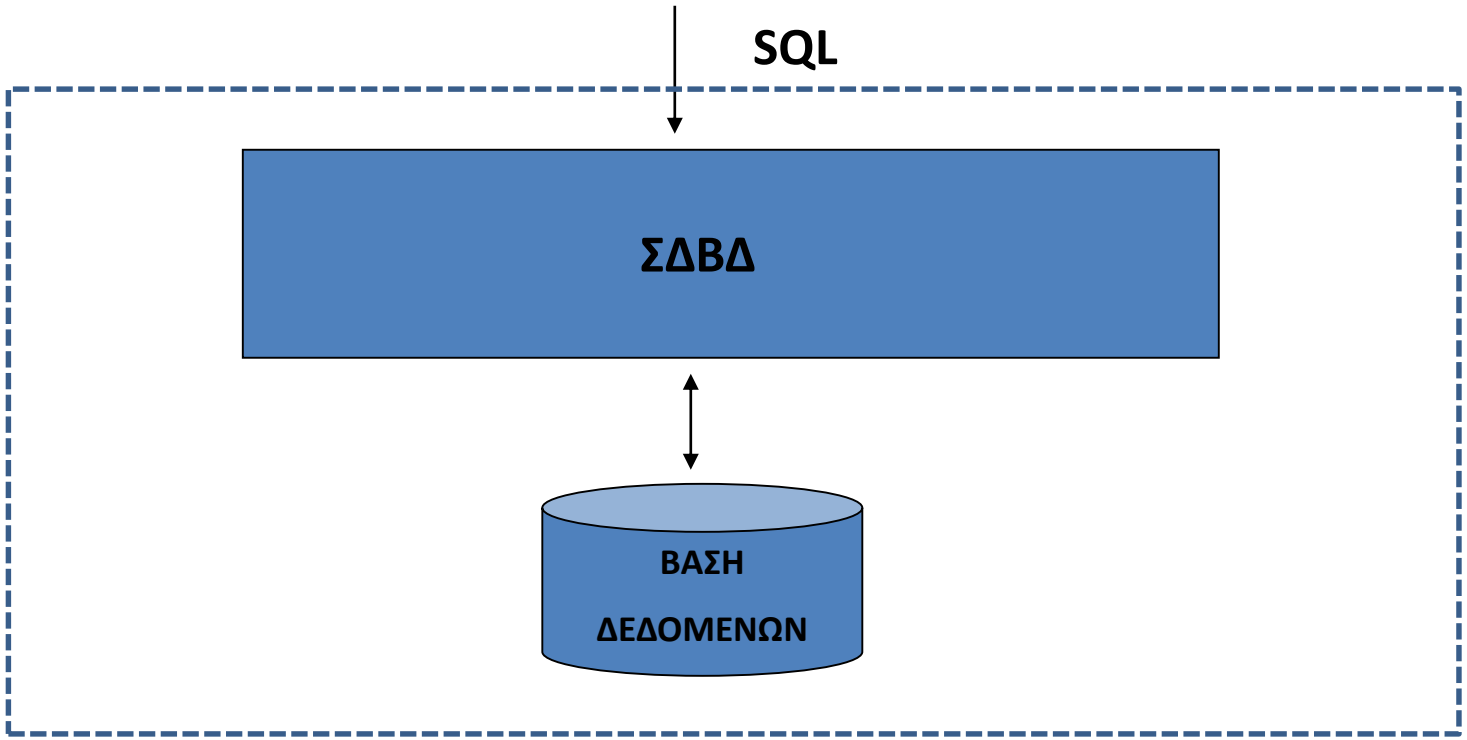
Αποθήκευση Δεδομένων

ΣΔΒΔ



Τυπικά,

- Κάθε *σχέση* (το στιγμιότυπο της) αποθηκεύεται σε *ένα αρχείο*

Δομή ενός ΣΔΒΔ (πιο αναλυτικά)

Η (εσωτερική) αρχιτεκτονική ενός ΣΔΒΔ είναι σε επίπεδα

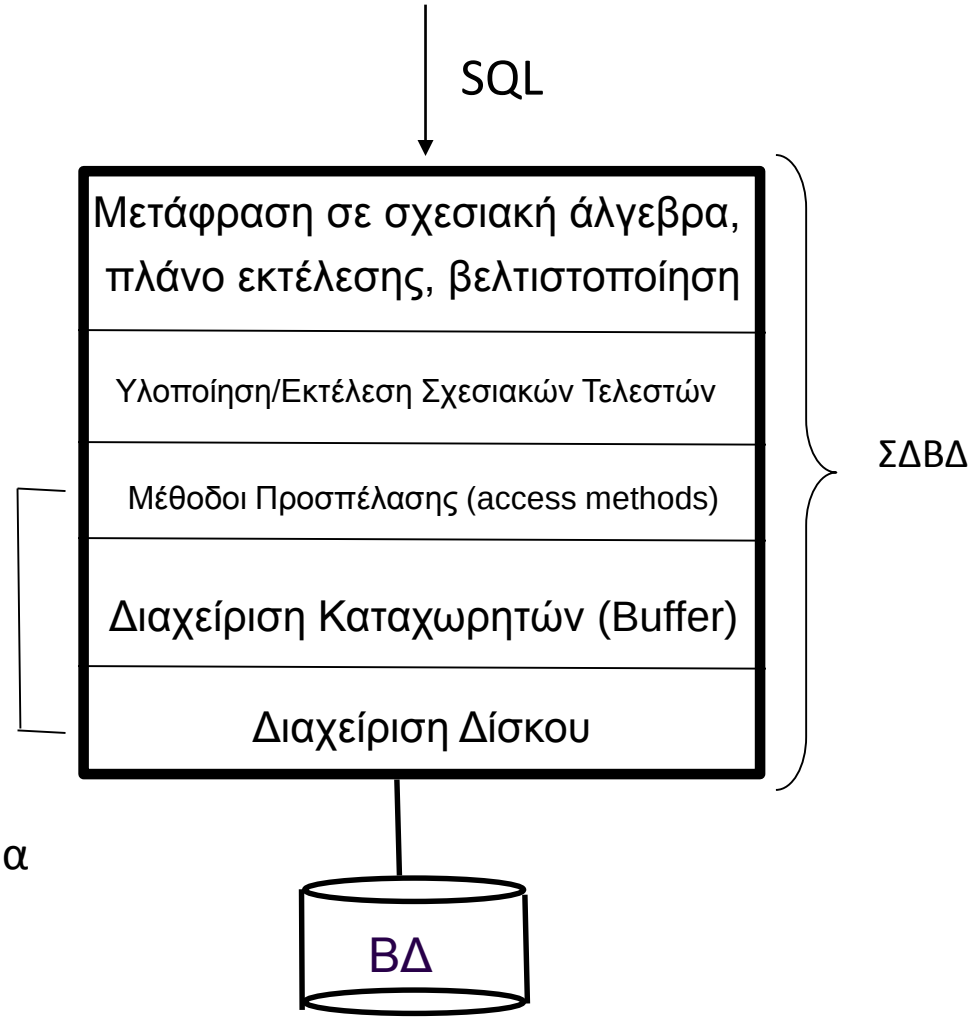
Αρχικά θα δούμε:

