Ευρετήρια

Ευρετήρια

Η δομή προσπέλασης (access path) συνήθως ορίζεται σε ένα πεδίο του αρχείου που λέγεται **πεδίο ευρετηριοποίησης** (indexing field)

Πρωτεύον ευρετήριο (primary index): ορισμένο στο κλειδί διάταξης του αρχείου

Ευρετήριο συστάδων (clustering index): ορισμένο στο πεδίο διάταξης το οποίο όμως δεν είναι κλειδί

Δευτερεύον ευρετήριο (secondary index): ορισμένο σε πεδία διαφορετικά του κλειδιού διάταξης

Πόσα ευρετήρια σε ένα αρχείο δεδομένων μπορεί να έχουμε ;

Ευρετήρια

Πυκνό ευρετήριο: μια καταχώρηση για κάθε εγγραφή του δίσκου

Μη πυκνό ευρετήριο

Πρωτεύον Ευρετήριο

Πρωτεύον ευρετήριο (primary index): ορισμένο στο κλειδί διάταξης του αρχείου

Για κάθε block του αρχείου (μη πυκνό ευρετήριο) η εγγραφή i του ευρετηρίου είναι της μορφής **(<K(i), P(i)>)** όπου:

- **K(i):** η τιμή του πρωτεύοντος κλειδιού της πρώτης εγγραφής του block (*άγκυρα* του block)
- **P(i):** δείκτης προς το block

Πρωτεύον Ευρετήριο

- Το ευρετήριο αρχείου είναι ένα *διατεταγμένο* αρχείο με σταθερού μήκους εγγραφές

- Το πρωτεύον ευρετήριο είναι ένα *μη πυκνό* ευρετήριο

- Το *μέγεθος* του αρχείου ευρετηρίου είναι μικρότερο από του αρχείου δεδομένων.

Πρωτεύον Ευρετήριο

Παράδειγμα: Έστω διατεταγμένο αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, όπου το πεδίο κλειδιού διάταξης έχει μέγεθος V = 9 bytes, μη εκτεινόμενη καταχώρηση. Κατασκευάζουμε πρωτεύον ευρετήριο, μέγεθος δείκτη block P = 6 bytes

Μέγεθος αρχείου δεδομένων: 3.000 blocks

Μέγεθος αρχείου ευρετηρίου: 45 blocks

Πρωτεύον Ευρετήριο

- **Αναζήτηση**

  Δυαδική αναζήτηση στο πρωτεύον ευρετήριο

  Ανάγνωση του block από το αρχείο δεδομένων

Πρωτεύον Ευρετήριο

Παράδειγμα: Έστω διατεταγμένο αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, όπου το πεδίο κλειδιού διάταξης έχει μέγεθος V = 9 bytes, μη εκτεινόμενη καταχώρηση. Κατασκευάζουμε πρωτεύον ευρετήριο, μέγεθος δείκτη block P = 6 bytes

Μέγεθος αρχείου δεδομένων: 3.000 blocks - Μέγεθος αρχείου ευρετηρίου: 45 blocks

Αναζήτηση χωρίς ευρετήριο: $\lceil \log 3.000 \rceil = 12$ blocks

Αναζήτηση με ευρετήριο: $\lceil \log 45 \rceil + 1 = 7$ blocks

Πρωτεύον Ευρετήριο

- **Εισαγωγή εγγραφής**

  αλλαγές και στο πρωτεύον ευρετήριο

  μη διατεταγμένο αρχείο υπερχείλισης

  συνδεδεμένη λίστα εγγραφών υπερχείλισης

- **Διαγραφή εγγραφής**

  αλλαγές και στο πρωτεύον ευρετήριο

  χρήση σημαδιών διαγραφής

Ευρετήριο Συστάδων

Ευρετήριο συστάδων (clustering index): ορισμένο στο πεδίο διάταξης το οποίο όμως δεν είναι κλειδί

Υπάρχει μια εγγραφή για κάθε διακεκριμένη τιμή του πεδίου διάταξης (συστάδας) του αρχείου που περιέχει:

- την τιμή αυτή
- ένα δείκτη προς το πρώτο block του αρχείου δεδομένων που περιέχει μια εγγραφή με την τιμή αυτή στο πεδίο συστάδας

Ευρετήριο Συστάδων

• Το *μέγεθος* του αρχείου ευρετηρίου είναι μικρότερο από του αρχείου δεδομένων.

Παράδειγμα: Έστω διατεταγμένο αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο διάταξης έχει μέγεθος V = 9 bytes και υπάρχουν 1000 διαφορετικές τιμές και οι εγγραφές είναι ομοιόμορφα κατανεμημένες ως προς τις τιμές αυτές. Υποθέτουμε ότι χρησιμοποιούνται άγκυρες block, κάθε νέα τιμή του πεδίου διάταξης αρχίζει στην αρχή ενός νέου block. Κατασκευάζουμε ευρετήριο συστάδων, μέγεθος δείκτη block P = 6 bytes

Μέγεθος αρχείου δεδομένων: 3.000 blocks

Μέγεθος ευρετηρίου συστάδων: 15 blocks

Ευρετήριο Συστάδων

• **Αναζήτηση**

Δυαδική αναζήτηση στο ευρετήριο

Ανάγνωση block από το αρχείο δεδομένων

Ευρετήριο Συστάδων

Παράδειγμα: Έστω διατεταγμένο αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο διάταξης έχει μέγεθος V = 9 bytes και υπάρχουν 1000 διαφορετικές τιμές και οι εγγραφές είναι ομοιόμορφα κατανεμημένες ως προς τις τιμές αυτές. Υποθέτουμε ότι χρησιμοποιούνται άγκυρες block, κάθε νέα τιμή του πεδίου διάταξης αρχίζει στην αρχή ενός νέου block. Κατασκευάζουμε ευρετήριο συστάδων, μέγεθος δείκτη block P = 6 bytes

Μέγεθος αρχείου δεδομένων: 3.000 blocks - Μέγεθος αρχείου ευρετηρίου: 15 blocks

Αναζήτηση χωρίς ευρετήριο: $\lceil \log 3.000 \rceil$ + ταιριάσματα (= 3) ≈ 15 blocks

Αναζήτηση με ευρετήριο: $\lceil \log 15 \rceil$ + 3 = 7 blocks

Δευτερεύον Ευρετήριο

Δευτερεύον ευρετήριο (secondary index): ορισμένο σε πεδία διαφορετικά του κλειδιού διάταξης

Δευτερεύον Ευρετήριο

Περίπτωση 1: Το πεδίο ευρετηριοποίησης είναι κλειδί (καλείται και δευτερεύον κλειδί)

Υπάρχει μια εγγραφή για κάθε εγγραφή του αρχείου που περιέχει:

- την τιμή του δευτερεύοντος κλειδιού για αυτήν την εγγραφή
- ένα δείκτη προς το block (ή την εγγραφή) του αρχείου δεδομένων που περιέχει την εγγραφή με την τιμή αυτή

Δευτερεύον Ευρετήριο

- Το ευρετήριο αρχείου είναι ένα *διατεταγμένο* αρχείο με σταθερού μήκους εγγραφές
- Το δευτερεύον ευρετήριο είναι ένα *πυκνό* ευρετήριο
- Το *μέγεθος* του δευτερεύοντος ευρετηρίου είναι μικρότερο από του αρχείου δεδομένων (αν και μεγαλύτερο από το πρωτεύον).

