Εργασία 2
Σχεδιασμός Βάσεων Δεδομένων

Καρράς Κωνσταντίνος 3180076

❖ Άσκηση 1

- **1**

  I. Το ευρετήριο συστάδων έχει κόστος ίσο με τον αριθμό σελίδων της σχέσης / διακριτές τιμές

  a) Κόστος = B(R) / V(R, a) = 20.000 / n IO
  b) Κόστος = B(R) / V(R, a) = (n / 10) * (20.000 / n) = 2.000 IO

  II. Το απλό ευρετήριο κόστος ίσο με τον αριθμό πλειάδων της σχέσης / διακριτές τιμές

  a) Κόστος = T(R) / V(R, a) = 1.000.000 / n IO
  b) Κόστος = T(R) / V(R, a) = (n / 10) * (1.000.000 / n) = 100.000 IO

- **2**

  a) Πρέπει 1.000.000 / n <= 20.000 ⇔ n >= 50
  b) Εδώ δε συμφέρει να χρησιμοποιήσουμε το απλό ευρετήριο, γιατί το κόστος του ευρετηρίου είναι 100.000 IO, ενώ αν κάνουμε table scan το κόστος είναι 20.000 IO.

❖ Άσκηση 2

- **1**

  a) Κόστος = [T(R) * T(S)] / (20 - 1 + 1) = (0*10) / 20 = 0
  Κόστος = [T(R) * T(S)] / (40 - 21 + 1) = (80*100) / 20 = 400
  Κόστος = [T(R) * T(S)] / (60 - 41 + 1) = (100*60) / 20 = 300
  Κόστος = [T(R) * T(S)] / (80 - 61 + 1) = (20*60) / 20 = 60
  Κόστος = [T(R) * T(S)] / (100 - 81 + 1) = (30*0) / 20 = 0
  Τα παραπάνω κόστη είναι υπολογισμένα για τα διαστήματα [1, 20], [21, 40], [41, 60], [61, 80], [81, 100] αντίστοιχα.
  Συνολικά, Κόστος = 0 + 400 + 300 + 60 + 0 = 760

  b) Βάσει της ομοιόμορφης κατανομής αθροίζω όλες τις πλειάδες της κάθε σχέσης ανεξάρτητα από το διάστημα. Και στις 2 περιπτώσεις οι πλειάδες είναι ίσες με 230. Άρα, το κόστος ισούται με:
  Κόστος = 230 * 230 / (100 - 1 + 1) = 529

❖ Άσκηση 3

- **1**

  B(R) = 20.000 / 25 = 800, B(S) = 45.000 / 30 = 1.500

  a) Εξωτερικά τοποθετούμε πάντα τη μικρότερη σχέση
  Τότε, το Κόστος = B(R) + ceil[B(R) / (M-1)] * B(S) =
  = 800 + ceil[800 / 40] * 1.500 = 800 + 20 * 1.500 =
  = **30.800 IO.**

  b) Ο αποδοτικός αλγόριθμος SMJ έχει απαιτήσεις σε μνήμη. Η απαίτηση είναι η εξής: η μνήμη θα πρέπει να έχει περισσότερες σελίδες από τη ρίζα του αθροίσματος των σελίδων και των 2 σχέσεων. Δηλαδή, θα πρέπει να ισχύει 41 >= sqrt(800 + 1.500) ⇔ 41 >= 47,96, πράγμα που δεν ισχύει. Άρα η αποδοτική έκδοση του

SMJ απορρίπτεται. Για την απλή έκδοση απαιτείται η μνήμη να έχει περισσότερες σελίδες από τη ρίζα και των δύο σχέσεων (προφανώς αν μία σχέση είναι μεγαλύτερη (σε αριθμό σελίδων) και ισχύει ο παραπάνω περιορισμός τότε θα ισχύει και για την μικρότερη σχέση). Άρα, πρέπει να ισχύει 41 >= sqrt(1.500) ⇔ 41 >= 38,73
Αυτή η υλοποίηση, όμως, του SMJ έχει κόστος 5 * [B(R) + B(S)] =
= 5 * (800 + 1.500) = **11.500 IO.**

c) Το κόστος εδώ είναι ίσο με 3 * [B(R) + B(S)] = 3 * (800 + 1.500) =
= **6.900 IO.**

Για το συγκεκριμένο join, λοιπόν, συμφέρει περισσότερο ο αλγόριθμος Hash Join.