Αποθήκευση

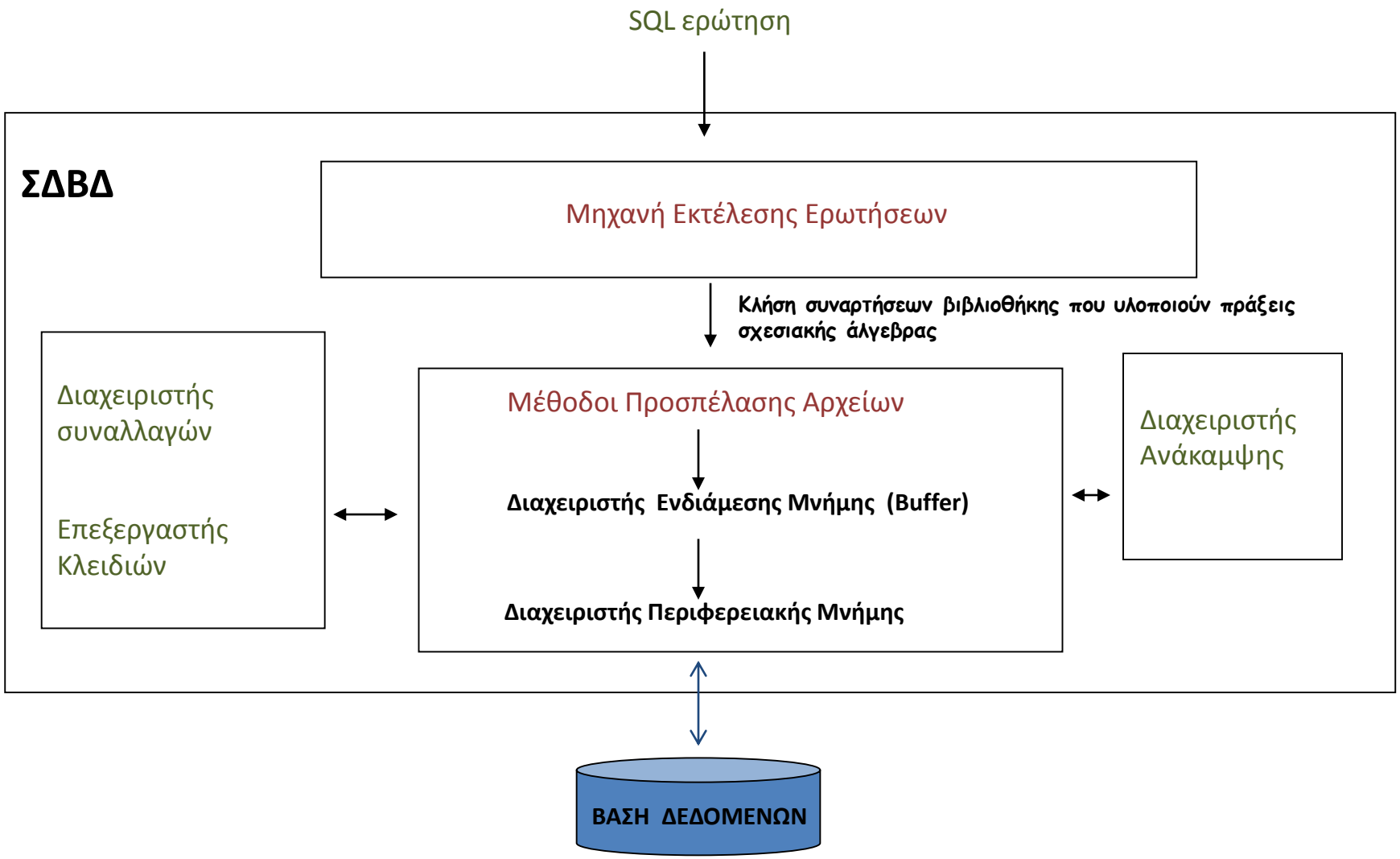
Δομή αρχείων

Στη συνέχεια

Τα παραπάνω επίπεδα



Δομή ενός ΣΔΒΔ



Αρχεία

Τυπικά,

- Κάθε *σχέση* (το στιγμιότυπο της) αποθηκεύεται σε *ένα αρχείο*
- Η αποθήκευση είναι οριζόντια: κάθε πλειάδα της σχέσης αντιστοιχεί σε μια εγγραφή του αρχείου
  - Δηλαδή, ένα αρχείο είναι μια ακολουθία από πλειάδες

Σύγχρονη Τάση: Column stores (κάθετη αποθήκευση ή αποθήκευση ανά στήλη)

Ευρετήρια – Βοηθητικές δομές για την προσπέλαση στα αρχεία

Κατάλογος Συστήματος

Για κάθε σχέση:

- όνομα, αρχείο, δομή αρχείου (πχ αρχείο σωρού)
- Όνομα και τύπο για κάθε γνώρισμα
- *Όνομα ευρετηρίου για κάθε ευρετήριο*
- Περιορισμοί ακεραιότητας

Για κάθε ευρετήριο:

Δομή (πχ B+ δέντρο) και κλειδιά αναζήτησης

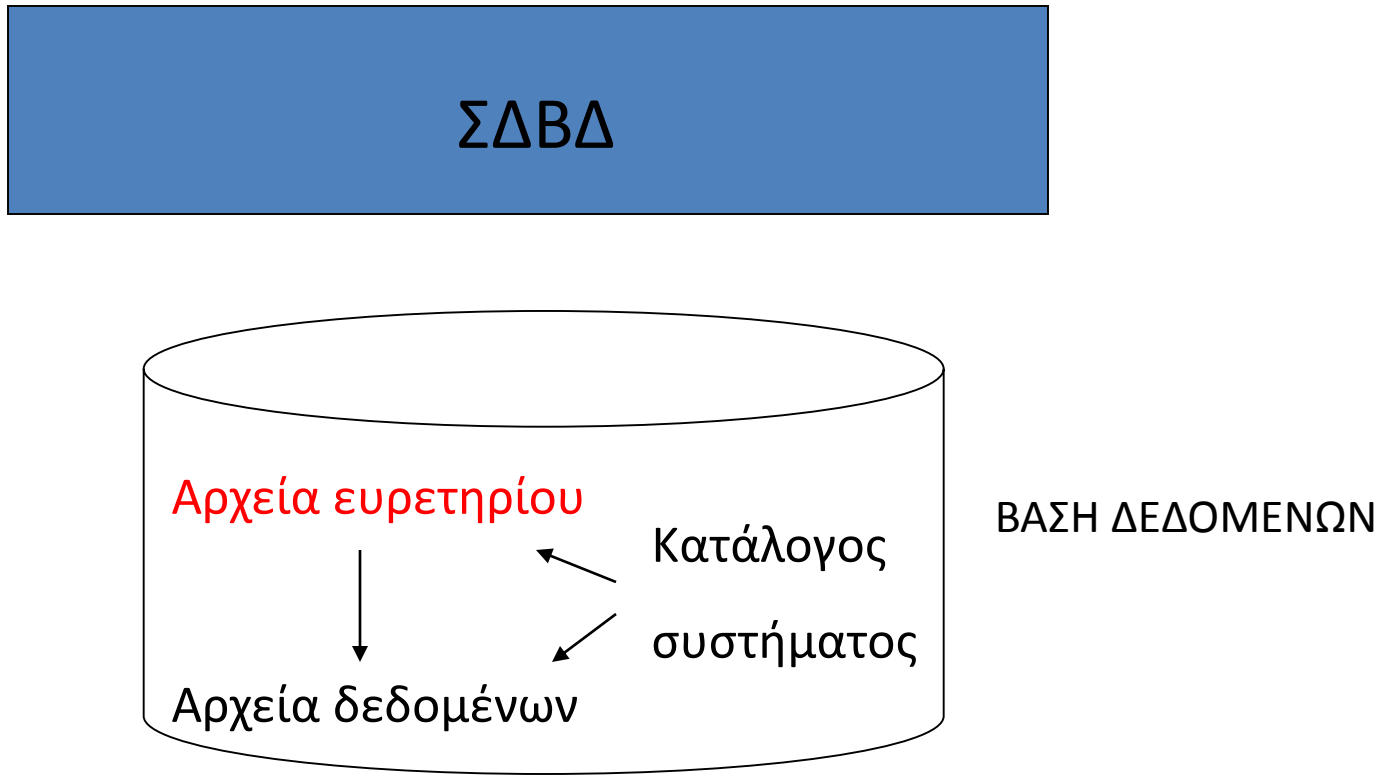
Για κάθε όψη:

