ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

$2^{η}$ σειρά ασκήσεων

Αποστολόπουλος Θεμιστοκλής p318013

Ασκηση 1

T(R)=1.000.000 B(R)=20.000 V(R,a) = n

Έστω Χ το query Σα=2(R), Υ το Σκ<=a<=L

1a)Για Χ: έχω n διακριτές τιμές στο γνώρισμα a του R. Η πιθανότητα να πάρω ένα τυχαίο a από αυτές τις τιμές αν υποθέσω ομοιόμορφη κατανομή είναι 1/n. Η σχέση r έχει συνολίκα 1.000.000 εγγραφές.

Οπότε έχω T(X) = 1.000.000/n. Για την R έχω block size = 1.000.000 / 20.000 = 50. Επομένως B(X) = (1.000.000/n)/50 = 1.000.000/50n = 20.000/n

Για την περίπτωση που έχω clustered index το κόστος θα είναι B(X)(εγγραφές ταξινομημένες και οι κοντινές θα βρίσκονται στην ίδια σελίδα). Αν δεν είναι clustered, το κόστος = T(X) (στην χειρότερη περίπτωση θα χρειαστεί να περάσω από όλες τις εγγραφές).

Για Υ: έχω n/10 διακριτές τιμές στο γνώρισμα a του R που ανήκουν στο [K,L]. Η πιθανότητα να πάρω ένα τυχαίο a από αυτές τις τιμές αν υποθέσω ομοιόμορφη κατανομή είναι 1/(n/10). Η σχέση r έχει συνολίκα 1.000.000 εγγραφές.

Οπότε έχω T(Y) = 1.000.000/(n/10) = 10.000.000/n. . Για την R έχω block size = 1.000.000 / 20.000 = 50. Επομένως B(Y) = (10.000.000/n)/50 = 10.000.000/50n = 200.000/n

Για την περίπτωση που έχω clustered index το κόστος θα είναι B(X)(εγγραφές ταξινομημένες και οι κοντινές θα βρίσκονται στην ίδια σελίδα). Αν δεν είναι clustered, το κόστος = T(X) (στην χειρότερη περίπτωση θα χρειαστεί να περάσω από όλες τις εγγραφές).

b)Μας συμφέρει να χρησιμοποιήσουμε ευρετήριο όταν ο συνολικός αριθμός που θα διαβάσουμε με το index < αριθμό σελιδών με table scan.

Για X:

B(X)<B(R) => (1.000.000/N)<20.000 =>N >50

Για Y:

B(Y)<B(R) => (10.000.000/N)<20.000 =>N >500

2)

a)Για να βρώ τις συνολικές πλειάδες, πρέπει να βρώ τις πλειάδες που «επιστρέφονται» από το ερώτημα ΑΠΟ ΚΆΘΕ ΔΙΑΣΤΗΜΑ.οπότε θα βρώ τα T() μετά από το join κάθε γραμμής(που δείχνει διάστημα).Κάθε διάστημα έχει 20 διακριτές τιμές άρα V(R,b) = V(S,b)=20.

Με τον συμβολισμό R.bi , S.bi δείχνω στην εγγραφή R, S στο I διάστημα το όρισμα b για I =1,2,3,4,5.

Οπότε τα tuples που ψάχνω = T(R.b1)*T(S.b1)/V(R.B) + T(R.b2)*T(S.b2)/V(R.B) + T(R.b3)*T(S.b3)/V(R.B) +

T(R.b4)*T(S.b4)/V(R.B) + T(R.b4)*T(S.b4)/V(R.B) = 0 + 80*100/20 + 100*60/20 + 20*60/20 + 0 = 400 +300 + 60 = 760

Β) Αν πάμε τώρα με την υπόθεση ότι οι τιμές κατανέμονται ομοιόμορφα, θα δουλέψω με τους εξής τρόπους

Α) έχω συνολικά 230 tuples και στο R.b και στο S.b ανεξαρτήτως διαστήματος.Οι συνολικές διακριτές τιμές του είναι 100(από το 1 εώς στο 100). Οπότε μένει απλά να υπολογίσω το join το οποίο ισούται με 230*230/100 = 529.