Δευτερεύον Ευρετήριο

Παράδειγμα: Έστω αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V = 9 bytes αλλά δεν είναι πεδίο διάταξης,. Κατασκευάζουμε δευτερεύον ευρετήριο, μέγεθος δείκτη block P = 6 bytes

Μέγεθος αρχείου δεδομένων: 3.000 blocks

Μέγεθος αρχείου ευρετηρίου: 442 blocks

45 για πρωτεύον

Δευτερεύον Ευρετήριο

- **Αναζήτηση**

Δυαδική αναζήτηση στο δευτερεύον ευρετήριο

Ανάγνωση του block από το αρχείο δεδομένων

Δευτερεύον Ευρετήριο

Παράδειγμα: Έστω αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V = 9 bytes αλλά δεν είναι πεδίο διάταξης,. Κατασκευάζουμε δευτερεύον ευρετήριο, μέγεθος δείκτη block P = 6 bytes

Μέγεθος αρχείου δεδομένων: 3.000 blocks - Μέγεθος αρχείου ευρετηρίου: 442 blocks

Αναζήτηση χωρίς ευρετήριο: 3.000/2 = 1500 blocks

Αναζήτηση με ευρετήριο: $\lceil \log 442 \rceil + 1 = 10$ blocks

Για πρωτεύον ήταν 45 και 7 blocks αντίστοιχα

Δευτερεύον Ευρετήριο

Περίπτωση 2: Το πεδίο ευρετηριοποίησης δεν είναι κλειδί

1. Πυκνό ευρετήριο: μία καταχώρηση για κάθε εγγραφή

2. Μεταβλητού μήκους εγγραφές με ένα επαναλαμβανόμενο πεδίο για το δείκτη

3. Μία εγγραφή ευρετηρίου για κάθε τιμή του πεδίου ευρετηριοποίησης + ένα ενδιάμεσο επίπεδο για την διαχείριση των πολλαπλών δεικτών

Δευτερεύον Ευρετήριο

- **Αναζήτηση**

Δυαδική αναζήτηση στο δευτερεύον ευρετήριο

Ανάγνωση του block (ή των blocks) από το ενδιάμεσο επίπεδο

Ανάγνωση των blocks (συνήθως τόσα όσες οι εγγραφές που ταιριάζουν) από το αρχείο δεδομένων

Δευτερεύον Ευρετήριο

- **Εισαγωγή**

Πολύ απλή αν δεν αφορά εισαγωγή νέας τιμής στο ευρετήριο

- Εύκολη η λογική διάταξη των εγγραφών με βάση το πεδίο ευρετηριοποίησης
- Ανακτήσεις με σύνθετες συνθήκες

χτίζουμε ευρετήρια πάνω στα αρχεία ευρετηρίου

- Έστω ότι το αρχείο ευρετηρίου είναι το *πρώτο ή βασικό* επίπεδο

Έστω ότι ο παράγοντας ομαδοποίησης είναι f_0 και ότι έχει r_1 blocks

Το αρχείο είναι διατεταγμένο και το πεδίο διάταξης είναι και κλειδί

- Δημιουργούμε ένα πρωτεύον ευρετήριο για το ευρετήριο πρώτου επιπέδου - *δεύτερο* επίπεδο

  Παράγοντας ομαδοποίησης: f_0 Αριθμός block $\lceil (r_1/f_0) \rceil$

  - Δημιουργούμε ένα πρωτεύον ευρετήριο για το ευρετήριο δεύτερου επιπέδου - *τρίτο* επίπεδο

    Παράγοντας ομαδοποίησης: f_0 Αριθμός block $\lceil (r_1/(f_0)^2) \rceil$

- Μέχρι πόσα επίπεδα:

Μέχρι όλες οι εγγραφές του ευρετηρίου να χωρούν σε ένα block

Έστω t κορυφαίο επίπεδο (top level) $\lceil (r_1/(f_0)^t) \rceil = 1$

- Το f_0 ονομάζεται και *παράγοντας διακλάδωσης* του ευρετηρίου

Ευρετήριο Πολλών Επιπέδων

Παράδειγμα: Έστω αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V = 9 bytes αλλά δεν είναι πεδίο διάταξης,. Κατασκευάζουμε δευτερεύον ευρετήριο, μέγεθος δείκτη block P = 6 bytes

$$f_0 = \lfloor (1024 / (9 + 6)) \rfloor = 68$$

Μέγεθος αρχείου δεδομένων: 3.000 blocks

Μέγεθος αρχείου ευρετηρίου *πρώτου* επιπέδου: 442 blocks

Μέγεθος αρχείου ευρετηρίου *δεύτερου* επιπέδου: $\lceil (442 / 68) \rceil = 7$ blocks

Μέγεθος αρχείου ευρετηρίου *τρίτου* επιπέδου: $\lceil (7 / 68) \rceil = 1$ block

Άρα t = 3

Ευρετήριο Πολλών Επιπέδων

- **Αναζήτηση**

p := διεύθυνση του block του κορυφαίου επιπέδου του ευρετηρίου

t := αριθμός επιπέδων του ευρετηρίου

for j = t to 1 step -1 do

read block με διεύθυνση p του ευρετηρίου στο επίπεδο j

αναζήτηση στο block p της εγγραφής i με τιμή $K_j(i) \leq K < K_j(i+1)$

read το block του αρχείου δεδομένων με διεύθυνση p

Αναζήτηση στο block p της εγγραφής i με τιμή $K_j(i) \leq K < K_j(i+1)$

Ευρετήριο Πολλών Επιπέδων

Παράδειγμα: Έστω αρχείο με r = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V = 9 bytes αλλά δεν είναι πεδίο διάταξης,. Κατασκευάζουμε δευτερεύον ευρετήριο, μέγεθος δείκτη block P = 6 bytes