Το όνομα και τον ορισμό της

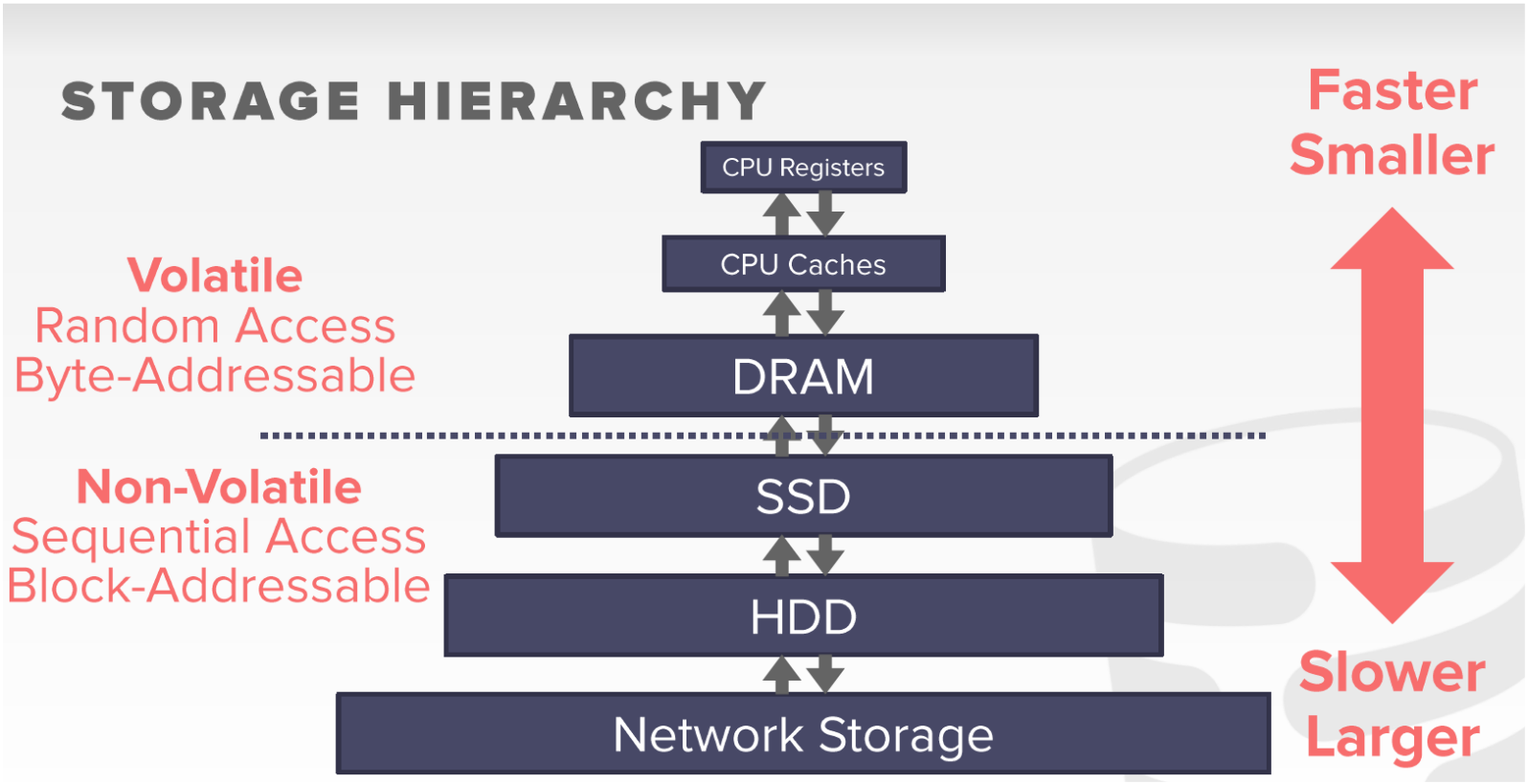
Επίσης, στατιστικά, μέγεθος του buffer pool, δικαιώματα προσπέλασης κλπ.

Ο κατάλογος αποθηκεύεται επίσης ως σχέση

Το εσωτερικό ενός ΣΔΒΔ



Ιεραρχία Αποθήκευσης



Αποθηκευτικές Μονάδες

πρωτεύουσα αποθήκευση (primary storage)

κύρια μνήμη (main memory) - κρυφή μνήμη (cache)

- τυχαία προσπέλαση (random access)
- σε επίπεδο bytes
- άμεση προσπέλαση από την κύρια ΚΜΕ (CPU)
- γρήγορη προσπέλαση
- περιορισμένη χωρητικότητα αποθήκευσης

Αποθηκευτικές Μονάδες

Δευτερεύουσα αποθήκευση

- για την επεξεργασία των δεδομένων *απαιτείται η μεταφορά των δεδομένων στην πρωτεύουσα αποθήκευση*
- πιο αργή προσπέλαση
- μεγάλη χωρητικότητα
- μικρότερο κόστος (για την ίδια ποσότητα χώρου η κύρια μνήμη 100 φορές ακριβότερη από τη δευτερεύουσα)
- σειριακή προσπέλαση (τυχαία πιο αργή)
- σε επίπεδο block

Αποθηκευτικές Μονάδες

Η βάση δεδομένων θα πρέπει να αποθηκευτεί σε κάποιο αποθηκευτικό μέσο

Οι περισσότερες βάσεις δεδομένων αποθηκεύονται **σε δευτερεύουσες αποθηκευτικές μονάδες** (κυρίως σε δίσκους)

- ✓ πολύ μεγάλες (10-100 TB) $\Rightarrow$ μεγάλο κόστος ($1/GB – 100$/GB)
- ✓ μόνιμη αποθήκευση (nonvolatile storage)

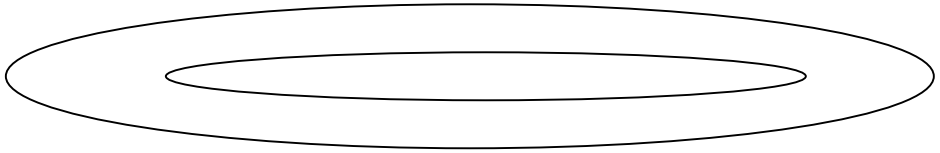
Σε αργότερους αποθηκευτικούς χώρους για

- τήρηση εφεδρικών αντιγράφων
- αρχειοθέτηση (archiving) (δεδομένα που θέλουμε να κρατήσουμε για πολύ καιρό αλλά η προσπέλαση τους είναι σπάνια)

Μαγνητικοί Δίσκοι

- Μαγνητισμός μιας περιοχής του δίσκου κατά ορισμένο τρόπο ώστε 1 ή 0

- *Χωρητικότητα* (capacity) σε Kbyte - Mbyte - Gbyte
- Μαγνητικό υλικό σε σχήμα κυκλικού δίσκου

