορούμε να την χρησιμοποιήσουμε. Για να μπορέσουμε πρέπει να ισχύει ο τύπος $M \geq \sqrt{(\max(B(R), B(S)))}$.

Αν κάνουμε αντικατάσταση με τις τιμές της άσκησης προκύπτει 41>38,7 επομένως μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο γιατί ο τύπος επαληθεύτηκε. Άρα έχουμε 5*($B(R)$+$B(S)$)=11500 I/O's.

Γ) Για τον αλγόριθμο Hash Join:

Για να εφαρμοστεί ο αλγόριθμος Hash Join πρέπει να ισχύει ο τύπος √min(B(R),B(S)) και στη δική μας περίπτωση με αντικατάσταση έχουμε 41>28,8 επομένως μπορούμε να εφαρμόσουμε τον αλγόριθμο.

Έχουμε τον τύπο 3*((B(R)+B(S)) ο οποίος με αντικατάσταση μας δίνει 6900 I/O's.

## 2ο Ερώτημα

Για να πετύχουμε το ελάχιστο κόστος στον αλγόριθμο SMJ πρέπει να έχουμε τη βέλτιστη περίπτωση η οποία είναι να είναι οι πίνακες ταξινομημένοι. Τότε το κόστος είναι B(R)+B(S)=2300. Αυτή είναι και η πρώτη περίπτωση με την οποία πετυχαίνουμε το ελάχιστο δυνατό κόστος.

Η δεύτερη περίπτωση να πετύχουμε το ελάχιστο κόστος, είναι αν αυξήσουμε τη μνήμη του buffer ώστε να χωρέσουν όλα τα δεδομένα μέχρι και τη μεγαλύτερη σχέση (δηλαδή M>= max(B(R),B(S))) και να μην χρειαστεί να τα ξαναφορτώσουμε, πραγματοποιώντας τη διαδικασία σε ένα πέρασμα. Τότε και πάλι θα έχω B(R)+B(S)=2300.

## 4Η Άσκηση

Για να βρούμε το total cost πρέπει να βρούμε το κόστος από το καθένα από τα 4 στάδια που φαίνονται στο δοσμένο σχήμα. Παρακάτω θα τα υπολογίσουμε το καθένα ξεχωριστά.

### 1ο ΣΤΑΔΙΟ

Έχουμε τον περιορισμό ότι ο εκδότης πρέπει να είναι ο «Σαββάλας». Υπάρχει και ένα non-clustered ευρετήριο σε αυτό το column το οποίο επηρεάζει το κόστος I/O. Αρχικά είμαστε στον πίνακα ΒΙΒΛΙΑ όπου ισχύει T(ΒΙΒΛΙΑ) = 50,000 εγγραφές και B(ΒΙΒΛΙΑ)= 5,000 σελίδες. Επίσης έχουμε V(ΒΙΒΛΙΑ,Εκδότης) = 500. Επομένως, αφού έχουμε και non-clustered έχουμε 50,000/500 = 100 I/O κόστος στη χειρότερη περίπτωση. Όσο για τις εγγραφές που επιστρέφονται, αφού τα δεδομένα είναι ομοιόμορφα κατανεμημένα και έχουμε 50,000 βιβλία και 500 διαφορετικούς εκδότες τότε ο κάθε εκδότης έχει 100 βιβλία και επομένως και ο «Σαββάλας». Άρα επιστρέφονται 100 εγγραφές.

### 2ο ΣΤΑΔΙΟ

Τώρα πρέπει να υπολογίσουμε το κόστος του Indexed Nested Loop Join. Από το προηγούμενο στάδιο έχουμε 100 εγγραφές. Η λογική που θα ακολουθήσουμε είναι η εξής. Αρχικά στο εξωτερικό loop θα έχουμε τον πίνακα ΒΙΒΛΙΑ, διότι ο πίνακας ΔΑΝΕΙΣΜΟΣ και συγκεκριμένα το column ΚΒ του πίνακα αυτού έχει clustered index το οποίο μας οδηγεί και στο INLJ. Έτσι, για κάθε μία από τις 100 εγγραφές που έχουμε συλλέξει από το προηγούμενο στάδιο θα ψάχνουμε αν και πόσες φορές υπάρχει στον πίνακα ΔΑΝΕΙΣΜΟΣ. Το θέμα είναι ότι εδώ θεωρούμε ότι είναι εντελώς ομοιόμορφα κατανεμημένα τα δεδομένα και επομένως υποθέτουμε ότι από τους 300,000 δανεισμούς έχουν δανειστεί όλα τα βιβλία από 6 φορές

(αφού είναι 300,000/50,000 = 6). 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. Για τον υπολογισμό των I/O's υποθέτουμε ότι το clustered index είναι sparsed, καθώς δεν έχουμε κάποια ένδειξη από την άσκηση που να μας οδηγεί σε dense clustered index. Άρα έχουμε κόστος ίσο με B(R) + T(R)*1 (επί 1 γιατί το index θα βρεί ένα ΚΒ 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.

