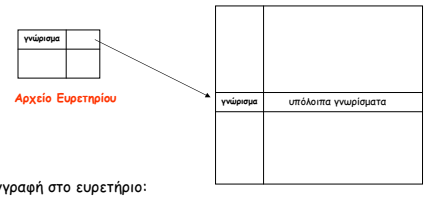
Ευρετήρια

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 που καλείται πεδίο ευρετηριοποίησης (indexing field)



Αρχείο Ευρετηρίου

Αρχείο Δεδομένων

Εγγραφή στο ευρετήριο:

Τιμή Πεδίου Ευρετηριοποίησης	Δείκτης στο **block** της εγγραφής

Ευρετήρια

Διαφορετικού τύπου εγγραφές ανάλογα με το πεδίο ευρετηριοποίησης:

(α) κλειδί ή όχι,

(β) πεδίο διάταξης ή όχι

Ευρετήρια

- **Πυκνό ευρετήριο**: μια καταχώρηση για κάθε εγγραφή του αρχείου
- **Μη πυκνό ευρετήριο**

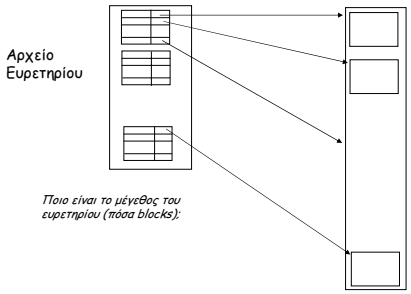
Πρωτεύον Ευρετήριο

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



Παράδειγμα (υπολογισμός μεγέθους αρχείου ευρετηρίου)

Έστω διατεταγμένο αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, όπου το πεδίο κλειδιού διάταξης έχει μέγεθος V_A = 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

- 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 αυτό του αρχείου δεδομένων.

- **Αναζήτηση**

Δυαδική αναζήτηση στο πρωτεύον ευρετήριο

Ανάγνωση του block από το αρχείο δεδομένων

Παράδειγμα (υπολογισμός κόστους αναζήτησης)

Δεδομένα όπως πριν

(Έστω διατεταγμένο αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, όπου το πεδίο κλειδιού διάταξης έχει μέγεθος V_A = 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

Δυαδική γιατί το αρχείο ταξινομημένο

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

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

- **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

Access paths (μονοπάτια προσπέλασης)

- Το αρχείο ευρετηρίου καταλαμβάνει *μικρότερο χώρο* από το ίδιο το αρχείο δεδομένων (οι καταχωρήσεις είναι μικρότερες και λιγότερες)
- Κάνοντας *δυαδική αναζήτηση* στο ευρετήριο (γιατί το ευρετήριο είναι διατεταγμένο αρχείο!) βρίσκουμε τον δείκτη στο block όπου αποθηκεύεται η εγγραφή που θέλουμε

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

Παράδειγμα (υπολογισμός μεγέθους ευρετηρίου)

Έστω διατεταγμένο αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο διάταξης έχει μέγεθος V_A = 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

- **Αναζήτηση**

Δυαδική αναζήτηση στο ευρετήριο

Ανάγνωση blocks από το αρχείο δεδομένων

Παράδειγμα (υπολογισμός κόστους αναζήτησης)

(στοιχεία όπως πριν) Έστω διατεταγμένο αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο διάταξης έχει μέγεθος V_A = 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 (secondary index): ορισμένο σε πεδίο *διαφορετικό του κλειδιού διάταξης*

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

Υπάρχει μια εγγραφή για κάθε εγγραφή του αρχείου που περιέχει:

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

Παράδειγμα (υπολογισμός μεγέθους ευρετηρίου)

Έστω αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V_A = 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

- **Αναζήτηση**

Δυαδική αναζήτηση στο δευτερεύον ευρετήριο

Ανάγνωση του block από το αρχείο δεδομένων

Παράδειγμα (υπολογισμός κόστους αναζήτησης)

Στοιχεία όπως πριν

(Έστω αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V_A = 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 (σειριακή αναζήτηση, γιατί το αρχείο δεδομένων δεν είναι ταξινομημένο): 3.000/2 = 1500 blocks