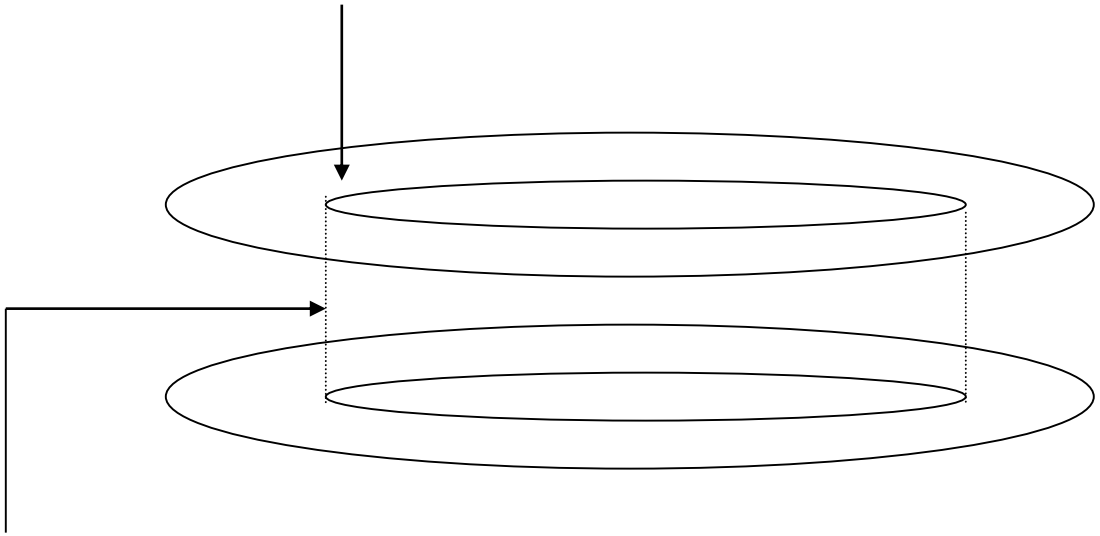


  - Απλής και διπλής όψης

Δίσκοι

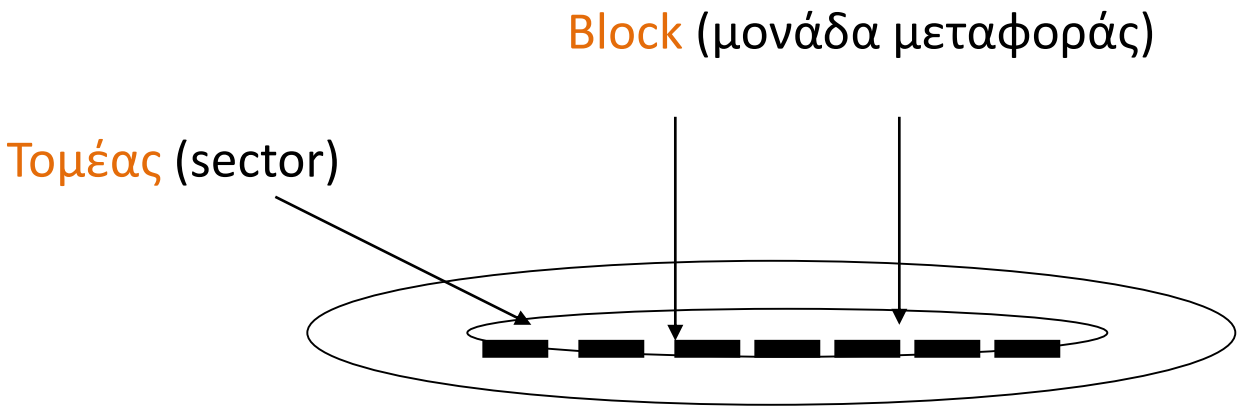
Σε πακέτα δίσκων

Οι πληροφορίες σε ομόκεντρους κύκλους διαφορετικής διαμέτρου: άτρακτοι (track)
(συνήθως κάθε άτρακτος την ίδια ποσότητα πληροφορίας)



Ομόκεντροι κύκλοι σε διαφορετικές επιφάνειες: κύλινδρος (cylinder)

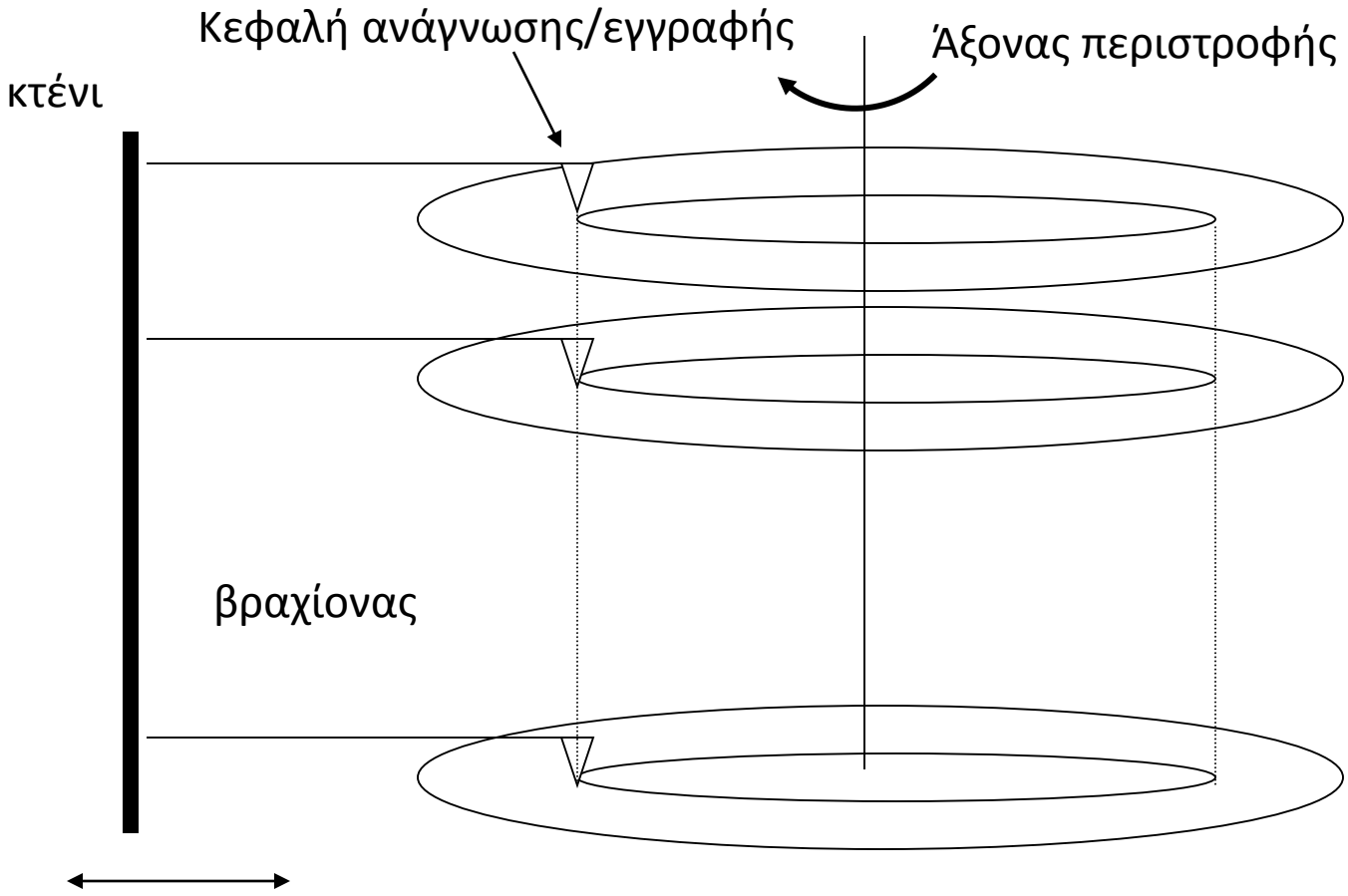
Δίσκοι



Κάθε άτρακτος χωρίζεται σε τόξα που ονομάζονται τομείς (sectors) και είναι χαρακτηριστικό του κάθε δίσκου και δε μπορεί να τροποποιηθεί

Το μέγεθος ενός block τίθεται κατά την αρχικοποίηση του δίσκου και είναι κάποιο πολλαπλάσιο του τομέα

Δίσκοι


Κεφαλή ανάγνωσης/εγγραφής
Άξονας περιστροφής
κτένι
βραχίονας

Δίσκοι

χρόνος εντοπισμού (seek time) Τοποθέτηση κεφαλής στη σωστή άτρακτο

χρόνος περιστροφής (rotational delay ή latency) Ώσπου η αρχή του σωστού block να βρεθεί κάτω από την κεφαλή

χρόνος μεταφοράς block (block transfer time) χρόνος μεταφοράς δεδομένων από το δίσκο στη μνήμη