3ο ΣΤΑΔΙΟ

Θα κάνουμε Block Nested Loop join. Στο προηγούμενο στάδιο επιστράφηκαν 600 εγγραφές. Ο τύπος του BNLJ για τα I/O's είναι B(R) + B(R)*B(S). Το B(R) ουσιαστικά είναι το σύνολο των σελίδων που έχουν φορτωθεί ως εδώ και το B(S) είναι 10,000/1,000 σύμφωνα με τα δεδομένα της εκφώνησης. Άρα το συνολικό αποτέλεσμα είναι 200 + 200*10 = 2200 I/O's. Επιπλέον, έχουμε για κάθε εγγραφή ένα ακριβώς αποτέλεσμα στον πίνακα ΔΑΝΕΙΖΟΜΕΝΟΙ (αφού το ΚΔ είναι primary key) και έχουμε 600 εγγραφές. Άρα θα επιστραφούν και πάλι 600 εγγραφές.

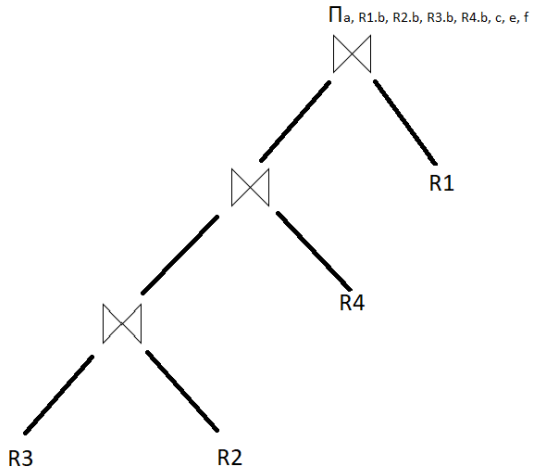
4ο ΣΤΑΔΙΟ

Έχουμε ως τώρα 600 εγγραφές. Αφού ακολουθείται η κανονική κατανομή υποθέτουμε ότι ο πίνακας των δανειζόμενων έχει σε κάθε ηλικία ίδια αριθμό δανειζόμενων. Στο διάστημα 7-24 υπάρχουν 18 ηλικίες άρα από κάθε ηλικία υπάρχουν κατά προσέγγιση 10,000/18 δανειζόμενοι. Προηγουμένως κάναμε join μεταξύ των εγγραφών του σταδίου 2 και του πίνακα ΔΑΝΕΙΖΟΜΕΝΟΙ επομένως θεωρούμε ότι ισχύει η ίδια αναλογία και στις 600 εγγραφές που έχουμε τώρα. Άρα η κάθε ηλικία έχει 600/18 εγγραφές. Από τις 18 ηλικίες θέλουμε να κρατήσουμε τις 7 (13, 14, 15, 16, 17, 18 ,19) και επομένως θεωρούμε ότι θα επιστραφούν προσεγγιστικά 7*600/18 = 233 εγγραφές στο στάδιο 4. Άρα και συνολικά θα επιστραφούν από το query 233 εγγραφές. Το κόστος του σταδίου 4 είναι μηδενικό γιατί απλά κάνουμε SELECT τις τιμές που είχαμε από πριν.

Εν τέλει το συνολικό I/O κόστος είναι 2200 και το query επέστρεψε 233 εγγραφές.

## 5Η Άσκηση

Α)




Β) Ο αλγόριθμος που θα επέλεγα είναι ο Nested Loop Join σε συνδυασμό με index scan. Η επιλογή που έκανα βασίζεται κυρίως στον περιορισμό που θέτει η άσκηση όσον αφορά τη μνήμη. Ο Hash Join δημιουργεί ένα Hash Function στη RAM, το οποίο χρειάζεται αρκετό χώρο και η εκφώνηση διευκρινίζει ότι δεν χωράει κάποια άλλη σχέση στη μνήμη. Επομένως οδηγούμαστε στη λογική να διαλέξουμε έναν αλγόριθμο, που είναι οικονομικότερος όσον αφορά το κόστος σε μνήμη. Παράλληλα, ο Sort Merge Join είναι blocking, 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 χωράει όμως στη μνήμη σε αυτή την άσκηση. Αυτό βέβαια δεν ισχύει και για τον NLJ όπου λειτουργεί με pipelined execution. Επίσης, ο SMJ (όπως και ο Hash Join) λειτουργεί με full table scan ενώ ο NLJ μπορεί να λειτουργήσει και με index scan το οποίο εκμεταλλεύεται πλήρως τα clustering indices. Έτσι, επιλέγω όπως προανέφερα τον NLJ σε συνδυασμό με index scan το οποίο είναι αποδοτικότερο με βάση τα δεδομένα της άσκησης.

Γ) Θέλω να βρω το I/O κόστος του αλγόριθμου INLJ όπου το index είναι clustering index. Σύμφωνα με τις διαφάνειες ο τύπος για ένα clustered indexed loop join 2 πινάκων R,S είναι B(R)+T(R) *⌈X/(αριθμός εγγραφών ανά σελίδα της S)⌉ (παίρνω τη χειρότερη περίπτωση και το θεωρώ sparse) και στην περίπτωση παραπάνω το εκτελούμε 3 φορές για τους 4 πίνακες. Η εκφώνηση ζητά να κρατήσουμε μόνο το κόστος των σελίδων της εκάστοτε σχέσης επομένως μία εκτίμηση του συνολικού κόστους είναι B(R2 ή R3) + B(R4) + B(R1).