Β τρόπος.Θέλω ομοιόμορφη κατανομή στα “buckets” διαστημάτων άρα πιθανότητα 20% κάποια τιμή να ανήκει σε ένα από τα 5 διαστήματα.επομένως κάθε διάστημα θα έχει 230*20% = 43 tuples.

(οπότε έχω τον ακόλουθο πίνακα :

1..20	43	43
21..40	43	43
41..60	43	43
61..80	43	43
81..100	43	43

Τώρα εργάζομαι όπως και στο ερώτημα α) άρα θα έχω συνολικά tuples = (43*43/20)*5 = 529.

3)

T(R) = 20.000, 1 BLOCK χωράει 25 εγγραφες της R άρα
B(R)=20.000/25 = 800

T(S) = 45.000, 1 BLOCK χωράει 30 εγγραφες της S άρα
B(S)=45.000/30 = 1500

1)

NLJ: Ξέρω από τον αλγόριθμο της NLJ ότι θα διαβάσω την μικρότερη σε μέγεθος block σχέση B(X) φορές και από την διαθέσιμη μνήμη και κάθε φορά θα χρησιμοποιώ M-1 σελίδες για το διάβασμα της οπότε για τον έλεγχο join με την άλλη σχέση χρειάζομαι ceil(b(x)/(m-1))*b(y) οπου X, Y εξωτερική και εσωτερική σχέση αντίστοιχα).

Άρα έχω ότι το κόστος ισούται με : B(R) + ceil((B(R)/(M-1))*BS) = 800 + ceil(800/40) * 1500 = 30.800

SMJ:

Αρχικά θα δω αν μπορώ να χρησιμοποιήσω την βελτιωμένη SMJ.

Έχω μ=41 και sqrt(b(r) + b(s))=48 άρα δεν μπορώ να υπολοποιήσω την καλύτερη εκδοχή. Max(b(s), b(r)) = b(s) = 1500 , sqrt(bs) = 39 άρα μπορώ να εκτελέσω τον αλγόριθμο.

Κόστος: 5*(B(R) + B(S))=11.500

HASH JOIN:

Ως εξωτερική σχέση, παίρνω την μικρότερη, δηλαδή εδώ την B(R).

Για την δεύτερη φάση πρέπει να χωράνε τα buckets από την R στην μνήμη οπότε θέλω M^2>B(R) => 1681>800 που ισχύει.

Οπότε το κόστος θα είναι : 3*(B(R) + B(S)) = 3*2300 = 6.900

2)

Αρχικά, ένας «εύκολος» τρόπος για την βελτιστοποίηση του αλγορίθμου είναι η αύξηση μνήμης ώστε να χωράνε και οι 2 σχέσεις για να είναι δυνατή η εκτέλεση του καλύτερου SMJ με κόστος:

3*(B(R) + B(S)) = 6.900.

Ένας άλλος τρόπος είναι η ύπαρξη clustered index στις σχέσεις R,S στο γνώρισμα, ώστε να έχουμε ήδη ταξινομημένες τις σχέσεις μας με αποτέλεσμα το κόστος του smj να πέσει απλά στο διάβασμα των 2 σχέσεων, δηλαδή B(S) + B(S) = 1500 + 800 = 2300

4)

Από την εκφώνηση έχω τα στοιχεία για τα εξής χαρακτηριστικά:

T(δανειζόμενοι) = 10.000, B(δανειζόμενοι) = 1000, block size = 10.000/1000 = 10

T(Βιβλία) = 50.000, B(Βιβλία) = 5000, block size = 50.000/5000 = 10

T(Δανεισμοί) = 300.000, B(Δανεισμοι) = 15.000, block size = 300.000/15.000 = 20

V(βιβλία, εκδότης) = 500

V(δανειζόμενος, ηλικία) = 18

M=20

Στο βιβλία.εκδότης έχω nonclustered index

Στο δανεισμοί.ΚΒ έχω clustered index

(έχω μόνο αυτά τα ευρετήρια)