Άρα t = 3

Παράδειγμα

t + 1 = 4 προσπελάσεις

Για το δευτερεύον ήταν 10 και χωρίς ευρετήριο 1500

Ευρετήριο Πολλών Επιπέδων

- **Εισαγωγή/διαγραφή**

τροποποιήσεις πολλαπλών ευρετηρίων

Δυναμικό πολυεπίπεδο ευρετήριο: B-δέντρα και B+-δέντρα

Δενδρικά Ευρετήρια

Τα πολυεπίπεδα ευρετήρια μπορεί να θεωρηθούν ως δέντρα αναζήτησης

Κάθε κόμβος (block) έχει f_0 δείκτες και f_0 τιμές κλειδιού

Ευρετήρια

- Ένα ευρετήριο (index) είναι μια βοηθητική δομή αρχείου που κάνει πιο αποδοτική την αναζήτηση μιας εγγραφής σε ένα αρχείο
- Το ευρετήριο καθορίζεται (συνήθως) σε ένα γνώρισμα του αρχείου
- Συχνά αποκαλείται access path (μονοπάτι πρόσβασης) στο γνώρισμα
- Μια καταχώρηση / εγγραφή στο Ευρετήριο έχει την μορφή:

Τιμή Κλειδιού	Δείκτης στο block της εγγραφής

- Η δομή ευρετηρίου (αρχείο) καταλαμβάνει μικρότερο χώρο από το ίδιο το αρχείο (οι καταχωρήσεις είναι μικρότερες και λιγότερες)
- Κάνοντας Δυαδική αναζήτηση στο Ευρετήριο βρίσκουμε τον Δείκτη στο block όπου αποθηκεύεται η εγγραφή που θέλουμε

Είδη Ευρετηρίων

- **Πυκνό Ευρετήριο** μια καταχώρηση για κάθε εγγραφή του δίσκου
- **Μη Πυκνό Ευρετήριο**

- **Ευρετήριο συστάδων** (clustered index) στο πεδίο διάταξης το οποίο όμως δεν είναι κλειδί - γενικότερα, όταν η διάταξη των καταχωρήσεων στο ευρετήριο ακολουθεί τη διάταξη των εγγραφών στο αρχείο
- **Ευρετήριο χωρίς συστάδες** (unclustered index)
- **Πρωτεύον**
- **Δευτερεύον (πλήρως αντεστραμμένο ευρετήριο)**

Ευρετήρια

Τα Ευρετήρια (ενός επιπέδου) χωρίζονται σε:

- **Πρωτεύον Ευρετήριο:** ορίζεται σε ένα αρχείο που είναι **διατεταγμένο στο (κύριο) κλειδί**. Περιλαμβάνει μια καταχώρηση για κάθε block. Η καταχώρηση έχει την τιμή του κλειδιού της πρώτης εγγραφής στο block. (συχνά ονομάζεται, μη-πυκνό ευρετήριο --- sparse index ή non-dense index)

- **Ευρετήριο Συστάδων (Clustering Index):** ορίζεται σε ένα αρχείο που είναι **διατεταγμένο σε γνώρισμα που δεν είναι κλειδί**. Περιλαμβάνει μια καταχώρηση για κάθε ξεχωριστή τιμή του γνωρίσματος. Η καταχώρηση ¨δείχνει¨ το πρώτο block που περιέχει εγγραφές με αυτή την τιμή γνωρίσματος

Ευρετήρια

- **Δευτερεύον Ευρετήριο (Secondary Index):** ορίζεται σε ένα αρχείο που είναι **μη – διατεταγμένο** στο γνώρισμα. Περιλαμβάνει μια καταχώρηση για κάθε Εγγραφή (συχνά ονομάζεται, πυκνό ευρετήριο -dense index)

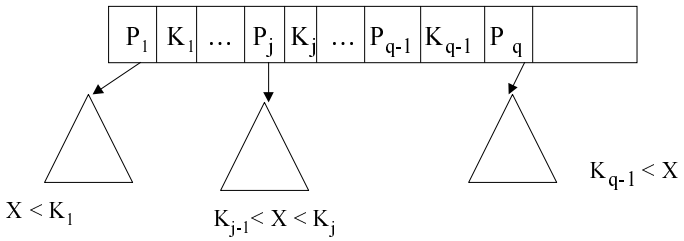
- Τα Αρχεία Ευρετηρίων είναι απλά Αρχεία, άρα και σε αυτά μπορούν να οριστούν Ευρετήρια
- Καταλήγουμε λοιπόν σε μια ιεραρχία δομών ευρετηρίων (πρώτο επίπεδο, δεύτερο επίπεδο, κλπ.)
- Κάθε επίπεδο του ευρετηρίου είναι ένα διατεταγμένο αρχείο, συνεπώς, Εισαγωγές / Διαγραφές εγγραφών απαιτούν επιπλέον δουλειά
- Ένα πολύ-επίπεδο ευρετήριο αποτελεί ένα Δέντρο Αναζήτησης

Είδη Ευρετηρίων

- **Ευρετήριο ενός επιπέδου** ένα διατεταγμένο αρχείο με εγγραφές (<K(i), P(i)>
- **Ευρετήριο πολλών επιπέδων**
- **Ευρετήρια δομής δέντρου**
- **Ευρετήρια κατακερματισμού**

Δέντρα Αναζήτησης

Ένα **δέντρο αναζήτησης** (search tree) τάξεως p είναι ένα δέντρο τέτοιο ώστε κάθε κόμβος του περιέχει το πολύ p - 1 τιμές αναζήτησης και ρ δείκτες ως εξής

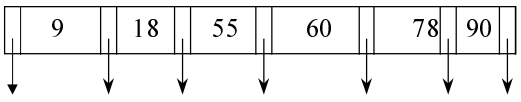


Υποθέτουμε ότι οι τιμές αναζήτησης είναι μοναδικές

$K_1 < K_2 < \ldots K_{q-1}$ και για όλες τις τιμές X στα υποδέντρα ισχύει
$K_{j-1} < X < K_j$ για $1 < j < q$, $X < K_j$ για $j = 1$, και $K_{j-1} < X$ για $i = q$

Δέντρα Αναζήτησης

Οι κόμβοι του Δέντρου κρατούν **< τιμή κλειδιού, δείκτης >** ζεύγη, ταξινομημένα στην **τιμή κλειδιού**. Οι εσωτερικοί κόμβοι ¨δείχνουν¨ σε χαμηλότερου επιπέδου κόμβους ενώ τα φύλλα-κόμβοι ¨δείχνουν¨ σε Μπλοκ του αρχείου (Σχέσης). Ένας Δείκτης ¨δείχνει¨ ένα υπό-δέντρο με τιμές κλειδιού ΜΕΓΑΛΥΤΕΡΕΣ ή ΙΣΕΣ της αντίστοιχης τιμής κλειδιού και ΜΙΚΡΟΤΕΡΕΣ της τιμής Κλειδιού του επόμενου Δείκτη



Δέντρα Αναζήτησης

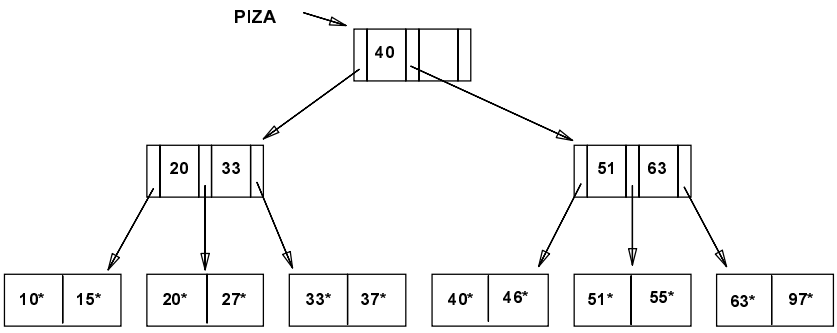
Για άμεση πρόσβαση σε εγγραφές αρχείου διατεταγμένου στο κλειδί

Κάθε κόμβος του Δέντρου είναι ένα block στο Δίσκο.