- **2**

  Ένας τρόπος για να επιτύχουμε το ελάχιστο δυνατό κόστος είναι και η σχέση R και η σχέση S να είναι **ταξινομημένες** στο γνώρισμα c. Έτσι θα χρειαστεί απλά ένα διάβασμα από το δίσκο, όπου οι 2 σχέσεις θα γίνουν join. Τότε, λοιπόν, το κόστος θα είναι ίσο με B(R) + B(S) = 800 + 1.500 = **2300 IO.**

  Δεύτερο τρόπο δεν βρήκα.

❖ Άσκηση 4

<u>Δεδομένα</u>

Δ = Δανειζόμενοι, Β = Βιβλία, Δ’ = Δανεισμοί

Τ(Δ) = 10.000, Β(Δ) = 1.000, Εγγραφές ανά σελίδα = 10
Τ(Β) = 50.000, Β(Β) = 5.000, Εγγραφές ανά σελίδα = 10
Τ(Δ’) = 300.000, Β(Δ’) = 15.000, Εγγραφές ανά σελίδα = 20

V(Β, εκδότης) = 500
Δ.ηλικία >= 7 & Δ.ηλικία <= 24
Μνήμη = 20 σελίδες

nonclustered index on Β(εκδότης)
clustered index on Δ’(ΚΒ)

Τα ευρετήρια στη μνήμη.
Υπάρχει στα δεδομένα ομοιόμορφη κατανομή.

<u>Λύση</u>

- **1**

  Από τη στιγμή που έχω nonclustered index σημαίνει ότι το ευρετήριο αυτό θα μου δείχνει το που είναι αποθηκευμένη η κάθε εγγραφή στο δίσκο. Οι εγγραφές αυτές εκτιμώνται σε Τ(Β) / V(Β, εκδότης) = 50.000 / 500 = **100 εγγραφές**, αφού ισχύει ομοιόμορφη κατανομή. Στη χειρότερη περίπτωση, επειδή το ευρετήριο είναι απλό και όχι συστάδων, θα πρέπει να διαβάσω 100 σελίδες. Άρα, θα έχω **100 ΙΟ.** Αυτές, τις 100 εγγραφές που χωράνε σε 10 σελίδες (κάθε σελίδα χωράει 10 εγγραφές, άρα οι 10 σελίδες θα χωράνε και τις 100 εγγραφές) θα τις κρατήσω στη μνήμη, με αποτέλεσμα να μένουν κενές στη μνήμη άλλες 10 σελίδες.

- **2**

  Επειδή τώρα το δανεισμοί είναι 300.000 εγγραφές και αποτελούνται από δανειζόμενους και βιβλία σημαίνει ότι, αφού έχουμε ομοιόμορφη κατανομή, θα υπάρχουν 300.000 / 50.000 = 6 εγγραφές για κάθε βιβλίο.

Από το 1 έχω πάρει όμως 100 εγγραφές. Για κάθε μία από αυτές τις 100 εγγραφές θα ρωτάω το clustered index των δανεισμών αν υπάρχει η εγγραφή στους δανεισμούς, προκειμένου να γίνει το join. Το ευρετήριο θα μου επιστρέφει τη σελίδα (αφού είναι συστάδων). Άρα το κόστος είναι ίσο με: Κόστος = 0 + 100 * ceil[6 / 20] = **100 IO** (δεν προσθέτω το κόστος των 10 σελίδων από το 1, γιατί αναφέρω πως αυτές βρίσκονται στη μνήμη). Οι εγγραφές που θα γίνουν join είναι 100 * 6 = **600 εγγραφές.** Τώρα, όμως, επειδή αυτές οι 600 εγγραφές πιάνουν χώρο ίσο με 600 * (1 / 10 + 1 / 20) = 1.800 / 20 = **90 σελίδες** και δε χωράνε στη μνήμη θα πάω και θα τις γράψω στο δίσκο (1/10 είναι ο χώρος που πιάνει κάθε εγγραφή του βιβλίου και 1/20 είναι ο χώρος που πιάνει κάθε εγγραφή του δανειζόμενου) . Άρα, συνολικά το κόστος (μαζί με την εγγραφή στο δίσκο) είναι 100 + 90 = **190 σελίδες = 190 IO** .Έτσι, η διαθέσιμη μνήμη θα είναι πάλι 20 σελίδες.

