1η Εργασία στη Σχεδίαση ΒΔ 2024-2025

➢ **1ο** Ερώτημα: Ευρετήριο με B+ δέντρο

1) Ποιος είναι ο παράγοντας ομαδοποίησης;

bfr = B/R = 1024/400 = 2.

2) Πόσα byte μένουν αναξιοποίητα σε κάθε block, λόγω της μη εκτεινόμενης καταχώρησης;

1024-800 = 224 bytes μένουν αναξιοποίητα.

3) Πόσα block απαιτούνται για την αποθήκευση ολόκληρου του αρχείου;

Απαιτούνται 50000/2 = 25000 blocks.

4) Αν το κόστος μεταφοράς ενός block από/προς το δίσκο είναι 0,05ms, ποιο είναι το κόστος (χρόνος) αναζήτησης με συνθήκη ισότητας; Ποιο το κόστος (χρόνος) αναζήτησης με συνθήκη εύρους;

Κόστος αναζήτησης με συνθήκη ισότητας:

Δεδομένου ότι οι εγγραφές είναι αταξινόμητες, προκειμένου να βρω τη χειρότερη περίπτωση υπολογίζω 25000*0.05 = 1250 ms. Για τη μέση περίπτωση κάνω 1250/2 = 625 ms.

Κόστος αναζήτησης με συνθήκη εύρους:

Και εδώ, η χειρότερη περίπτωση είναι 25000*0.05 = 1250 ms. Αλλά, αν υποθέσω πως το εύρος είναι πχ. 10%, τότε το κόστος αναζήτησης θα είναι 25000*0.05*0.1 = 125ms.

Βέβαια, αν οι εγγραφές ήταν ταξινομημένες (πχ. με B+ δέντρο), θεωρώ πως η αναζήτηση με συνθήκη εύρους θα ήταν πιο αποτελεσματική.

5) Πάνω σε ένα πεδίο char(20) που μπορεί και να περιέχει διπλότυπα, έχει χτιστεί ευρετήριο με χρήση B+ δέντρου. Κάθε δείκτης προς τις εγγραφές του αρχείου έχει μέγεθος 11 bytes, κάθε δείκτης προς block του ευρετηρίου μαζί και ο δείκτης προς επόμενο φύλλο έχει μέγεθος 16 bytes.

5α) Ποια είναι η τάξη και ποιο το fanout του B+ δέντρου, ώστε κάθε κόμβος να είναι όσο το δυνατόν γεμάτος;

Θεωρώ ότι:

- στα φύλλα εφαρμόζεται απλά συνδεδεμένη λίστα.
- το fanout είναι δυναμικό και όχι σταθερό (διαφορετικό μεταξύ φύλλων-εσωτερικών κόμβων)
- ως "τάξη" εννοείται η ελάχιστη χωρητικότητα του B+ δέντρου
- οι pointers των εγγραφών δείχνουν απευθείας προς block εγγραφών και
- κόμβος = block μεγέθους 1024 bytes

Από τις διαφάνειες γνωρίζουμε το τύπο fanout = 2d+1 και πως η τάξη(*d*) είναι η ελάχιστη χωρητικότητα κόμβων του δέντρου. Από την εκφώνηση, το block είναι μεγέθους 1024 bytes, ο δείκτης προς εγγραφή(*στα φύλλα*) είναι 11 bytes, ο δείκτης προς block(*εσωτερικών κόμβων*) είναι 16 bytes, το κλειδί είναι 20 bytes και ο διαθέσιμος χώρος είναι 1024-16 = 1008 bytes.

Στα φύλλα, το μέγεθος κάθε εγγραφής είναι 31 bytes(*20+11*), ο μέγιστος αριθμός εγγραφών ανά block φύλλου είναι 1008/31(*στρογγυλοποίηση προς τα κάτω*) = 32, οπότε η τάξη(*d*) για τα φύλλα είναι 32/2(*στρογγυλοποίηση προς τα κάτω*) = 16 και το fanout είναι 2*16+1 = 33.