Χρόνος προσπέλασης =

χρόνος εντοπισμού + χρόνος περιστροφής + χρόνος μεταφοράς

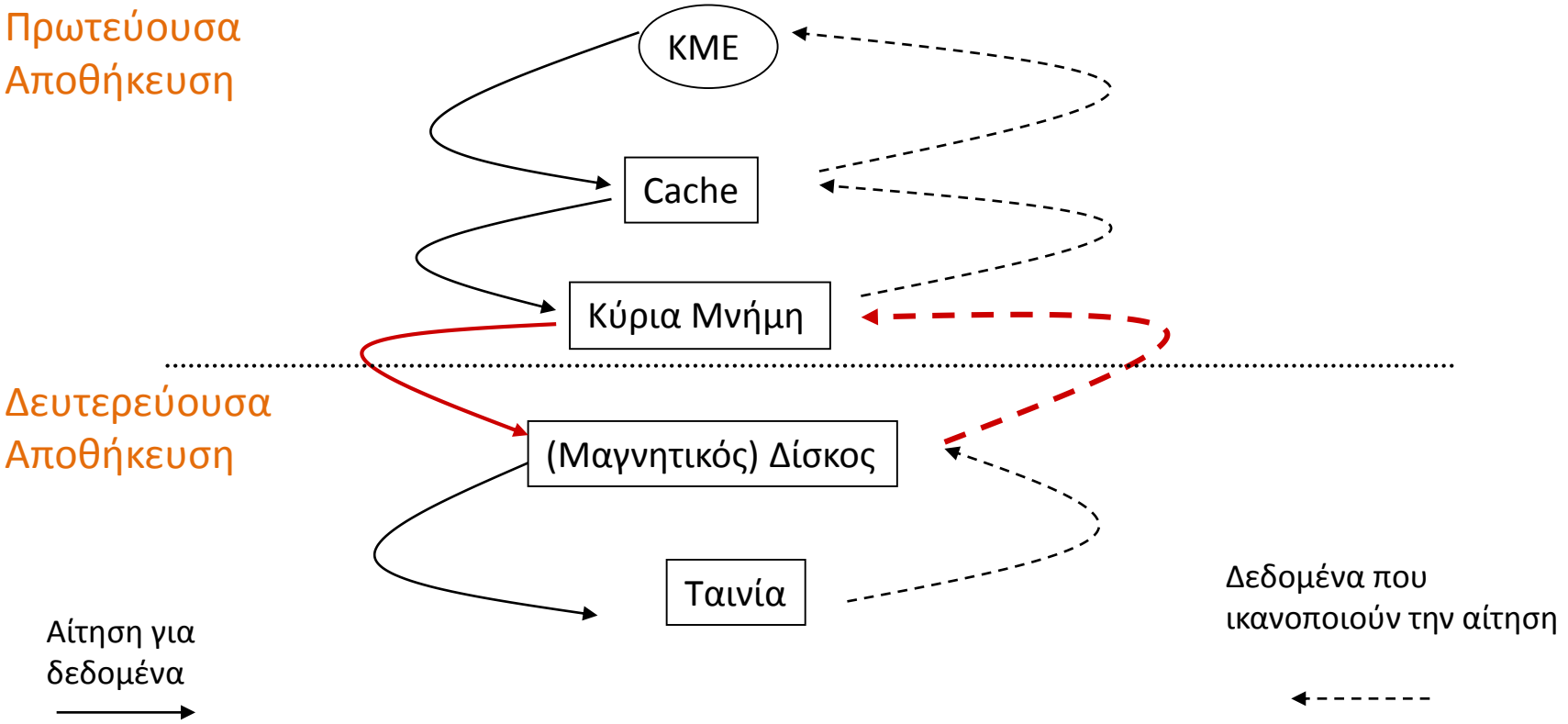
Μεταφορά αρκετών γειτονικών block

(Solid State) Disks

Flash memory (solid state) δευτερεύουσα αποθήκευση

- Μεγαλύτερη αντοχή από μαγνητικούς δίσκους, πιο ελαφριά, γρηγορότερη προσπέλαση (access time)
- Μνήμη (κύκλωμα) -- Δεν έχουν κινητό μηχανικό μέρος -- Δεν έχει χρόνο εντοπισμού και περιστροφής
- Τρεις λειτουργίες: Read, Write, Erase
- Πριν γίνει εγγραφή, πρέπει να προηγηθεί Erase
- Erase και Write πολύ πιο αργά από το Read

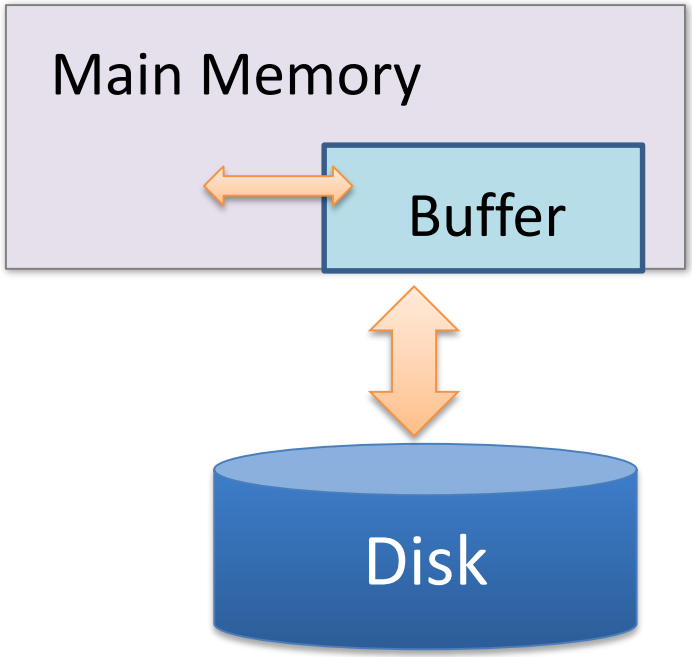
Ιεραρχία Μνήμης



Καταχωρητής

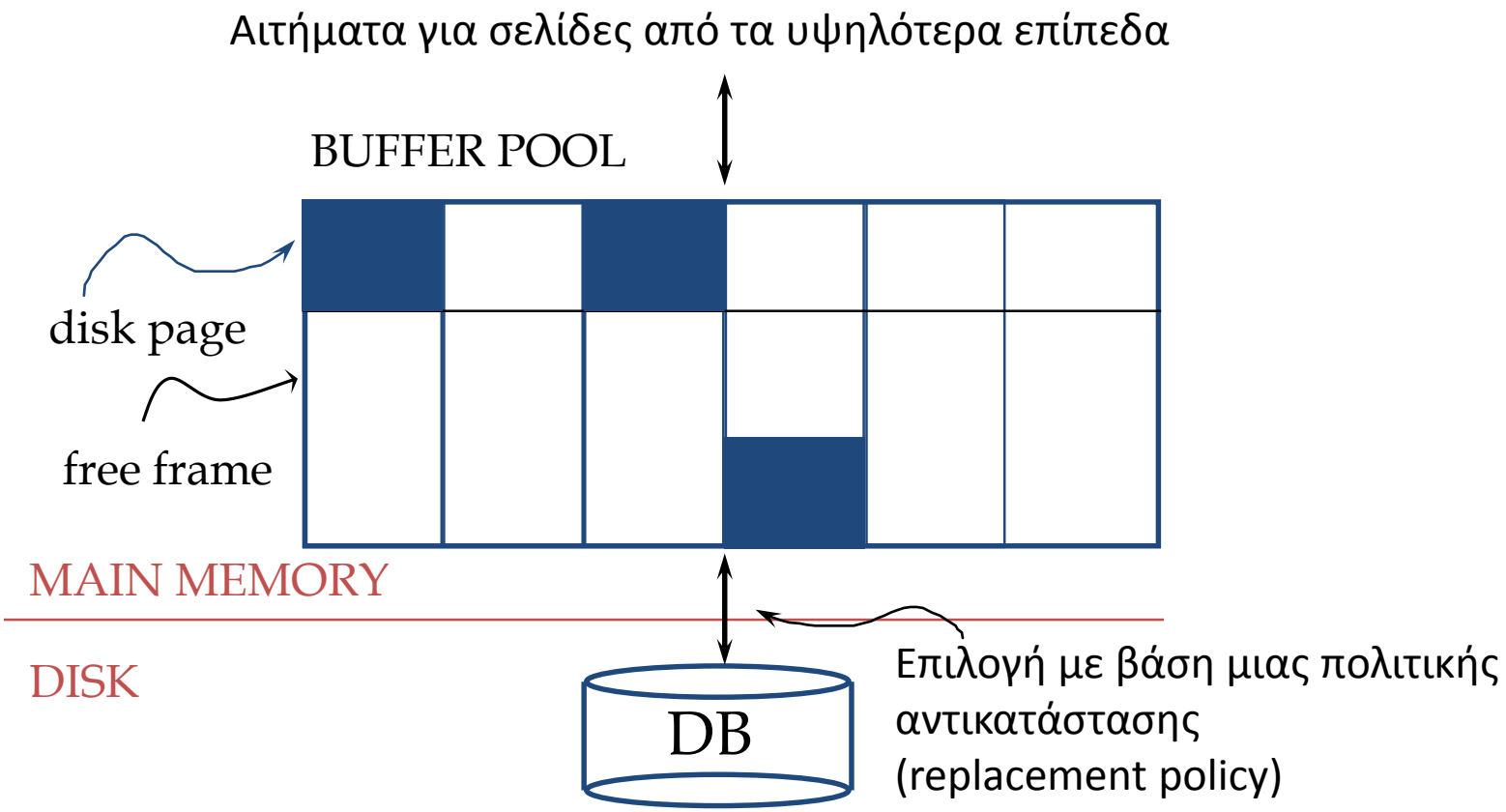
- Ο διαχειριστής ενδιάμεσης μνήμης (buffer management) είναι υπεύθυνος για την μεταφορά όταν χρειάζεται σελίδων από το δίσκο στην κύρια μνήμη

- Τα ΣΔΒΔ διατηρούν τον δικό τους buffer γιατί έχουν περισσότερη πληροφορία από το ΛΣ (Βασική λειτουργικότητα: πολιτική αντικατάστασης σελίδων)



Καταχωρητής

Μεταφορά σε επίπεδο σελίδας
Διαθέσιμες σελίδες (buffer pool) – ονομάζονται και πλαίσια (frame)



Βασικά Σημεία

1. Για να γίνει οποιοσδήποτε υπολογισμός, τα δεδομένα πρέπει να βρίσκονται στη μνήμη

2. Η μονάδα μεταφοράς από το δίσκο στη μνήμη είναι ένα block. *Ακόμα και αν χρειαζόμαστε ένα μόνο αντικείμενο, πρέπει να μεταφερθεί ολόκληρο το block του.* Το διάβασμα ή γράψιμο ενός block ονομάζεται λειτουργία Εισόδου/Εξόδου (Input/Output – I/O).