1:Υποθέτοντας ότι έχω ομοιόμορφη κατανομή στην σχέση βιβλία στο εκδότης θα έχω συνολικά T(βιβλία)/v(βιβλία,εκδότης) = 50.000/500 = 100 tuples με εκδότης = σαββάλας.Στο συγκεκρίμενο γνώρισμα της σχέσης βιβλία έχω non clustered index οπότε το κόστος θα είναι ίσο με τις εγγραφές όπου εκδότης = σαββάλας, δηλαδή cost1 = t(σΕκδότης='Σαββάλας') = 100). Αυτές οι πλειάδες χωράνε σε 100/block size βιβλία = 100 / 10 = 10 blocks.

2) θέλω να βρώ τις πλειάδες που μου επιστρέφει το inlj.

Έχω T(1) * T(Δανεισμοί) / max{V(1,ΚΒ), V(Δανεισμοι, ΚΒ)}

Από την σχέση Βιβλία που παίρνουμε την 1, έχω το ΚΒ ως primary key άρα V(βιβλία, ΚΒ) = 50.000. επομένως αφού 100 εγγραφές με εκδότη = σαββάλα έχω V(1,ΚΒ) = 100. Υποθέτοντας ομοιόμορφη κατανομή έχω και V(Δανεισμοι, ΚΒ) = V(βιβλία, ΚΒ) = 50.000

Επομένως τα tuples που ψάχνω θα είναι 100*300.000/50.000 = 600

Έχω T(σΚΒ=Χ) = T(ΔΑΝΕΙΣΜΟΙ) / V(ΔΑΝΕΙΣΜΟΙ,ΚΒ) = 300.000/50.000 =6. Αυτές οι εγγραφές χωράνε σε μία σελίδα καθώς 6<block size δανειζόμενοι = 20.

Επειδή το ΚΒ έχει clustered index στο ΚΒ και υπολογίζουμε την καλύτερη περίπτωση ότι οι εγγραφές που βρήκαμε πριν θα υπάρχουν στον μικρότερο δυνατό αριθμό από σελίδες, δηλαδή 1 (αφού 6<20 όπως προείπα).Καθώς έχω το clustered index στο δανεισμοί(ΚΒ) δεν έχω κάποιο κόστος να ακολουθήσω κάποιον pointer και οι εγγραφές που επιστρέφονται χωράνε σε 1 σελίδα άρα το index θα «διαβάζει» μια σελίδα.

Τώρα έχω:

Cost2 = cost1 + B(σΕκδότης = ‘Σαββάλας’) + T(σΕκδότης = ‘Σαββάλας’)*ceil(#σελιδών που γυρνάει το index / block size δανεισμοί) = 100 + 0 + 100*ceil(1/20) = 100+ 100*ceil(0.05) = 100+100*1= 200

Το B(σΕκδότης = ‘Σαββάλας’) = 0 καθώς υπολόγισα στο 1 ότι αυτές οι σελίδες είναι 10, άρα χωράνε στην μνήμη που έχω(M=20-10 = 10).

Στην σχέση δανεισμοί έχω 1εγγραφή/συνολικός αριθμός εγγραφών = 1/20. Η Σχέση έχει 3 ορίσματα άρα το ένα όρισμα καταλαμβάνει το 1 / 20*3 = 1/60 του block.(με υπόθεση ότι κάθε όρισμα πιάνει ίδιο χώρο)

Ομοίως για την σχέση βιβλία έχω 1εγγραφή/ συνολικός αριθμός εγγραφών = 1/10.Έχω 6 ορίσματα άρα το ένα όρισμα καταλαμβάνει το 1 / 10*6 = 1/60 του block.

Οπότε αυτές οι εγγραφές θα χωράνε στον αριθμό σελιδών για 600 εγγραφές από την σχέση βίβλία συν ριθμό σελιδών για 600 εγγραφές από την σχέση δανεισμοί μειον τον αριθμό των σελίδων που χρειάζονται 600 εγγραφές με μοναδικό όρισμα το ΚΒ.