Στους εσωτερικούς κόμβους, το μέγεθος κάθε εγγραφής είναι 36 bytes(*20+16*), ο μέγιστος αριθμός παιδιών ανά block εσωτερικού κόμβου είναι 1008/36(*στρογγυλοποίηση προς τα κάτω*) = 28, οπότε η τάξη(*d*) για τους εσωτερικούς κόμβους είναι 28/2(*στρογγυλοποίηση προς τα κάτω*) = 14 και το fanout είναι 2*14+1 = 29.

5β) Πόσα επίπεδα υπάρχουν στο B+ δέντρο (συμπεριλαμβανομένου και του τελευταίου επιπέδου) και πόσοι κόμβοι ανά επίπεδο; Θεωρήστε ότι οι εσωτερικοί κόμβοι είναι γεμάτοι κατά τα 2/3.

Φύλλα του B+ δέντρου:
Κάθε εγγραφή στο ευρετήριο με χρήση B+ δέντρου για φύλλο είναι 20(*char(20)*)+11(*δείκτης προς εγγραφή*) = 31 bytes. Ο χώρος που χρησιμοποιείται ανά block είναι 1024-16(*bytes δείκτη*) = 1008 bytes. Υπολογίζω 1008/31 = 32(*στρογγυλοποίηση προς τα κάτω*) εγγραφές ανά block, οπότε κάνοντας

50000/32(*στρογγυλοποίηση προς τα πάνω*), βρίσκω 1563 block για τα φύλλα του B+ δέντρου.

Εσωτερικοί κόμβοι:
Κάθε εγγραφή στο ευρετήριο με χρήση B+ δέντρου για εσωτερικούς κόμβους είναι 20(*char(20)*)+16(*δείκτης προς block και επόμενο φύλλο*) = 36 bytes. Ο χώρος που χρησιμοποιείται ανά block εξακολουθεί να είναι 1024-16(*bytes δείκτη*) = 1008 bytes. Υπολογίζω 1008/36 * 2/3 = 18(*στρογγυλοποίηση προς τα κάτω*) εγγραφές ανά block για τους εσωτερικούς κόμβους. Κάνοντας 1563/18(*στρογγυλοποίηση προς τα πάνω*), βρίσκω 87 κόμβους για το 1ο επίπεδο των εσωτερικών κόμβων και υπολογίζοντας 87/18(*στρογγυλοποίηση προς τα πάνω*), βρίσκω 5 κόμβους για το 2ο επίπεδο των εσωτερικών κόμβων.

Ρίζα:
Για τη ρίζα κάνω υπολογισμούς, με παρόμοια λογική όπως και προηγουμένως. Υπολογίζω 5/18(*στρογγυλοποίηση προς τα πάνω*) = 1. Άρα, η ρίζα έχει 1 κόμβο.

Επομένως, υπάρχουν 4 επίπεδα στο B+ δέντρο. Το 1ο επίπεδο (*φύλλα*) με 1563 κόμβους, το 2ο επίπεδο (*1ο επίπεδο εσωτερικών κόμβων*) με 87 κόμβους, το 3ο επίπεδο (*2ο επίπεδο εσωτερικών κόμβων*) με 5 κόμβους και το 4ο και τελευταίο επίπεδο (*Ρίζα*) με 1 κόμβο.

5γ) Ποιο είναι το μέγεθος (σε block) του ευρετηρίου συνολικά;

Θεωρώ πως οι εσωτερικοί κόμβοι εξακολλουθούν να είναι γεμάτοι κατά τα 2/3, άρα το fanout για τους εσωτερικούς κόμβους είναι 2/3*29=19.