3. Ο χρόνος προσπέλασης (εγγραφής ή ανάγνωσης) ενός block διαφέρει και εξαρτάται από τη θέση του block

χρόνος προσπέλασης = χρόνος εντοπισμού + χρόνου περιστροφής + χρόνος μεταφοράς

Βασικός στόχος η ελαχιστοποίηση της επικοινωνίας με το δίσκο:

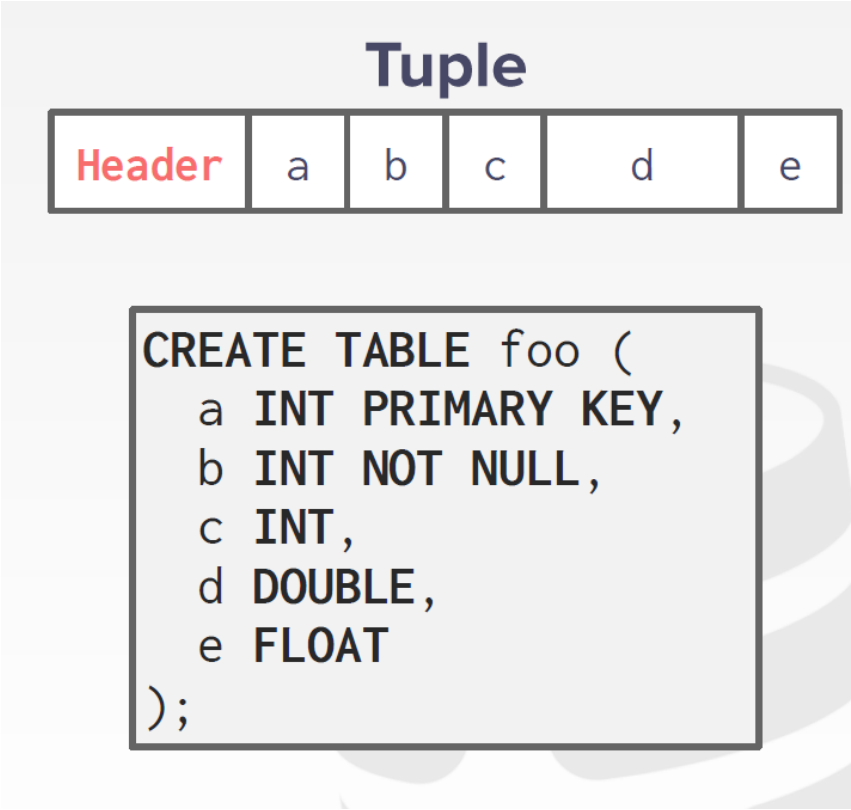
ελαχιστοποίηση του αριθμού των blocks που μεταφέρονται μεταξύ της πρωτεύουσας (κύριας μνήμης, cache – ενδιάμεση μνήμη – buffers-καταχωρητές) και της δευτερεύουσας αποθήκευσης (δίσκος)

Οργάνωση Αρχείων

Αρχεία

Κάθε πίνακας αποθηκεύεται σε ένα αρχείο

Μια πλειάδα του πίνακα – μια εγγραφή του αρχείου



Επικεφαλίδα, θα την αγνοούμε

Αρχεία

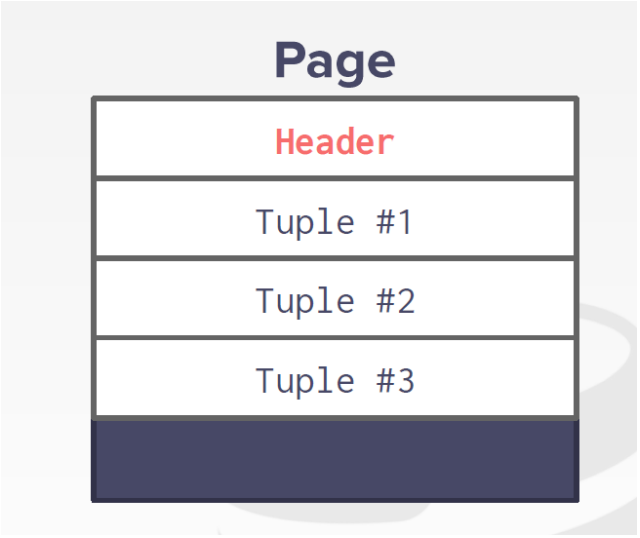
Ένα αρχείο είναι λογικά οργανωμένο ως μια συλλογή από σελίδες (pages)

Κάθε σελίδα – ακολουθία από εγγραφές

θα θεωρούμε page = block

Κάθε σελίδα έχει ένα μοναδικό αναγνωριστικό (page id)

✓ *Μπορούμε να βλέπουμε μια σελίδα ως μια συλλογή «θέσεων» που κάθε μία περιέχει μια εγγραφή*
✓ *Μια εγγραφή προσδιορίζεται από τη χρήση του ζεύγους (page id, slot number)*



Εγγραφές

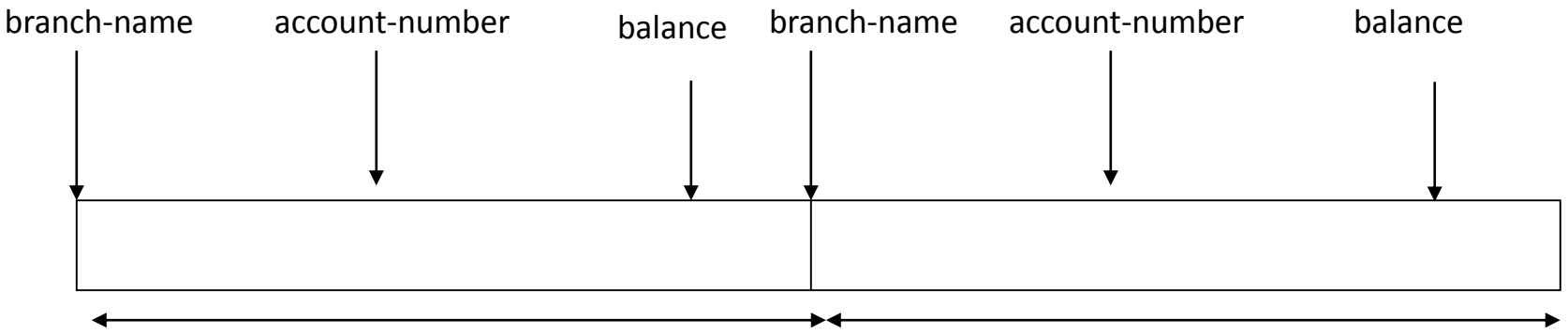
Πως οργανώνονται τα πεδία μέσα σε μία εγγραφή;

Εγγραφές σταθερού και μεταβλητού μήκους

```
type account = record
        branch-name: char(22);
        account-number: char(20);
        balance: real;
        end
```

Έστω κάθε char 1 byte - real 8 bytes

Κάθε εγγραφή 50 bytes



Εγγραφές

Γιατί είναι προτιμότερες οι εγγραφές σταθερού μήκους:

εύκολος ο εντοπισμός ενός πεδίου και η διατήρηση πληροφορίας για «άδειες» θέσεις

Εγγραφές

Πως προκύπτουν οι εγγραφές μεταβλητού τύπου;

Στο σχεσιακό μοντέλο κάθε εγγραφή (πλειάδα) μιας σχέσης περιέχει το ίδιο πλήθος πεδίων (αριθμό γνωρισμάτων).