Άρα έχω 600/20 + 600/10 – 600 * 1/60 = 30 + 60 – 10 = 80 σελίδες

3)

Για το πΚΔ θέλω να πάρω από τις 600 εγγραφές μόνο το όρισμα ΚΔ.Το όρισμα πιάνει το 1/60 της σελίδας για κάθε εγγραφή και εμείς

ζητάμε μόνο το ΚΔ άρα θα έχουμε 600*1/60 = 10 σελίδες. Λόγω Μ=20 έχω 20 σελίδες από τις 80 που προείπα πριν άρα cost Πκδ = 60+10 = 70

Από το nlj παίρνουμε αυτόν τον αριθμό εγγραφών:

Τ(Πκδ) * Τ(Δανειζόμενοι)) / max(V(Πκδ,κδ) , V(Δανειζόμενοι,ΚΔ))= (600 * 10.000) / 10.000 = 600 εγγραφές

Αυτές οι εγγραφές χωράνε σε 600/block size δανειζόμενοι = 60 σελίδες = Β(3)

Άρα cost3 = Cost2 + CostΠκδ + $\lceil$ Β(Πκδ)/19 $\rceil$ * Β(Δανειζόμενοι) = 200 + 70 + 1 * 1000 = 1270 I/Os

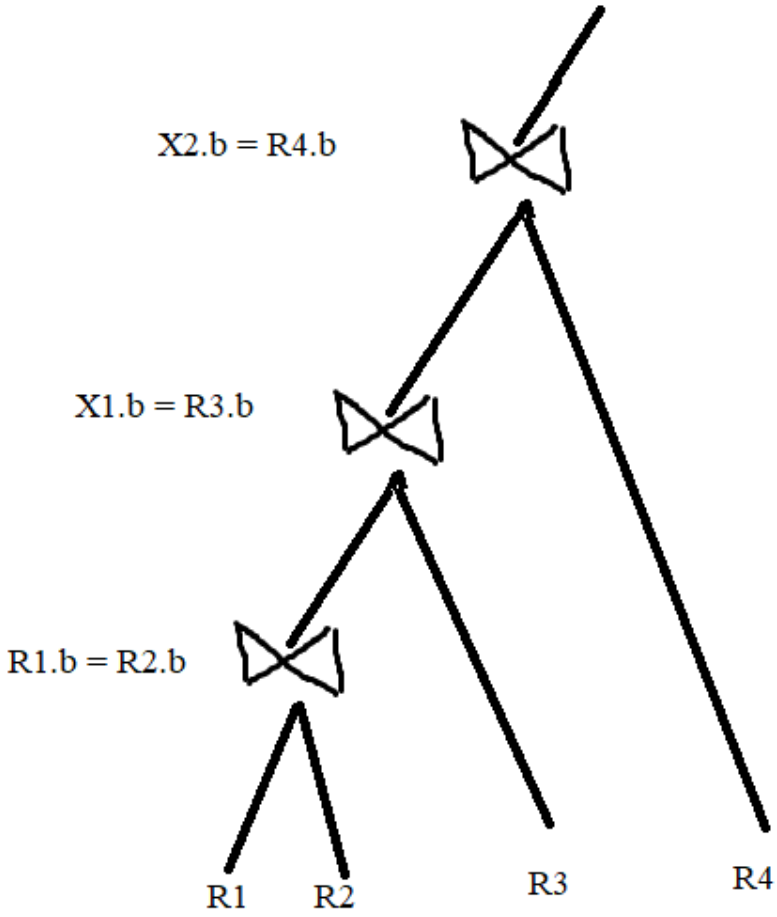
4)

Θεωρώ πως τα δεδομένα κατανέμονται ομοιόμορφα στο όρισμα Ηλικία.Έχω Τ(Ηλικία=Χ)= 600/18 = 100/3(12<Χ<20) Επομένως το Τ(σ12<Ηλικία<20) = 7*Τ(Ηλικία = 13) = 7 * (100/3) = 234.

Άρα cost4= cost3 + Β(3) = 1270 + 60 = 1330.

5)

1)


X2.b = R4.b
X1.b = R3.b
R1.b = R2.b
R1
R2
R3
R4