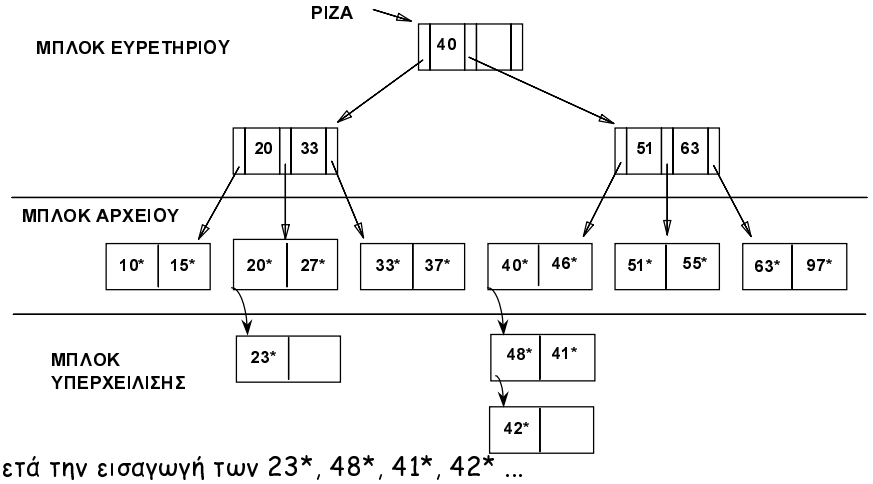
Ισοζυγισμένο: όλοι οι κόμβοι-φύλλα στο ίδιο επίπεδο

Β-δέντρο: ένα δέντρο αναζήτησης που παραμένει ισοζυγισμένο

Δέντρα Αναζήτησης



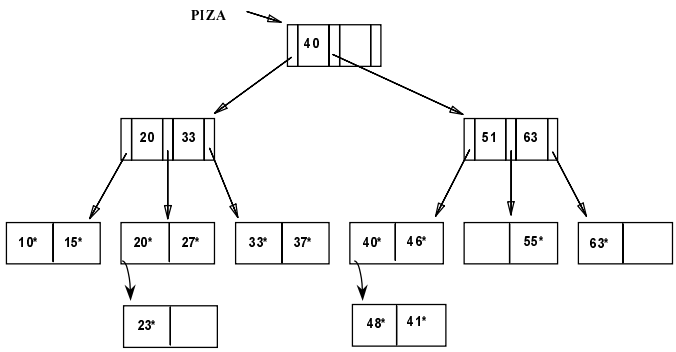
Δέντρα Αναζήτησης (ISAM)



Μετά την εισαγωγή των 23*, 48*, 41*, 42* ...

Δέντρα Αναζήτησης (ISAM)

Μετά την διαγραφή των 42*, 51*, 97*



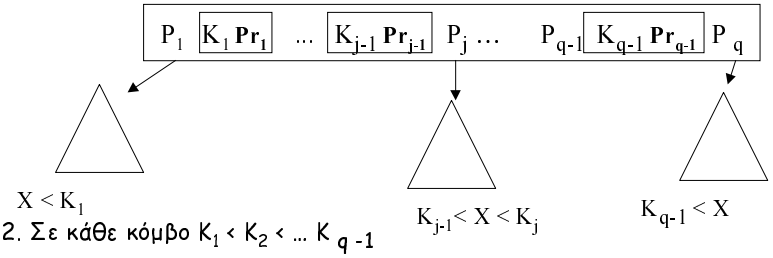
Παρατηρείστε ότι το 51* είναι στο ευρετήριο, αλλά όχι στο Αρχείο!

B-δέντρα

Ένα **B-δέντρο τάξεως (order) p** ορίζεται ως εξής:

1. Κάθε **εσωτερικός κόμβος** είναι της μορφής

<P_1, <K_1, Pr_1>, P_2, <K_2, Pr_2>, … <K_{q-1}, Pr_{q-1}>, P_q>, q < p, όπου P_i δείκτης δέντρου, K_i τιμή αναζήτησης, Pr_i δείκτης δεδομένων



2. Σε κάθε κόμβο $K_1 < K_2 < \ldots K_{q-1}$

3. Για όλες τις τιμές X στο υποδέντρο που δείχνει το P_j ισχύει $K_{j-1} < X < K_j$ για $1 < j < q$, $X < K_j$ για $j = 1$, και $K_{j-1} < X$ για $i = q$

B-δέντρα

4. Κάθε κόμβος έχει το **πολύ** p δείκτες δέντρου

5. Κάθε κόμβος εκτός της ρίζα και των φύλλων έχει **τουλάχιστον** $\lceil (p/2) \rceil$. Η ρίζα έχει τουλάχιστον 2 εκτός αν είναι ο μόνος κόμβος του δέντρου.

6. Ένας κόμβος με q δείκτες δέντρου περιέχει q - 1 τιμές πεδίου αναζήτησης (και άρα και q - 1 δείκτες δεδομένων)

7. Όλα τα φύλλα βρίσκονται στο ίδιο επίπεδο. Τα φύλλα έχουν την ίδια δομή εκτός του ότι οι δείκτες δέντρου είναι null.