- **3**

  Το κόστος να διαβάσουμε αυτές τις 600 εγγραφές από το 2 είναι ίσο με 1. Και αυτό, διότι ΔΕΝ φέρνω ολόκληρες τις εγγραφές στη μνήμη, αλλά την προβολή του ΚΔ. Για το ΚΔ που έχει περιοχή τιμών από 1 έως 10.000 χρειάζεται απλά ένας integer. Ο integer πιάνει χώρο ίσο με 4 bytes σε κάθε record. Άρα, για 600 εγγραφές που είχα από το 2, ο χώρος που θα καταλαμβάνει η προβολή θα είναι ίσος με 4 bytes * 600 = 2.400 bytes = **2,4 KB**. Ο SQL server έχει page size ίσο με 4 KB, άρα **η προβολή θα χωράει σε μία σελίδα**. Δεδομένου αυτού, από την άλλη πλευρά γίνεται table scan και άρα θα πρέπει να διαβαστούν και οι 1.000 σελίδες που έχουν οι δανειζόμενοι. Στον NLJ εξωτερικά πάει πάντα η μικρότερη σχέση, άρα έτσι προκύπτει το κόστος ίσο με: Κόστος = 1 + ceil[1 / 19] * 1.000 = **1.001 IO**. Το join θα έχει ως αποτέλεσμα κάθε μία από τις 600 εγγραφές από το 2 να γίνουν join με μία ακριβώς εγγραφή από τους δανειζόμενους αφού το ΚΔ είναι πρωτεύων κλειδί. Άρα συνολικά **600 εγγραφές**. Αυτές θα περιέχουν ότι περιέχουν και οι πλειάδες των δανειζόμενων και επιπλέον το ΚΔ, που είναι integer και πιάνει 4 bytes. Γνωρίζω ότι κάθε σελίδα χωράει 10 εγγραφές από δανειζόμενους και η σελίδα έχει μέγεθος 4KB. Επομένως, κάθε εγγραφή θα έχει μέγεθος ίσο με 4KB / 10 εγγραφές = 400 bytes. Σε αυτό θα προστεθεί το ΚΔ που είναι 4 bytes. Άρα, κάθε εγγραφή θα έχει

κόστος ίσο με 404 bytes. Και έχω 600 εγγραφές. Επομένως, αυτές οι 600 εγγραφές θα χωράνε σε (600 * 404 bytes) / 4KB = ceil[59,18 σελίδες] = **60 σελίδες** και επειδή στη μνήμη δεν χωράνε θα τις γράψω στο δίσκο. Άρα, το επιπλέον κόστος γραψίματος στο δίσκο είναι **60 IO.** Το συνολικό κόστος μαζί με το γράψιμο στο δίσκο θα είναι 1.001 + 60 = **1.061 IO.**

- **4**

  Αυτές τις 600 εγγραφές που έχω από το 3, που χωράνε σε **60 σελίδες**, θα τις διαβάσω και θα επιλέξω τις ηλικίες από 13 έως και 19. Οι ηλικίες κατανέμονται ομοιόμορφα, άρα θα έχω εγγραφές ίσες με 600 * (19 – 13 + 1) / (24 – 7 + 1) = 233,33 εγγραφές = **234 εγγραφές**. Το κόστος IO ανέρχεται σε **60 IO** , όσος και ο αριθμός των σελίδων που πρέπει να διαβάσω από το δίσκο.