- Εγγραφές του ίδιου τύπου αλλά έχουν ένα ή περισσότερα πεδία *μεταβλητού μεγέθους*
- Ανάμεικτο (mixed) αρχείο: εγγραφές διαφορετικού τύπου

Εγγραφές

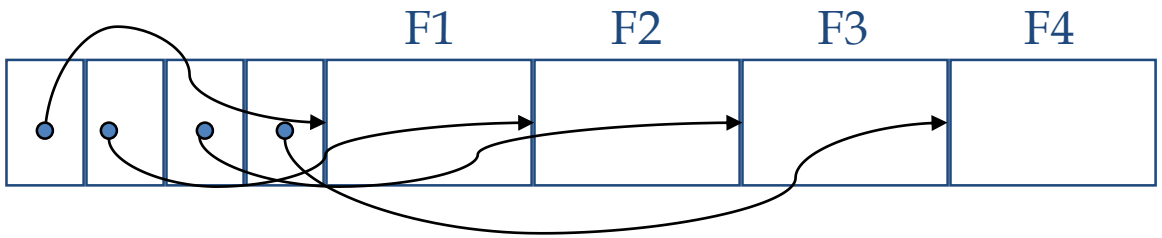
- Αποθήκευση των πεδίων συνεχόμενα, χωρισμένα με διαχωριστές (ειδικούς χαρακτήρες που δεν εμφανίζονται ως δεδομένα)

Εγγραφές

- Χώρο στην αρχή κάθε εγγραφής – πίνακας ακεραίων I[j] όπου j η μετατόπιση (offset) του j-οστής πεδίου (κρατά την αρχή του j-οστού πεδίου) + τη μετατόπιση του τέλους της εγγραφής

  απευθείας πρόσβαση σε οποιαδήποτε πεδίο

  καλό χειρισμό της τιμής null



Εγγραφές

- Ως εγγραφές σταθερού μήκους, θεωρώντας το μέγιστο μέγεθος για κάθε εγγραφή

Παράγοντας Ομαδοποίησης (blocking factor)

Η μονάδα μεταφοράς μεταξύ δίσκου και μνήμης είναι ένα block δίσκου (σελίδα)

Έστω εγγραφές σταθερού μήκους R – μέγεθος block B

Όταν $B \geq R$ περισσότερες από μια εγγραφή ανά block

Παράγοντας ομαδοποίησης (blocking factor), όταν $B \geq R$

$bfr = \lfloor (B / R) \rfloor$, όπου B μέγεθος block σε byte και R μέγεθος εγγραφής σε bytes

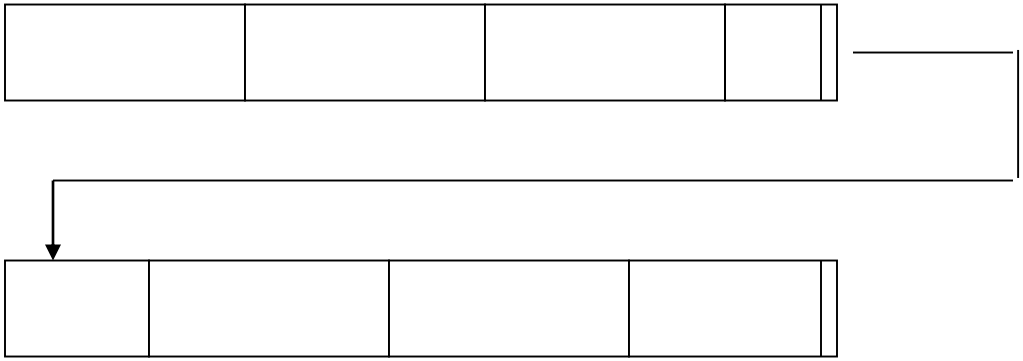
Δηλαδή, πόσες «ολόκληρες» εγγραφές χωρούν σε ένα block

Εκτεινόμενη (spanned) Καταχώρηση

Μη εκτεινόμενη (unspanned) οργάνωση: οι εγγραφές δεν επιτρέπεται να διασχίζουν τα όρια ενός block

- Αχρησιμοποίητος χώρος: B - bfr * R bytes ανά block
- Πιο εύκολη η προσπέλαση

Εκτεινόμενη (spanned) οργάνωση: αποθήκευση μέρους μιας εγγραφής σε ένα block και το υπόλοιπο σε ένα άλλο block - δείκτης στο τέλος του πρώτου τμήματος δείχνει στο block που περιέχει το υπόλοιπο



Μη Εκτεινόμενη Καταχώρηση

Αριθμός blocks για την (μη εκτεινόμενη) αποθήκευση ενός αρχείου r εγγραφών:

$$b = \lceil (r/bfr) \rceil$$

Αποθήκευση Δεδομένων

Έστω μία σχέση R(A, B, C, D, E), τα γνωρίσματα A, B, D και E είναι τύπου ακέραιοι μεγέθους 16 bytes και το γνώρισμα C σειρά χαρακτήρων μεγέθους 36 bytes. Έστω αρχείο με r_A = 30.000 εγγραφές, μέγεθος block B = 1024 bytes, και μη εκτεινόμενη καταχώρηση.

Μέγεθος αρχείου δεδομένων: 3.000 blocks

Τοποθέτηση σελίδων αρχείου στο δίσκο

συνεχόμενη τοποθέτηση (contiguous allocation) τα block του αρχείου τοποθετούνται σε διαδοχικά blocks του δίσκου

συνδεδεμένη τοποθέτηση (linked allocation) κάθε block του αρχείου περιλαμβάνει ένα δείκτη προς το επόμενο block του αρχείου

Εύκολη επέκταση - πιο αργή ανάγνωση όλου του αρχείου

συστάδες διαδοχικών blocks δίσκου: τμήματα (segments) ή επεκτάματα (extents)

ευρετηριοποιημένη τοποθέτηση (indexed allocation)

Επικεφαλίδα Αρχείου

Μια επικεφαλίδα ή περιγραφέας αρχείου (file header ή file descriptor) περιέχει πληροφορίες σχετικά με ένα αρχείο που είναι απαραίτητες στα προγράμματα που προσπελαύνουν τις εγγραφές του αρχείου

Πληροφορίες για προσδιορισμό διεύθυνσης των blocks αρχείου στο δίσκο + περιγραφές μορφοποίησης εγγραφών

Αποθηκεύεται στο αρχείο

θεωρούμε ότι «ξέρουμε» σε ποιο block του δίσκου είναι αποθηκευμένη η i-οστή σελίδα (block) του αρχείου

Οργάνωση Εγγραφών Αρχείου

Θα συζητήσουμε πως πρέπει να οργανώσουμε τις εγγραφές σε ένα αρχείο για αποδοτική επεξεργασία ερωτήσεων

Βασικές λειτουργίες:

- Εισαγωγή/διαγραφή/τροποποίηση εγγραφής
- Εντοπισμός (αναζήτηση) μια συγκεκριμένης εγγραφής με βάση συνθήκη ισότητας ή διαστήματος τιμών
- Διάσχιση (scan) όλων των εγγραφών του αρχείου

Οργάνωση Αρχείων

Βασικός στόχος η *ελαχιστοποίηση του αριθμού* των blocks που μεταφέρονται

Θεωρούμε ότι η πληροφορία για τη θέση στο δίσκο ενός block υπάρχει (π.χ., στην επικεφαλίδα του αρχείου)

Σε πραγματικά συστήματα

- Ίσως και άλλοι τύποι κόστους (πχ κόστος CPU)
- Πρόσβαση κατά block (διάβασμα γειτονικών block με μια μόνο αίτηση I/O: αναζήτηση 1^{ου} block + μεταφορά όλων των επόμενων)

Οργάνωση Αρχείων

Οργάνωση αρχείων: πως είναι τοποθετημένες οι εγγραφές ενός αρχείου όταν αποθηκεύονται στο δίσκο

1. Αρχεία Σωρού
2. Διατεταγμένα Αρχεία

   Φυσική διάταξη των εγγραφών ενός αρχείου με βάση την τιμή ενός από τα πεδία του το οποίο λέγεται πεδίο διάταξης (ordering field)

Αρχεία Σωρού (Heap Files)

Αρχείο Σωρού (heap file ή pile file): Οι εγγραφές τοποθετούνται στο αρχείο με τη σειρά που εισάγονται

Μη διατεταγμένο αρχείο

1. Εισαγωγή

$2 * T_D + T_C$

2. Αναζήτηση (μέσος χρόνος)

$0.5 * B * (T_D + R * T_C)$

B #blocks

R #εγγραφών ανά block

$\mathbf{T_D}$ χρόνος μεταφοράς block

$\mathbf{T_C}$ χρόνος επεξεργασίας ανά εγγραφή

Αρχεία Σωρού (Heap Files)