B-δέντρα

- Τάξη p ώστε κάθε κόμβος να χωρά σε ένα block

Έστω B μέγεθος block, V μέγεθος πεδίου αναζήτησης, Pr μέγεθος δείκτη δεδομένων (εγγραφής) και P μέγεθος δείκτη δέντρου (block)

$p * P + (p - 1) * Pr + (p - 1) * V \leq B$

$p * (P + Pr + V) \leq B + V + Pr$

$p \leq (B + V + Pr) / (P + Pr + V)$

Παράδειγμα, V = 9 bytes, B = 512, Pr = 7 bytes, P = 6 bytes,

τότε p = 23

B-δέντρα

- Υπολογισμός επιπέδων

Παράδειγμα, V = 9 bytes, B = 512, Pr = 7 bytes, P = 6 bytes, τότε p = 23. έστω ότι κάθε κόμβος είναι γεμάτος κατά 69% και έστω ότι το αρχείο δεδομένων έχει 65.000 εγγραφές

Ρίζα	1 κόμβος	15 (23*0,69) καταχωρήσεις	16 δείκτες
Επίπεδο 1:	16 κόμβοι	240 (16*15) καταχωρήσεις	256 δείκτες
Επίπεδο 2:	256 κόμβοι	3.840 (256*15) καταχωρήσεις	4.096 δείκτες
Επίπεδο 3:	4.096 κόμβοι	61.440	

Σύνολο: 61.440 + 3.840 + 240 + 15

B+-δέντρα

Αποθηκεύουμε δείκτες δεδομένων μόνο στα φύλλα

Δύο τύποι κόμβων:

- εσωτερικοί κόμβοι
- φύλλα

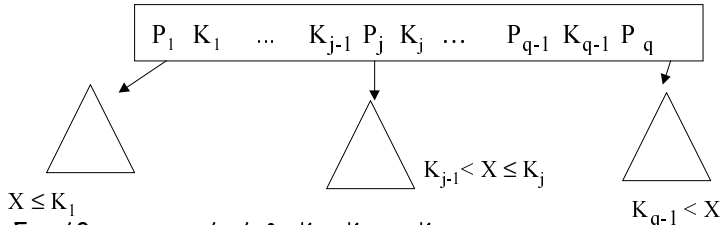
Κάποιες τιμές του πεδίου αναζήτησης μπορεί να εμφανίζονται παραπάνω από μια φορά

B+-δέντρα

Ένα **B+-δέντρο** τάξεως (order) p ορίζεται ως εξής:

1. Κάθε **εσωτερικός κόμβος** είναι της μορφής

<P_1, K_1, P_2, K_2, , … K_{q-1}, P_{q-1}, P_q> q ≤ p, όπου P_i δείκτης δέντρου, K_i τιμή αναζήτησης



2. Σε κάθε εσωτερικό κόμβο $K_1 < K_2 < … K_{q-1}$

3. Για όλες τις τιμές X στο υποδέντρο που δείχνει το P_j ισχύει $K_{j-1} < X \leq K_j$ για $1 < j < q$, $X \leq K_j$ για j =1, και $K_{j-1} < X$ για i = q

B⁺-δέντρα

4. Κάθε εσωτερικός κόμβος έχει το **πολύ** p δείκτες δέντρου

5. Κάθε εσωτερικός κόμβος εκτός της ρίζα έχει **τουλάχιστον** $\lceil (p/2) \rceil$. Η ρίζα έχει τουλάχιστον 2 εκτός αν είναι ο μόνος κόμβος του δέντρου.

6. Ένας κόμβος με q δείκτες δέντρου περιέχει q - 1 τιμές πεδίου αναζήτησης

B⁺-δέντρα

1. Κάθε **κόμβος-φύλλο** είναι της μορφής

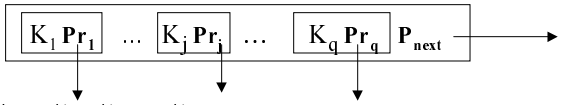
$<<K_1, Pr_1>, <K_2, Pr_2>, \ldots <K_q, Pr_q>, P_{next}>$, $q \leq p_{leaf}$, όπου

p_{leaf} είναι η τάξη των κόμβων-φύλλων

K_i τιμή αναζήτησης,

Pr_i δείκτης δεδομένων που δείχνει στο block (ή στην εγγραφή) με τιμή στο πεδίο αναζήτησης K_i (ή σε ένα block ενδιάμεσου επιπέδου αν το πεδίο αναζήτησης δεν είναι κλειδί),

P_{next} δείχνει στο επόμενο φύλλο και χρησιμοποιείται για τη γρήγορη ανάγνωση του αρχείου σε διάταξη



2. Σε κάθε κόμβο-φύλλο $K_1 < K_2 < \ldots K_q$

B+-δέντρα

3. Κάθε κόμβος-φύλλο έχει το **πολύ** p_{leaf} τιμές

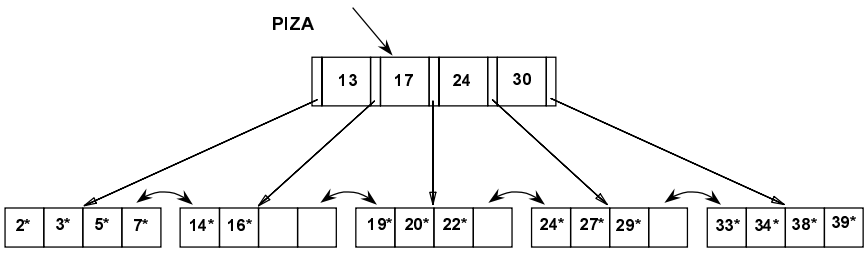
4. Κάθε κόμβος-φύλλο έχει **τουλάχιστον** $\lceil (p_{leaf} /2) \rceil$ τιμές.

5. Όλοι οι κόμβοι-φύλλα βρίσκονται στο ίδιο επίπεδο.

B+-δέντρα

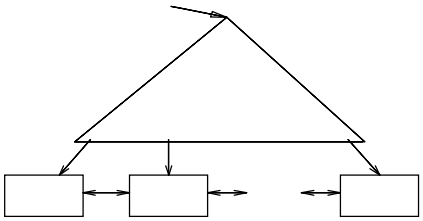
Η αναζήτηση ξεκινά από τη ρίζα, και οι συγκρίσεις των κλειδιών μας οδηγούν στα φύλλα

Αναζήτηση για τα 5*, 15*, όλες οι καταχωρήσεις >= 24* ...



B⁺-δέντρα

- Εισαγωγή / Διαγραφή με κόστος $\log_F N$ --- κρατούν το Δέντρο σε ισορροπημένη μορφή. (F = εξάπλωση, N = αριθμός των φύλλων)
- Ελάχιστη πληρότητα 50% (εκτός της ρίζας). Κάθε κόμβος περιέχει $p <= \underline{m} <= 2p$ καταχωρήσεις. Το p ονομάζεται τάξη του δέντρου.
- Εξαιρετική δομή ΚΑΙ για exact queries ΚΑΙ για range queries.