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

Για πρωτεύον ήταν 45 και 7 blocks αντίστοιχα

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

Παράδειγμα (υπολογισμός μεγέθους ευρετηρίου)

Έστω μη διατεταγμένο αρχείο (αρχείο σωρού) με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο ευρετηριοποίησης (δηλαδή, το πεδίο στο οποίο θα κατασκευάσουμε το ευρετήριο) έχει μέγεθος V_A = 9 bytes. Υπάρχουν 100 διαφορετικές τιμές και οι εγγραφές είναι ομοιόμορφα κατανεμημένες ως προς τις τιμές αυτές. Κατασκευάζουμε ευρετήριο συστάδων χρησιμοποιώντας την επιλογή (3), μέγεθος δείκτη block P = 6 bytes

Ευρετήριο bfr_{ET} = 68, b_{ET} = 2

Ενδιάμεσο επίπεδο $bfr_{EΔ}$ = ? -- Ποια είναι η οργάνωση του;

- **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**

Απλή αν δεν αφορά εισαγωγή νέας τιμής στο ευρετήριο

• Εύκολη η λογική διάταξη των εγγραφών με βάση το πεδίο ευρετηριοποίησης

• Ανακτήσεις με *σύνθετες συνθήκες,* μπορεί να γίνουν χρησιμοποιώντας τα blocks του ευρετηρίου

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).

Ευρετήρια (επανάληψη)

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 (επανάληψη)

- **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)

Πόσα ευρετήρια σε ένα αρχείο δεδομένων μπορεί να έχουμε;

Ευρετήριο Πολλών Επιπέδων

Ιδέα:

Τα ευρετήρια είναι αρχεία - 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 (άρα πρωτεύον ευρετήριο!)

Υπενθύμιση *(παράγοντας ομαδοποίησης: αριθμός εγγραφών ανά block*

Παράγοντας ομαδοποίησης (blocking factor), όταν B ≥ R

bfr = ⌊ (B / R) ⌋, όπου B μέγεθος block σε byte και R μέγεθος εγγραφής σε bytes

Ευρετήριο Πολλών Επιπέδων

• 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

Παράδειγμα (υπολογισμός μεγέθους ευρετηρίου)

Έστω αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V_A = 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

- **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 $K_j(i) \le K < K_j(i+1)$
  read το block του αρχείου δεδομένων με διεύθυνση p
  Αναζήτηση στο block p της εγγραφής i με τιμή $K_j(i) \le K < K_j(i+1)$

Παράδειγμα (υπολογισμός κόστους αναζήτησης)

Έστω αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, σταθερού μεγέθους εγγραφές μεγέθους R_A = 100 bytes, μη εκτεινόμενη καταχώρηση, όπου το πεδίο κλειδιού έχει μέγεθος V_A = 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

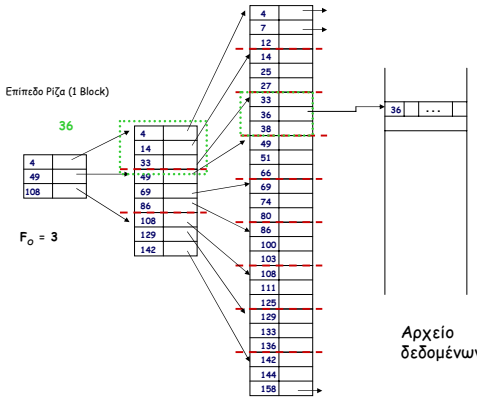
- **Εισαγωγή/διαγραφή**

  τροποποιήσεις πολλαπλών ευρετηρίων

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

Τα πολυεπίπεδα ευρετήρια μπορεί να θεωρηθούν ως δέντρα αναζήτησης

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

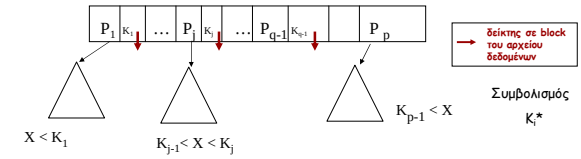


Ευρετήρια (επανάληψη)

- 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*

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 p 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