❖ Άσκηση 5

- **1**

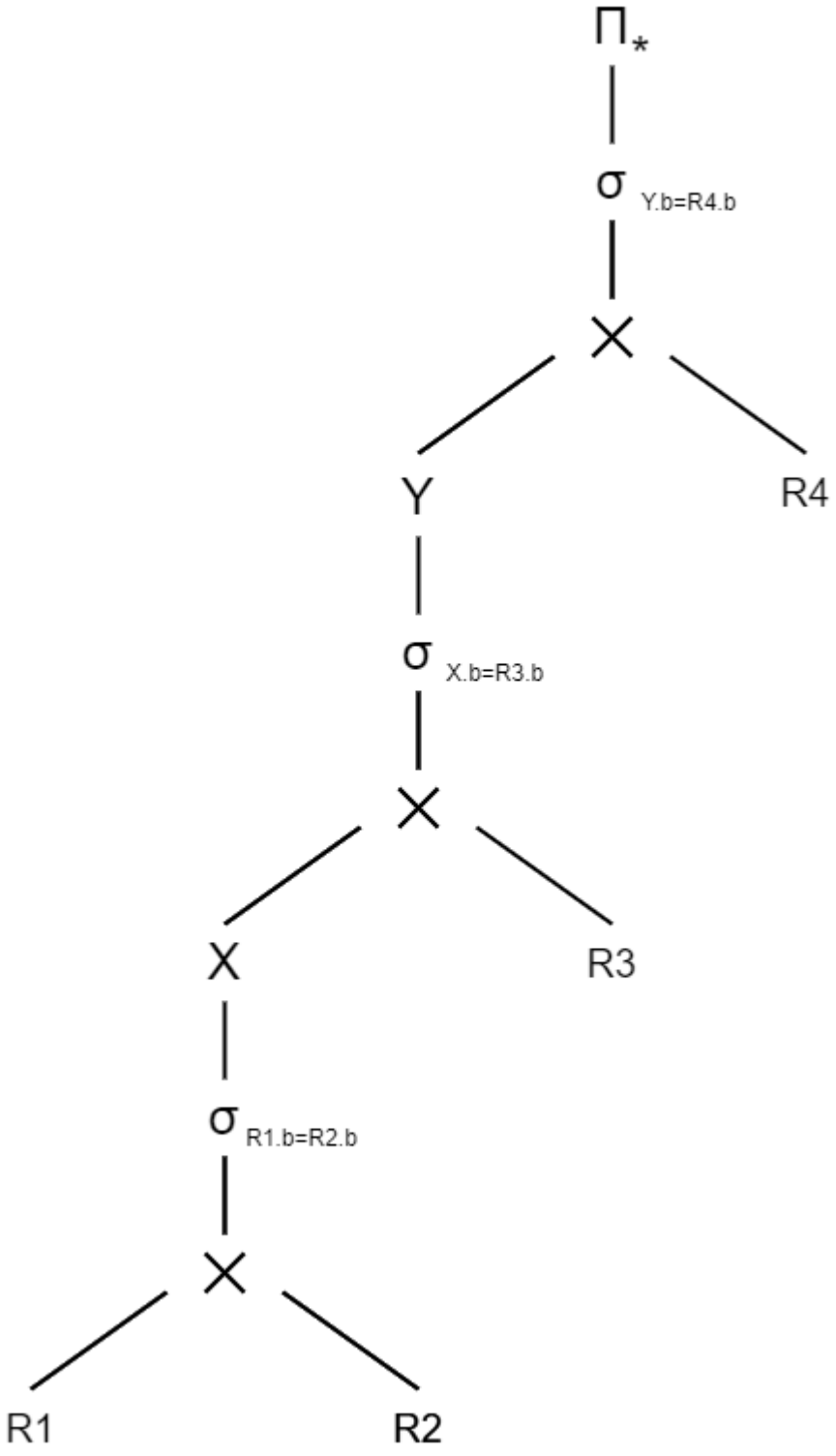
  <u>Σχεσιακή Άλγεβρα</u>

  $\Pi_{*}[\sigma_{R4.b=Y.b}(R4 \times [\sigma_{R3.b=X.b}(R3 \times [\sigma_{R1.b=R2.b}(R1 \times R2)])])]$