Καταχωρήσεις Ευρετηρίου
(Άμεση Αναζήτηση)

Καταχωρήσεις Δεδομένων
(«Σύνολο ακολουθίας»)

B⁺-δέντρα

- *Τάξη p ώστε κάθε εσωτερικός*-κόμβος *να χωρά σε ένα block*

Έστω B μέγεθος block, V μέγεθος πεδίου αναζήτησης, Pr μέγεθος δείκτη δεδομένων (εγγραφής) και P μέγεθος δείκτη δέντρου (block)

$p * P + (p - 1) * V \leq B$

$p * (P + V) \leq B + V$

$p \leq (B + V) / (P + V)$

Παράδειγμα, V = 9 bytes, B = 512, Pr = 7 bytes, P = 6 bytes, τότε
p = 34

Για B-δέντρο, p = 23

B+-δέντρα

- Τάξη p_{leaf} ώστε κάθε φύλλο να χωρά σε ένα *block*

Έστω B μέγεθος block, V μέγεθος πεδίου αναζήτησης, Pr μέγεθος δείκτη δεδομένων (εγγραφής) και P μέγεθος δείκτη δέντρου (block)

$p_{leaf} * (Pr + V) + P \leq B$

$p_{leaf} * (Pr + V) \leq B - P$

$p_{leaf} \leq (B - P) / (Pr + V)$

Παράδειγμα, V = 9 bytes, B = 512, Pr = 7 bytes, P = 6 bytes, τότε $p_{leaf} = 31$

B+-δέντρα

- Υπολογισμός επιπέδων

Παράδειγμα, V = 9 bytes, B = 512, Pr = 7 bytes, P = 6 bytes, τότε p = 34. έστω ότι κάθε κόμβος είναι γεμάτος κατά 69% και έστω ότι το αρχείο δεδομένων έχει 65.000 εγγραφές

Ρίζα	1 κόμβος	22 (34*0,69) καταχωρήσεις	23 δείκτες
Επίπεδο 1:	23 κόμβοι	506 (23*22) καταχωρήσεις	529 δείκτες
Επίπεδο 2:	529κόμβοι	11.638 (529*22) καταχωρήσεις	12.167 δείκτες

Επίπεδο φύλλων: 12.167 κόμβοι 255.507 (12.167 * 31 * 0.69) δείκτες δεδομένων

Σε 3 επίπεδα 255.507 εγγραφές έναντι 65.535 για B-δέντρο

B+-δέντρα

Τυπική Τάξη: 100. Τυπικός Παράγων Πληρότητας: 67%.
Μέση τιμή εξάπλωσης (fan out) = 133
Τυπικές Δυνατότητες:
Ύψος 4: 133^4 = 312,900,700 εγγραφές
Ύψος 3: 133^3 = 2,352,637 εγγραφές
Δύναται να κρατά τα υψηλότερα επίπεδα στον buffer :
Επίπεδο 1 = 1 Μπλοκ = 8 Kbytes
Επίπεδο 2 = 133 Μπλοκ = 1 Mbyte
Επίπεδο 3 = 17,689 Μπλοκ = 133 MBytes

B+-δέντρα: Αναζήτηση

Αναζήτηση (αναδρομική εκδοχή)

```
nodepointer find(keyvalue K):
        return tree_search(root, K);
end;
```

B⁺-δέντρα: Αναζήτηση

```
Nodepointer tree_search(nodepointer P, keyvalue K)
        if P is a leaf return(P);
        else
           if K ≤ K1 tree_search(P1, K)
             else
                 find i such that Ki < K ≤ Ki+1
                  return tree_search(Pi+1, K)
end
```


B⁺-δέντρα: Εισαγωγή

Εισαγωγή

1. Αναζήτηση του φύλλου για εισαγωγή: έστω φύλλο P

2. Εισαγωγή τιμής K στο κόμβο P

   Αν ο κόμβος-φύλλο δεν είναι γεμάτος

   εισαγωγή της τιμής

B^+-δέντρα: Εισαγωγή

Αν ο κόμβος-φύλλο είναι γεμάτος (έχει p_{leaf} εγγραφές)

διάσπαση του κόμβου:

-- οι πρώτες $k = \lceil((p_{leaf}+1)/2)\rceil$ παραμένουν στον κόμβο

-- οι υπόλοιπες σε καινούργιο κόμβο

-- εισαγωγή (αντιγραφή) της k-οστής τιμής (K_k) στον πατέρα

B^+-δέντρα: Εισαγωγή

Αν ο εσωτερικός κόμβος-φύλλο είναι γεμάτος (έχει p εγγραφές)

διάσπαση του κόμβου: έστω $k = \lfloor ((p+1)/2) \rfloor$

-- οι εγγραφές μέχρι το P_k (μετά την εισαγωγή παραμένουν στον κόμβο

-- η k-οστή K_k τιμή **μεταφέρεται (*δεν αντιγράφεται*)** στον πατέρα

-- οι υπόλοιπες σε καινούργιο κόμβο

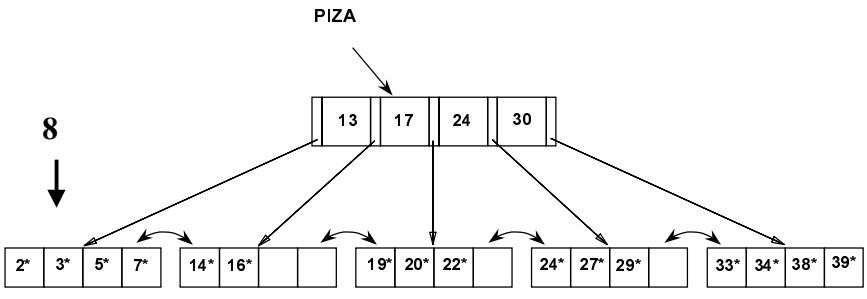
B+-δέντρα: Εισαγωγή

Οι διασπάσεις κόμβων "μεγαλώνουν" το δέντρο

Η διάσπαση της ρίζας " υψώνει " το δέντρο

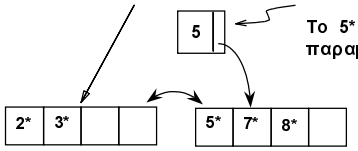
B+-δέντρα: Εισαγωγή

Εισαγωγή της καταχώρησης 8*



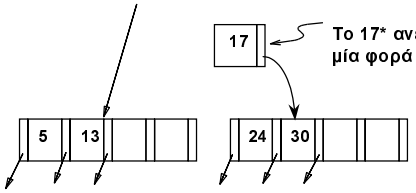
B+-δέντρα: Εισαγωγή

Καταχώρηση στον πατέρα κόμβο.