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

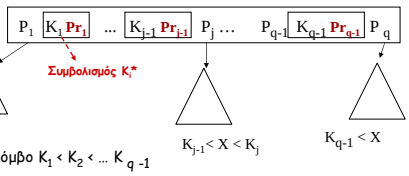
Β-δέντρο: ένα δέντρο αναζήτησης που παραμένει ισοζυγισμένο και χωρίς «πολύ αδειανούς» κόμβους

Β-δέντρα

Ένα **Β-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



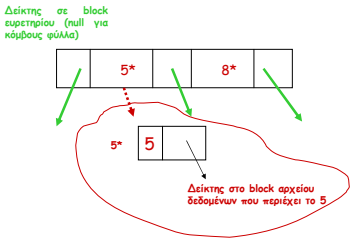
2. Σε κάθε κόμβο $K_1 < K_2 < \ldots K_{q-1}$
3. Για όλες τις τιμές X στο υποδέντρο που δείχνει το P_j ισχύει $K_{j-1} < X < K_j$ για $1 < j < q$, $X < K_j$ για $j = 1$, και $K_{j-1} < X$ για $i = q$

Β-δέντρα

4. Κάθε κόμβος έχει το **πολύ p δείκτες** δέντρου
5. Κάθε κόμβος *εκτός της ρίζα* και των *φύλλων* έχει **τουλάχιστον** $\lceil (p/2) \rceil$ δείκτες δέντρου. 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.

Β-δέντρα (παράδειγμα)

5, 8, 7, 14, 19, 6, 10 και τάξη p = 3 (2 τιμές ανά κόμβο, 3 δείκτες block ευρετηρίου)



Β-δέντρα

Εισαγωγή τιμής

Αρχικά ένας μόνο κόμβος (ρίζα) στο Επίπεδο 0

Όταν ο *κόμβος ρίζα* γεμίσει (p – 1 τιμές κλειδιού), νέα εισαγωγή οδηγεί στην διάσπαση του κόμβου σε δύο κόμβους στο Επίπεδο 1: η **μεσαία τιμή μένει** στη ρίζα, οι υπόλοιπες μοιράζονται εξίσου σε δύο κόμβους του Επιπέδου 1

Όταν *ένας κόμβος εκτός της ρίζας* γεμίσει, νέα εισαγωγή οδηγεί σε διάσπαση του κόμβου σε δύο κόμβους στο ίδιο επίπεδο και **μεταφορά της μεσαίας τιμής** στον γονέα του κόμβου

ΠΡΟΣΟΧΗ: η εισαγωγή της μεσαίας τιμής στο γονέα αν ο γονέας είναι γεμάτος μπορεί να οδηγήσει σε διάσπαση του γονέα. Η διάσπαση μπορεί να οδηγήσει ως τη ρίζα, οπότε δημιουργείται και νέο επίπεδο.

Β-δέντρα

Διαγραφή τιμής

Μια «προς διαγραφή» τιμή μπορεί να ανήκει σε εσωτερικό κόμβο

Αν σβήσουμε το K_i, το μικρότερο κλειδί του υποδέντρου P_{i+1} πρέπει να το *αντικαταστήσει (δηλαδή το μικρότερο κλειδί του κόμβου στα δεξιά του κλειδιού που διαγράφεται)*

Ειδικά για τα φύλλα, αν λόγω διαγραφής κάποιος κόμβος παραμείνει γεμάτος λιγότερο από το μισό: ενώνεται με τους γειτονικούς του κόμβους – αυτή η ένωση μπορεί να διαδοθεί ως τη ρίζα οπότε και οδηγεί σε μείωση των επιπέδων.