Βάσει των δεδομένων από τα προηγούμενα ερωτήματα, γνωρίζω τα εξής:

1. 50000 εγγραφές με μέσο όρο μεγέθους 400 bytes, 1024 bytes ανά block (εκφώνηση)
2. 2 εγγραφές ανά block (1ο ερώτημα)
3. 25000 blocks για αποθήκευση του αρχείου (3ο ερώτημα)

1ο επίπεδο:
Όπου, κάθε εγγραφή στο ευρετήριο για φύλλο είναι 20(*char(20)*)+11(*δείκτης προς εγγραφή*) = 31 bytes. Ο χώρος που χρησιμοποιείται ανά block είναι 1024-16(*bytes δείκτη*) = 1008 bytes. Υπολογίζω 1008/31 = 32(*στρογγυλοποίηση προς τα κάτω)* εγγραφές ανά block, οπότε κάνοντας 50000/32(*στρογγυλοποίηση προς τα πάνω*), βρίσκω 1563 block για τα φύλλα του B+ δέντρου.

2ο επίπεδο:
Με παρόμοια λογική, κάθε εγγραφή στο ευρετήριο για μη-φύλλο είναι 20(*char(20)*)+16(*δείκτης προς block και επόμενο φύλλο*) = 36 bytes. Ο χώρος που χρησιμοποιείται ανά block είναι 1024-16(*bytes δείκτη*) = 1008 bytes. Υπολογίζω 1008/36 = 28(*στρογγυλοποίηση προς τα κάτω*) εγγραφές ανά block, οπότε κάνοντας 1563/19(*στρογγυλοποίηση προς τα πάνω*), βρίσκω 83 block για τα μη-φύλλα του B+ δέντρου.

3ο επίπεδο:
Υπολογίζω 83/19(*στρογγυλοποίηση προς τα πάνω*) = 5 block, και

η ρίζα(*η οποία δείχνει στα 2 block του 3ου επιπέδου*)**:**
1 block.

Άρα, το συνολικό μέγεθος του ευρετηρίου (σε block) είναι:
1563(*φύλλα*) + (83+5(*μη-φύλλα*))+1(*ρίζα*) = 1652 block.

5δ) Ποιο θα είναι το κόστος (χρόνος) αναζήτησης για κλειδιά με τιμή (για το πεδίο αναζήτησης) μεγαλύτερη ή ίση κάποιας τιμής που εμφανίζεται μία και μόνο φορά στο προτελευταίο φύλλο; Θεωρήστε και πάλι ότι το κόστος μεταφοράς ενός block από/προς το δίσκο είναι 0,05ms.

Ο αλγόριθμος αναζήτησης του B+ δέντρου θα διασχίσει 4 block μέχρι να φτάσει στο επίπεδο των φύλλων(*στο προτελευταίο φύλλο*). Το κόστος του μονοπατιού που θα διασχίσει είναι 4(*block*)*0.05ms(*κόστος μεταφοράς*) = 0.2ms. Έπειτα, θα διαβάσει το προτελευταίο και το τελευταίο φύλλο(*για τη περίπτωση διπλότυπων*). Άρα, διαβάζονται 2 ακόμα block, οπότε υπολογίζω 2(*block*)*0.05ms(*κόστος μεταφοράς*) = 0.1ms.

Οπότε, το συνολικό κόστος αναζήτησης είναι 0.2ms + 0.1ms = 0.3ms.

➢ **2ο** Ερώτημα: Υλοποίηση με JDBC

Έστω ότι θέλουμε για κάθε υπάλληλο να μας επιστραφεί το επώνυμό του, το όνομά του, καθώς και το όνομα του τμήματος στο οποίο απασχολείται.