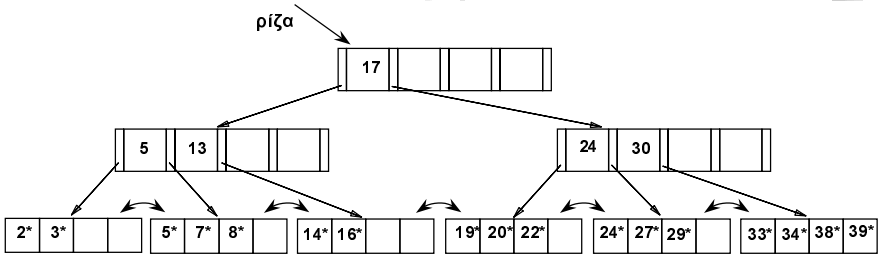
Το 5* ανεβαίνει επάνω, αλλά παραμένει και στο φύλλο

Καταχώρηση στον πατέρα κόμβο



Το 17* ανεβαίνει επάνω και παρουσιάζεται μόνο μία φορά στο ευρετήριο (σε αντίθεση με Φύλλα)

B+-δέντρα: Εισαγωγή



Η ρίζα διασπάστηκε οδηγώντας σε αύξηση του ύψους.

B^+-δέντρα

Όλες οι τιμές εμφανίζονται στα φύλλα και κάποιες επαναλαμβάνονται και σε εσωτερικούς κόμβους (η τιμή K σε ένα εσωτερικό κόμβο εμφανίζεται επίσης ως η δεξιότερη τιμή στο φύλλο του υποδέντρου με ρίζα το δείκτη στα αριστερά του K)

B^+-δέντρα: Διαγραφή

Διαγραφή

1. Αναζήτηση του φύλλου που περιέχει το K: έστω φύλλο P

*** αν η τιμή K εμφανίζεται σε κάποιο εσωτερικό κόμβο πρέπει να αντικατασταθεί από την τιμή που βρίσκεται στα αριστερά του K στο φύλλο

2. Αν υποχείλιση

αν είναι δυνατόν ανακατανομή με τον αριστερό αδελφό ($> \lceil (n/2) \rceil$)

αν όχι, προσπάθεια ανακατανομής με το δεξιό αδελφό

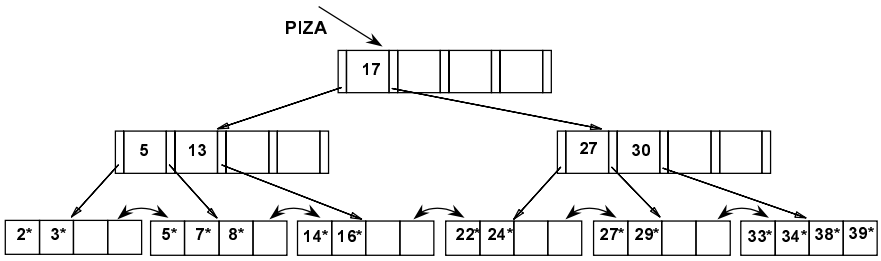
αν όχι, συγχώνευση και των τριών κόμβων σε δύο κόμβους

B+-δέντρα: Διαγραφή

Αν έγινε συγχώνευση, πρέπει να διαγραφεί η καταχώρηση (που δείχνει στο P ή τον αδελφό) από τον πατέρα του P.

Η συγχώνευση μπορεί να φτάσει στη ρίζα, μειώνοντας το ύψος του δέντρου.

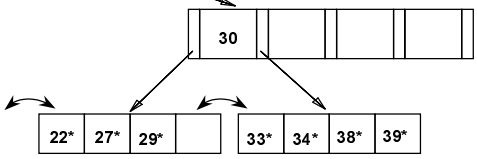
B+-δέντρα: Διαγραφή



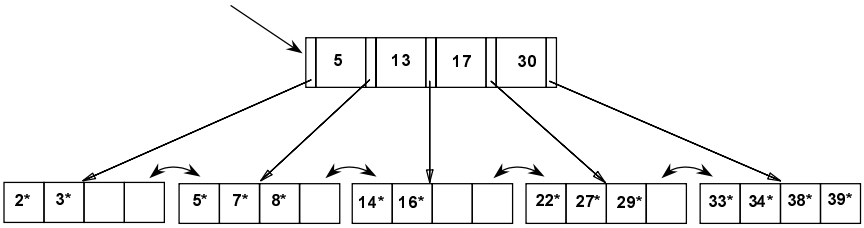
Το παράδειγμα μετά τη διαγραφή του 19* και του 20*

B+-δέντρα: Διαγραφή

Τέλος, η διαγραφή του 24*



PIZA



Ευρετήρια Κατακερματισμού

Εξωτερικός Κατακερματισμός

Κάδος: μια συστάδα από συνεχόμενα blocks του αρχείου

h(k) = i ← Σχετική διεύθυνση του κάδου (ποιος κάδος του αρχείου)

Τιμή του πεδίου κατακερματισμού

π.χ., η εγγραφή με τιμή k στο πεδίο κατακερματισμού βρίσκεται στον i-οστό κάδο

Ένας πίνακας που αποθηκεύεται στην επικεφαλίδα του αρχείου μετατρέπει τον αριθμό κάδου στην αντίστοιχη διεύθυνση block

0	διεύθυνση 1ου block του κάδου στο δίσκο
1	διεύθυνση 1ου block του κάδου στο δίσκο
2	διεύθυνση 1ου block του κάδου στο δίσκο
…	…
M-1	διεύθυνση 1ου block του κάδου στο δίσκο

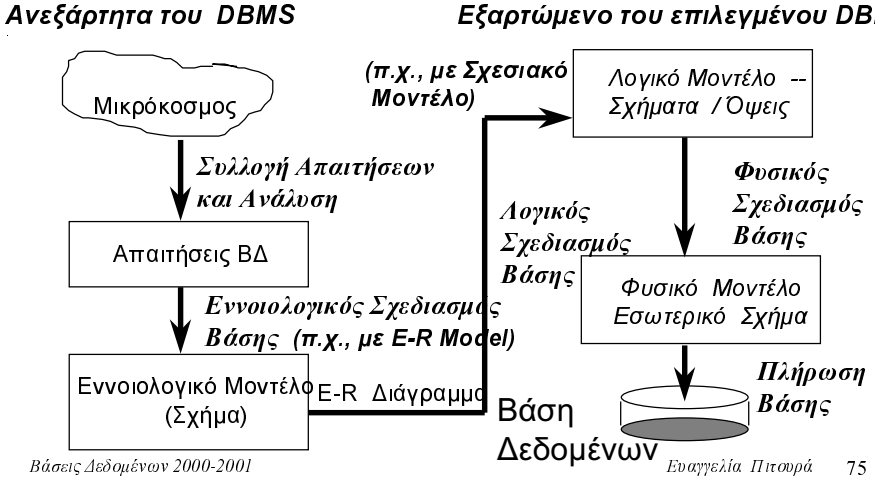
Δυναμικός Εξωτερικός Κατακερματισμός

Πρόβλημα:

Έστω M κάδους και r εγγραφές ανά κάδο - το πολύ M * r εγγραφές

- **Δυαδική αναπαράσταση του αποτελέσματος** της συνάρτησης κατακερματισμού, δηλαδή ως μια ακολουθία δυαδικών ψηφίων
  - Κατανομή εγγραφών με βάση την τιμή των αρχικών ψηφίων