Πειραματικά: τυχαίες εισαγωγές και διαγραφές οδηγούν σε 69% πληρότητα

Β-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)

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

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

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

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

τότε p = 23

Β-δέντρα

Υπολογισμός επιπέδων

Έστω όπως πριν, p = 23. Έστω ότι κάθε κόμβος είναι γεμάτος κατά 69%. Πόσα επίπεδα χρειαζόμαστε για να ευρετηριοποιήσουμε 65.000 τιμές;

(p - 1)*0,69 = 22*0,69 = 15 κλειδιά και 15 + 1 = 16 δείκτες ανά κόμβο

	#κόμβων	#τιμές	#δείκτες
Ρίζα	1 κόμβος	15 (22*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 (65.535)

Β-δέντρα

Αναζήτηση

Διαβάζουμε το block της ρίζας

Αν η εγγραφή δεν υπάρχει στο κόμβο διαβάζουμε το αντίστοιχο block στο επόμενο πεδίο

Τα Β-δέντρα τροποποιούνται αντίστοιχα (πως;) αν το πεδίο δεν είναι κλειδί ή πεδίο ταξινόμησης (π.χ, με χρήση ενδιάμεσου επιπέδου)

Διαφορά B+ από B-δέντρο: Αποθηκεύουμε δείκτες δεδομένων (στο αρχείο δεδομένων) μόνο στα φύλλα

Δύο τύποι κόμβων:

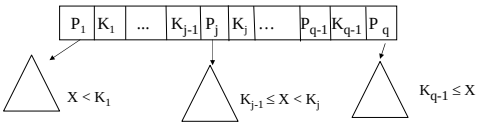
- εσωτερικοί κόμβοι
- φύλλα

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

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> $q \le p$, όπου P_i δείκτης δέντρου, K_i τιμή αναζήτησης



2. Σε κάθε εσωτερικό κόμβο $K_1 < K_2 < \dots < K_{q-1}$
3. Για όλες τις τιμές X στο υποδέντρο που δείχνει το P_j ισχύει $K \le X < K_j$ για $1 < j < q$, $X < K_j$ για $j = 1$, και $K_{j-1} \le X$ για $i = q$

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

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

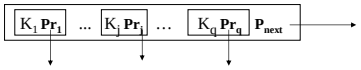
<<K_1, Pr_1>, <K_2, Pr_2>, … <K_q, Pr_q>, P_{next}>, $q \le p_{leaf}$, όπου

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 $K_1 < K_2 < \dots < K_q$

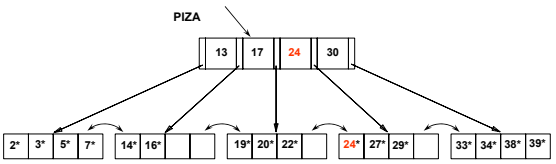
3. Κάθε κόμβος-φύλλο έχει το **πολύ p_{leaf}** τιμές

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

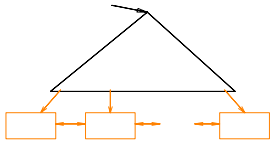
5. Όλοι οι κόμβοι-φύλλα βρίσκονται στο ίδιο επίπεδο.

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

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



- Εισαγωγή / Διαγραφή με κόστος $\log_F N$ --- κρατούν το δέντρο σε ισορροπημένη μορφή. (F = διακλάδωση, N = αριθμός των φύλλων)
- Ελάχιστη πληρότητα 50% (εκτός της ρίζας).
- Εξαιρετική δομή ΚΑΙ για ερωτήσεις ισότητας ΚΑΙ για ερωτήσεις διαστήματος (range queries).
- Το αρχείο δεδομένων μπορεί να είναι ή όχι ταξινομημένο



Καταχωρήσεις Ευρετηρίου (Άμεση Αναζήτηση)

Καταχωρήσεις Δεδομένων («Σύνολο ακολουθίας»)

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

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 $p_{leaf} = 31$

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%. Πόσες καταχωρήσεις (τιμές) χωρά αν 3 επίπεδα

Ρίζα	1 κόμβος	22 (33*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 το B-δέντρο

Σημείωση: εγγραφές μόνο στα φύλλα

Παρατηρήσεις

- Τυπική Τάξη: 100. Τυπικός Παράγων Πληρότητας: 67%.
- Μέση τιμή διακλάδωσης (fan out) = 133
- Τυπικές Δυνατότητες:
  - Ύψος 4: 133^4 = 312,900,700 εγγραφές
  - Ύψος 3: 133^3 = 2,352,637 εγγραφές
- Μπορεί να κρατά *τα υψηλότερα επίπεδα στη μνήμη* (buffer):
  - Επίπεδο 1 = 1 block = 8 Kbytes
  - Επίπεδο 2 = 133 blocks = 1 Mbyte
  - Επίπεδο 3 = 17,689 blocks = 133 MBytes


```
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, K)
end
```