1) Εκφράστε το παραπάνω ερώτημα σε SQL

```
SELECT fname, lname, dname FROM employee
JOIN department ON dno=dnumber;
```

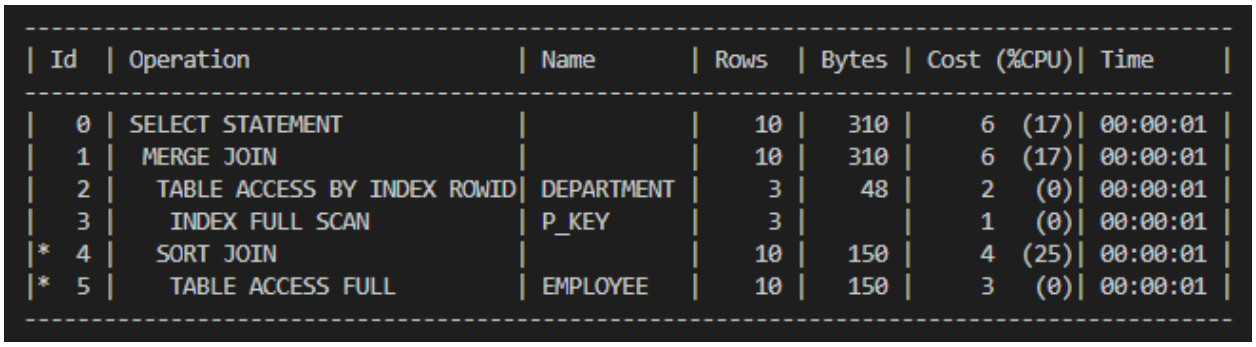
2) Φτιάξτε ένα πρόγραμμα Java που να εκτυπώνει τα αποτελέσματα της παραπάνω επερώτησης.

Το αρχείο με το πρόγραμμα Java ονομάζεται DBApplication.java και επισυνάπτεται στον φάκελο project1.

3) Μέσω του ίδιου προγράμματος Java, εκτυπώστε το πλάνο εκτέλεσης της παραπάνω επερώτησης. Εξηγήστε στην αναφορά σας κάθε βήμα και κάθε στοιχείο του πλάνου.

Ανάλυση του Πλάνου Εκτέλεσης:
Id: αρίθμηση των βημάτων που εκτελέστηκαν
Operation: η λειτουργία που εκτελέστηκε ανά βήμα (πχ. select, join)
Name: το όνομα του αντικειμένου που χρησιμοποιήθηκε ανά βήμα (πχ. πίνακας, κλειδί)
Rows: το πλήθος γραμμών που εκτυπώθηκαν ανά βήμα
Bytes: ο εκτιμώμενος χώρος σε bytes που χρησιμοποιήθηκαν ανά βήμα
Cost(%CPU): το εκτιμώμενο κόστος εκτέλεσης και το ποσοστό αξιοποίησης της CPU ανά βήμα, στη γραμμή με Id=0 υποδηλώνεται το συνολικό κόστος του query (6)
Time: ο εκτιμώμενος χρόνος που πήρε για να εκτελεστεί το κάθε βήμα

Στο διάγραμμα που ακολουθεί, συμπεραίνεται ότι για την συνένωση των πινάκων EMPLOYEE και DEPARTMENT μέσω του ROWID, έγινε χρήση του MERGE JOIN και ο δείκτης που χρησιμοποιήθηκε για την συνένωση αυτή είναι το κλειδί P_KEY. Επίσης, έγινε ταξινόμηση του πίνακα EMPLOYEE, ο οποίος σαρώθηκε πλήρως (TABLE ACCESS FULL). Σηγκεκριμένα, σαρώθηκε κάθε εγγραφή του πίνακα μέχρι να βρεθούν όλα τα δεδομένα που πληρούν τα κριτήρια του query (dno=dnumber). Το INDEX FULL SCAN επεξεργάζεται τα φύλλα block του δείκτη (P_KEY), μέχρι να βρει το πρώτο φύλλο block. Χρησιμοποιείται όταν όλες οι στήλες αναγκαίες για την ικανοποίηση της συνθήκης, βρίσκονται στον δείκτη και προκύπτει λιγότερο κόστος από ότι θα προέκυπτε για το σκανάρισμα του πίνακα.
Στη πράξη, τα βήματα του πλάνου εκτελούνται από κάτω προς τα πάνω και έτσι πρέπει να διαβάζονται.