  Όπου

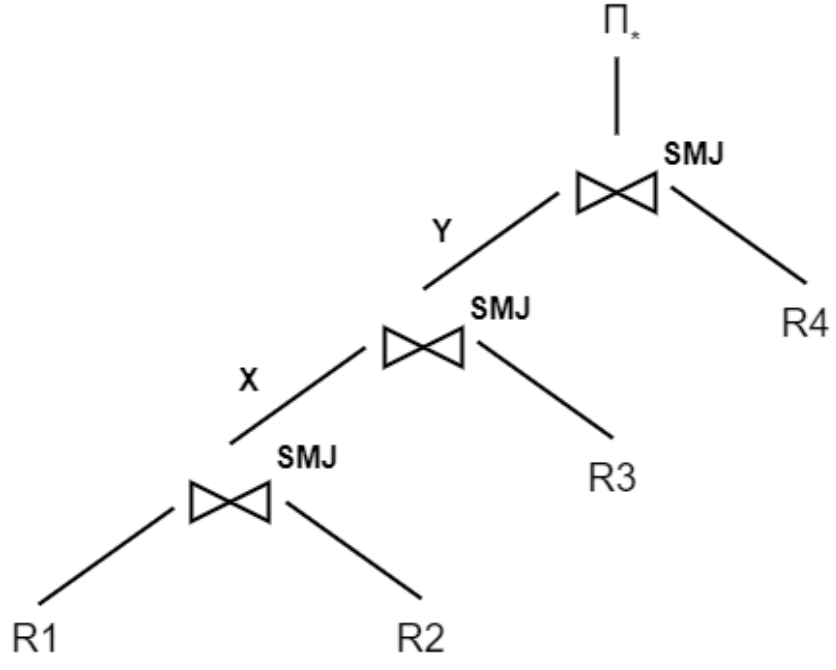
  - $X = \sigma_{R1.b=R2.b}(R1 \times R2)$
  - $Y = \sigma_{R3.b=X.b}(R3 \times [\sigma_{R1.b=R2.b}(R1 \times R2)])$

<u>Λογικό Πλάνο</u>



- **2**

  Το πρώτο λογικό πλάνο ήταν επεξηγηματικό της έκφρασης της σχεσιακής άλγεβρας που έγραψα στο 1. Το επόμενο που ακολουθεί είναι αυτό που δείχνει ποιος θα είναι ο αλγόριθμος ισοσύνδεσης που θα χρησιμοποιηθεί.



  Ο αλγόριθμος ισοσύνδεσης που θα χρησιμοποιηθεί θα είναι ο Sort Merge Join (SMJ). Και αυτό, επειδή έχω στη διάθεσή μου clustered indexes στο γνώρισμα b της κάθε σχέσης. Αυτό πρακτικά σημαίνει ότι κάθε σχέση είναι **ταξινομημένη** ως προς το b. Το κέρδος από αυτό είναι ότι **κάθε σχέση** θα διαβάζεται μία φορά από το δίσκο με **table scan**. Άρα το κόστος κάθε join θα είναι ίσο με B(T) + B(S) (όπου T και S δύο οποιεσδήποτε σχέσεις)

- **3**

  Κόστος X = B(R1) + B(R2)
  Κόστος γραψίματος του X στο δίσκο = B(R1) + B(R2)

  Κόστος Y = B(X) + B(R3)
  Κόστος γραψίματος του Y στο δίσκο = B(X) + B(R3)

  Τελικό κόστος = B(Y) + B(R4)
  Δεν χρειάζεται γράψιμο στο δίσκο.

Συνολικό Κόστος = Τελικό κόστος + Κόστος γραψίματος του Υ στο δίσκο + Κόστος Υ + Κόστος γραψίματος του Χ στο δίσκο + Κόστος Χ =
= B(Y) + B(R4) + B(X) + B(R3) + B(X) + B(R3) + B(R1) + B(R2) + B(R1) + B(R2)
= B(Y) + B(R4) + 2 * B(X) + 2 * B(R3) + 2 * B(R1) + 2 * B(R2)
= (B(X) + B(R3)) + B(R4) + 2 * (B(R1) + B(R2)) + 2 * B(R3) + 2 * B(R1) + 2 * B(R2)
= [((B(R1) + B(R2)) + B(R3)] + B(R4) + 2 * B(R3) + 4 * B(R2) + 4 * B(R1)
= *B(R4) + 3 * B(R3) + 5 * B(R2) + 5 * B(R1)*

Να σημειωθεί πως δεν χρησιμοποίησα INLJ γιατί το κόστος πχ για το join των R1, R2 θα ήταν ίσο με B(R1) + T(R1) * ceil[X/#εγγραφές ανά σελίδα]. Δεδομένου ότι το T(R1) κατά πάσα πιθανότητα θα είναι απαγορευτικά μεγάλο (τα ενδιάμεσα αποτελέσματα από την υπόθεση δε χωράνε στη μνήμη) το κόστος θα ήταν αρκετά μεγαλύτερο.
Επιπλέον, αν και δεν αναφέρεται κάτι σχετικό με το μέγεθος των σχέσεων στην εκφώνηση, **θεωρώ ότι τις λιγότερες πλειάδες θα τι έχουν με τη σειρά η R1, μετά η R2, μετά η R3 και τέλος η R4.**