B⁺-δέντρα: Αναζήτηση

Αναζήτηση (αναδρομική εκδοχή)

```
nodepointer find(keyvalue K):
        return tree_search(root, 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 $k = \lfloor ((p_{leaf}+1)/2) \rfloor$ παραμένουν στον κόμβο
-- οι υπόλοιπες σε καινούργιο κόμβο
-- εισαγωγή (αντιγραφή) της k-οστής τιμής (K_k) στον γονέα

B⁺-δέντρα: Εισαγωγή

Αν ένας εσωτερικός κόμβος 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 (εκτός ρίζας) "μεγαλώνουν" το δέντρο

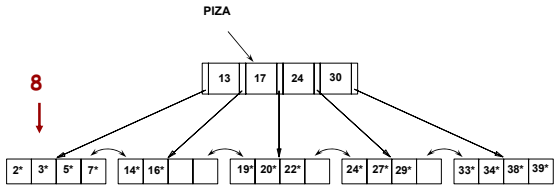
Η διάσπαση της ρίζας " υψώνει " το δέντρο

B⁺-δέντρα (παράδειγμα)

5, 8, 7, 14, 19, 6, 10 και τάξη $p = 3$ (2 τιμές ανά κόμβο, 3 δείκτες block ευρετηρίου) και $p_{leaf} = 2$

B+-δέντρα: Εισαγωγή

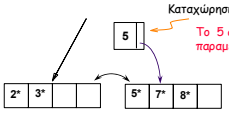
Εισαγωγή της καταχώρησης 8*



B+-δέντρα: Εισαγωγή

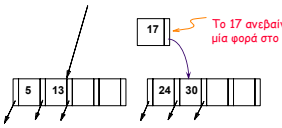
Καταχώρηση στον κόμβο γονέα (αντιγραφή)

Το 5 ανεβαίνει επάνω, αλλά παραμένει και στο φύλλο

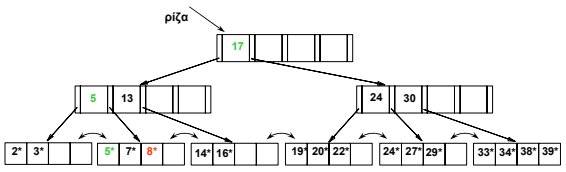


Καταχώρηση στον κόμβο γονέα (μεταφορά)

Το 17 ανεβαίνει επάνω και παρουσιάζεται μόνο μία φορά στο ευρετήριο (σε αντίθεση με τα 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 *πιο αριστερή τιμή* στο φύλλο του υποδέντρου με ρίζα το δείκτη στα δεξιά του K)

B+-δέντρα: Διαγραφή

Διαγραφή

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

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

2. Αν υποχείλιση (αναλυτικά)

‹ανακατανομή εγγραφών›

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

αν όχι, προσπάθεια ανακατανομής με το δεξιό αδελφό

ανακατανομή εγγραφών σε κάθε κόμβο

βρείτε την εγγραφή στο γονέα του δεξιού κόμβου N

αντικατάσταση της τιμής κλειδιού στο γονέα τους με τη μικρότερη τιμή του κόμβου N

‹συγχώνευση κόμβων›

Αν δεν είναι δυνατή η ανακατανομή

συγχώνευση κόμβων

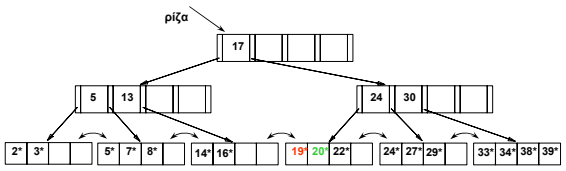
οδηγεί σε διαγραφή στο παραπάνω επίπεδο, σβήνεται η εγγραφή που δείχνει στον κόμβο (πιθανότητα νέας υποχείλισης)

B⁺-δέντρα: Διαγραφή

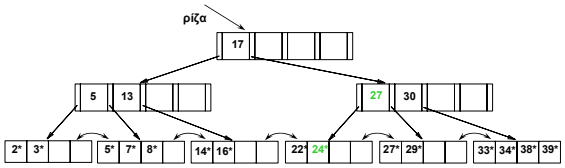