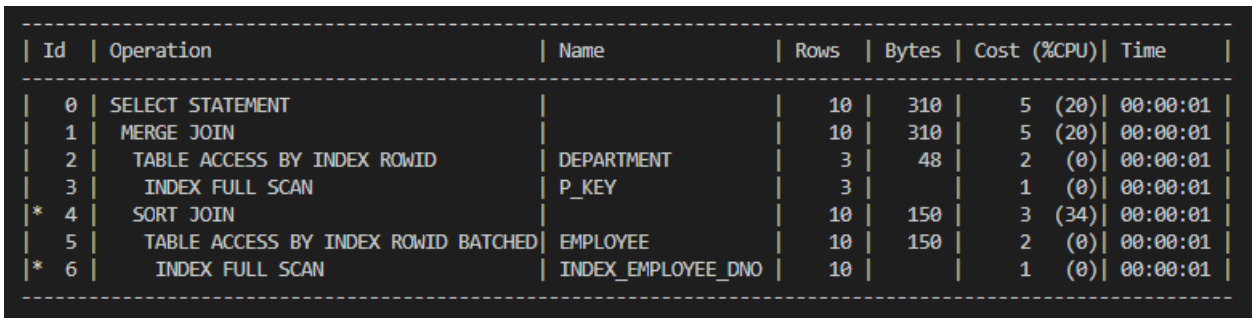


Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10	310	6 (17)	00:00:01
1	MERGE JOIN		10	310	6 (17)	00:00:01
2	TABLE ACCESS BY INDEX ROWID	DEPARTMENT	3	48	2 (0)	00:00:01
3	INDEX FULL SCAN	P_KEY	3		1 (0)	00:00:01
* 4	SORT JOIN		10	150	4 (25)	00:00:01
* 5	TABLE ACCESS FULL	EMPLOYEE	10	150	3 (0)	00:00:01

Συνολικό Κόστος: 6.

4) Δημιουργήστε μέσω του ίδιου προγράμματος Java ένα κατάλληλο ευρετήριο ώστε η εκτέλεση της παραπάνω επερώτησης να γίνει αποδοτικότερα. Εκτυπώστε ξανά το νέο πλάνο εκτέλεσης της ίδιας επερώτησης. Αιτιολογήστε στην αναφορά σας γιατί επιλέξατε αυτό το ευρετήριο και τι όφελος είχε η δημιουργία του ευρετηρίου αυτού.

Χρησιμοποίησα το default index της Oracle, δηλαδή B-δέντρο λόγω της αξιοπιστίας και αποδοτικότητάς του. Συκρίνοντας το προηγούμενο Explain Plan (χωρίς ευρετήριο) με το παρακάτω, φαίνεται πως αντί να σαρωθεί ολόκληρος ο πίνακας EMPLOYEE, χρησιμοποιείται το ευρετήριο INDEX_EMPLOYEE_DNO για τη προσπέλαση των δεδομένων του πίνακα. Αυτό έχει ως αποτέλεσμα τη μείωση του κόστους και του ποσοστού αξιοποίησης της CPU. Η αποτελεσματικότητα που προσφέρει το ευρετήριο θα γινόταν ακόμη πιο εμφανής αν ο πίνακας EMPLOYEE είχε μεγαλύτερο όγκο δεδομένων.



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10	310	5 (20)	00:00:01
1	MERGE JOIN		10	310	5 (20)	00:00:01
2	TABLE ACCESS BY INDEX ROWID	DEPARTMENT	3	48	2 (0)	00:00:01
3	INDEX FULL SCAN	P_KEY	3		1 (0)	00:00:01
* 4	SORT JOIN		10	150	3 (34)	00:00:01
5	TABLE ACCESS BY INDEX ROWID BATCHED	EMPLOYEE	10	150	2 (0)	00:00:01
* 6	INDEX FULL SCAN	INDEX_EMPLOYEE_DNO	10		1 (0)	00:00:01

Συνολικό Κόστος: 5.