Φυσικός Σχεδιασμός



Φυσικός Σχεδιασμός

- Μετά τον ER σχεδιασμό και τον λογικό σχεδιασμό (σχεσιακό μοντέλο), έχουμε τα εννοιολογικό και λογικό (με τις όψεις) σχήματα για τη Βάση Δεδομένων.
- Το επόμενο βήμα είναι ο **Φυσικός Σχεδιασμός**, δηλαδή η επιλογή των δομών αποθήκευσης των σχέσεων, η επιλογή των ευρετηρίων, οι αποφάσεις για συστάδες - γενικά ότι είναι απαραίτητο για να επιτευχθούν οι προσδοκώμενες επιδόσεις χρήσης της ΒΔ.
- Η υλοποίηση μιας (φυσικής) Σχεσιακής Βάσης Δεδομένων περιλαμβάνει τη δημιουργία ΚΑΤΑΛΟΓΩΝ ΣΥΣΤΗΜΑΤΟΣ (directory system tables)

Κατάλογος Συστήματος

- Για κάθε σχέση (Relation):
  - Όνομα, Όνομα Αρχείου, Δομή Αρχείου (π.χ., Αρχείο Σωρού)
  - Όνομα Γνωρίσματος και Τύπος, για κάθε Γνώρισμα
  - Όνομα Ευρετηρίου, για κάθε Ευρετήριο
  - Περιορισμοί Ακεραιότητας
- Για κάθε Ευρετήριο:
  - Δομή (π.χ. B+ δέντρο) και πεδία για αναζήτηση
- Για κάθε Όψη (view):
  - Όνομα Όψης και Ορισμός αυτής
- Επιπλέον, στατιστικά στοιχεία χρήσης, δικαιοδοσίες, μέγεθος ενδιάμεσης μνήμης, κλπ.

Οι κατάλογοι σε ένα σχεσιακό σύστημα αποθηκεύονται και οι ίδιοι σαν σχέσεις

Φυσικός Σχεδιασμός

Για να κάνουμε όσο το δυνατόν καλύτερο τον Φυσικό Σχεδιασμό, πρέπει να :

Κατανοήσουμε το **Φόρτο Εργασίας (workload)**

Ποια είναι τα πιο σημαντικά queries και πόσο συχνά εμφανίζονται.

Ποια είναι τα πιο σημαντικά updates και πόσο συχνά εμφανίζονται.

Ποια είναι η επιθυμητή επίδοση για την εκτέλεση αυτών των queries και updates.

Για κάθε ερώτηση (query) στο φόρτο εργασίας:

- Σε ποιες σχέσεις έχει πρόσβαση?
- Ποια γνωρίσματα ανακαλεί?
- Ποια γνωρίσματα υπεισέρχονται στις συνθήκες για selection/join? Πόσο επιλεκτικές είναι αυτές οι συνθήκες?

Για κάθε ενημέρωση (insert / delete/ update) στο workload:

- Ποια γνωρίσματα υπεισέρχονται στις συνθήκες για selection/join? Πόσο επιλεκτικές είναι αυτές οι συνθήκες?
- Ο τύπος της ενημέρωσης (INSERT/DELETE/UPDATE), και τα γνωρίσματα που θα επηρεασθούν

Αποφάσεις που Απαιτούνται

Τι ευρετήρια πρέπει να δημιουργηθούν;

- Ποιες σχέσεις πρέπει να έχουν ευρετήρια; Ποια γνωρίσματα χρησιμοποιούνται για αναζήτηση; Πρέπει να ορίσουμε πολλαπλά ευρετήρια;

Για κάθε ευρετήριο, τι είδους ευρετήριο πρέπει να είναι;

- Συστάδες; Δέντρο / Κατακερματισμός; Δυναμικό / Στατικό; Πυκνό / Μη-πυκνό;

Χρειάζονται αλλαγές και στο εννοιολογικό / λογικό Σχήμα;

- Διαφορετικό κανονικοποιημένο σχήμα;
- Denormalization (μήπως χρειάζεται από-κανονικοποίηση?)
- Όψεις, Επανάληψη Δεδομένων (replication) ...

Φυσικός Σχεδιασμός

Προσέγγιση: Θεώρησε τα πιο σημαντικά queries στη σειρά. Θεώρησε την καλύτερη εκτέλεση (σχέδιο) με τα υπάρχοντα ευρετήρια, και δες αν υπάρχει ακόμη καλύτερη εκτέλεση με ένα επιπλέον ευρετήριο. Αν είναι έτσι, δημιούργησέ το

Πριν δημιουργήσουμε ένα ευρετήριο, πρέπει να συνυπολογίσουμε και την επίδρασή του σε ενημερώσεις του φορτίου εργασίας!

Η εξισορρόπηση είναι ότι ένα ευρετήριο κάνει τις ερωτήσεις ΠΙΟ ΓΡΗΓΟΡΕΣ και τις ενημερώσεις ΠΙΟ ΑΡΓΕΣ

Επιπλέον, απαιτεί και χώρο στον Δίσκο

Φυσικός Σχεδιασμός

Γνωρίσματα στο WHERE clause είναι υποψήφια για ευρετηρίαση

Συνθήκη με ακριβές ταίριασμα (ισότητα) μας οδηγεί σε ευρετήριο κατακερματισμού (hash index.)

Ερωτήσεις διακύμανσης μας οδηγούν σε tree index.

Το ευρετήριο συστάδων είναι ιδιαίτερα αποδοτικό για τέτοιου είδους ερωτήσεις – ειδικά αν έχουμε πολλές ίσες τιμές.

Προσπαθούμε πάντα να επιλέξουμε ευρετήρια που εξυπηρετούν όσα το δυνατό περισσότερα queries.

Μια και μόνο μία συστάδα-ευρετήριο μπορεί να υπάρχει ανά σχέση, διάλεξε την ΜΟΝΟ αν υπάρχει σημαντικό query που επωφελείται.

Φυσικός Σχεδιασμός

- Ευρετήρια για πολλαπλά γνωρίσματα δημιουργούνται όταν η WHERE clause περιέχει πολλές συνθήκες
  - Αν υπάρχουν επιλογές διακύμανσης, πρέπει να προσεχθεί η σειρά των γνωρισμάτων
- Όταν υπάρχει συνθήκη συνένωσης:
  - Ανάλογα με τη μέθοδο υλοποίησης της συνένωσης που υποστηρίζεται από το Σύστημα, για παράδειγμα,
    - Το ευρετήριο μπορεί να είναι κατακερματισμού αν η μέθοδος υλοποίησης συνένωσης είναι nested loop
    - Clustered B+ δέντρο σε γνωρίσματα συνένωσης είναι καλά για Sort-Merge μέθοδο συνένωσης, κλπ.