Στην περίπτωση συγχώνευσης, πρέπει να *διαγραφεί* η καταχώρηση (που δείχνει στον P ή τον αδελφό) από τον πατέρα του P.

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

B⁺-δέντρα: Παράδειγμα



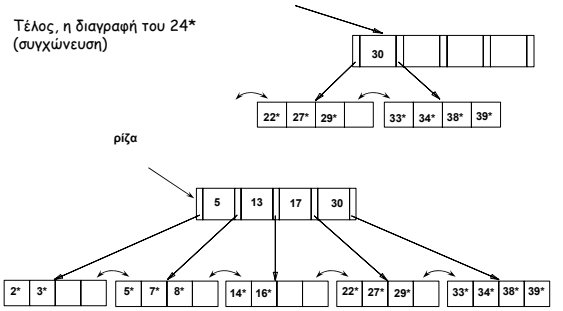
B⁺-δέντρα: Διαγραφή



Το παράδειγμα μετά τη διαγραφή του 19* και του 20* (ανακατανομή με δεξί αδελφό και αντικατάσταση του 24 με 27)

B⁺-δέντρα: Διαγραφή

Τέλος, η διαγραφή του 24* (συγχώνευση)

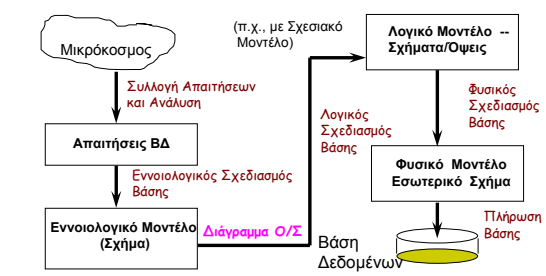


Ευρετήρια (ανακεφαλαίωση)

Είδη Ευρετηρίων

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

Φυσικός Σχεδιασμός



Φυσικός Σχεδιασμός

- Μετά τον σχεδιασμό Ο/Σ και το λογικό σχεδιασμό (σχεσιακό μοντέλο), έχουμε τα εννοιολογικά και λογικά (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)

Ευρετήρια

Η SQL-92 δεν περιλαμβάνει εντολές για τη δημιουργία ευρετηρίων. Τα περισσότερα εμπορικά ΣΔΒΔ το υποστηρίζουν

create [unique] index <index_name>
on <table_name> (<attr_list>);

- Η <attr_list> μπορεί να περιέχει παραπάνω από ένα γνωρίσματα.
- Προαιρετικό UNIQUE σημαίνει ότι το <attr_list> είναι κλειδί του <table_name>.

Ευρετήρια

drop index <index_name>

- Η Oracle δημιουργεί αυτόματα ευρετήρια για κάθε UNIQUE ή PRIMARY KEY ορισμό.

select <index_name> **from user_indexes**

Φυσικός Σχεδιασμός

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

Κατανοήσουμε το **Φόρτο Εργασίας (workload)**
- Ποιές είναι οι σημαντικές ερωτήσεις και πόσο συχνά εμφανίζονται.
- Ποιές είναι οι πιο σημαντικές τροποποιήσεις και πόσο συχνά εμφανίζονται.
- Ποια είναι η επιθυμητή επίδοση για την εκτέλεση αυτών των ερωτήσεων και τροποποιήσεων.

Φυσικός Σχεδιασμός

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

Ένα ευρετήριο κάνει τις ερωτήσεις ΠΙΟ ΓΡΗΓΟΡΕΣ και τις ενημερώσεις ΠΙΟ ΑΡΓΕΣ

Επιπλέον, απαιτεί και χώρο στον δίσκο

Φυσικός Σχεδιασμός

Για κάθε ερώτηση (query) το 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):
- 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) …

- 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