3. Διαγραφή εγγραφής

Σημάδι διαγραφής

Περιοδική αναδιοργάνωση

Χρόνος Αναζήτησης + (T_C + T_D)

Αρχεία Σωρού (Heap Files)

4. Τροποποίηση εγγραφής

- εγγραφή μεταβλητού μήκους

5. Σάρωση (scan) Ανάγνωση όλων των εγγραφών

$B*(T_D + R*T_C)$

6. Ανάγνωση όλων των εγγραφών σε διάταξη

Εξωτερική ταξινόμηση συνήθως μια παραλλαγή της ταξινόμησης με συγχώνευση

Διατεταγμένο Αρχείο

Ταξινομημένα/Διατεταγμένα Αρχεία

Φυσική διάταξη των εγγραφών ενός αρχείου με βάση την τιμή ενός από τα πεδία του το οποίο λέγεται πεδίο διάταξης (ordering field)

Διατεταγμένο ή φυσικό αρχείο

- Αν το πεδίο διάταξης είναι και κλειδί τότε λέγεται και **κλειδί διάταξης**

Διατεταγμένο Αρχείο

1. Εισαγωγή

i. Εύρεση της σωστής θέσης της εγγραφής στο αρχείο

ii. Μετακίνηση εγγραφών για να κάνουμε χώρο για την εισαγωγή της

Κατά μέσο όρο μετακίνηση των μισών εγγραφών

Χρόνος αναζήτησης +

$2 * (0.5 * B * (T_D + R * T_C))$

B #blocks

R #εγγραφών ανά block

$\mathbf{T_D}$ χρόνος μεταφοράς block

$\mathbf{T_C}$ χρόνος επεξεργασίας ανά εγγραφή

Διατεταγμένο Αρχείο

1. Εισαγωγή (συνέχεια)

- ✓ Διατήρηση κάποιου αχρησιμοποίητου χώρου ανά block
- ✓ Δημιουργία ενός προσωρινού μη διατεταγμένου αρχείου (αρχείο υπερχείλισης) + κυρίως αρχείο

Διατεταγμένο Αρχείο

2. Αναζήτηση εγγραφής (με επιλογή ισότητας)

αποδοτική αν η συνθήκη αναζήτησης είναι στο πεδίο ταξινόμησης

Έστω B blocks, αναζήτηση της εγγραφής με τιμή *K* στο πεδίο διάταξης

Σημείωση: Υποθέτουμε ότι οι διευθύνσεις των blocks του αρχείου είναι αποθηκευμένες στην επικεφαλίδα του αρχείου

Διατεταγμένο Αρχείο

2. Αναζήτηση εγγραφής (συνέχεια)

```
lower := 1; upper := B;
while (upper ≥ lower)
      i := (lower + upper) div 2;
      read block i
      if (K < τιμής διάταξης της πρώτης εγγραφής)
            upper := i - 1;
      else if (K > τιμής διάταξης της  τελευταίας εγγραφής)
            lower := i + 1;
      else ...
```

Χρόνος: log B * (T_D + log R * T_C)

B #blocks

R #εγγραφών ανά block

$\mathbf{T_D}$ χρόνος μεταφοράς block

$\mathbf{T_C}$ χρόνος επεξεργασίας ανά εγγραφή

✓ Συνθήκη πχ., <= ??

Διατεταγμένο Αρχείο

3. Διαγραφή εγγραφής

Μετακίνηση εγγραφών

- ✓ Χρήση σημαδιού διαγραφής

4. Τροποποίηση εγγραφής

Διατεταγμένο Αρχείο

5. Ανάγνωση όλων των εγγραφών σε διάταξη

Αρχεία Κατακερματισμού

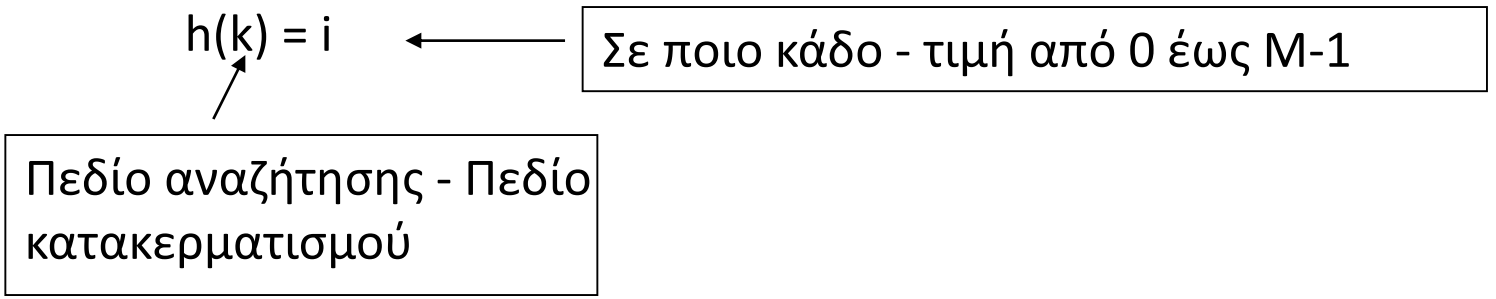
Βασική ιδέα: η τοποθέτηση των εγγραφών στα blocks του αρχείου γίνεται εφαρμόζοντας μια συνάρτηση κατακερματισμού σε κάποιο από τα πεδία της

Εσωτερικός Κατακερματισμός

Εσωτερικός Κατακερματισμός (τα δεδομένα είναι στη μνήμη, όπως στις δομές δεδομένων)

Πίνακας κατακερματισμού με Μ θέσεις - κάδους (buckets)

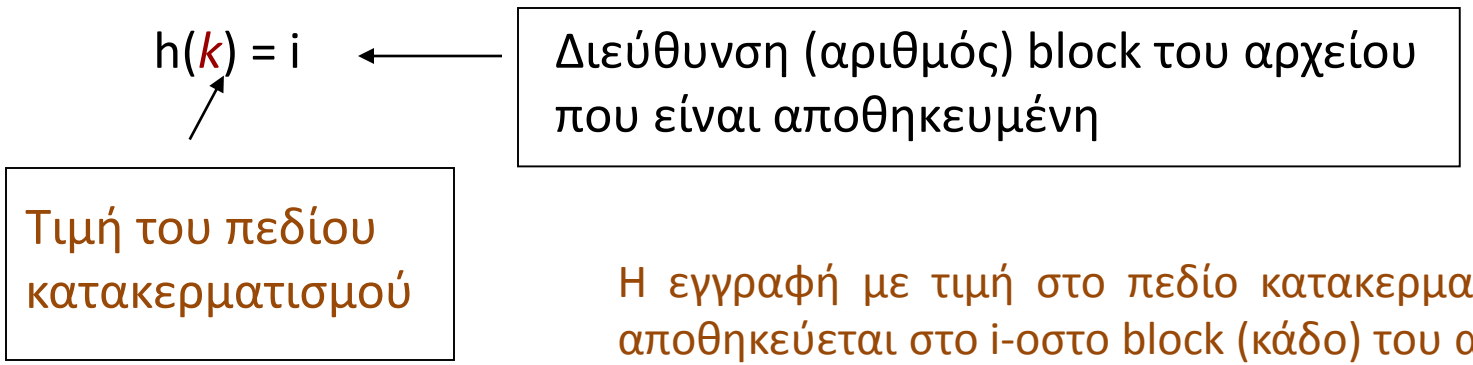
h: *συνάρτηση κατακερματισμού*



Αρχεία Κατακερματισμού

Εξωτερικός Κατακερματισμός (εφαρμογή σε δεδομένα αποθηκευμένα σε αρχεία)

Στόχος



Η εγγραφή με τιμή στο πεδίο κατακερματισμού *k* αποθηκεύεται στο i-οστο block (κάδο) του αρχείου

Κατακερματισμός

h: συνάρτηση κατακερματισμού

(Στόχος) Ομοιόμορφη κατανομή των κλειδιών στους κάδους (blocks)

- Συνηθισμένη συνάρτηση κατακερματισμού:

$$h(k) = k \bmod M$$

Κατακερματισμός

- Σύγκρουση (collision): όταν μια νέα εγγραφή κατακερματίζεται σε μία ήδη γεμάτη θέση

- Καλή συνάρτηση κατακερματισμού: κατανέμει τις εγγραφές ομοιόμορφα στο χώρο των διευθύνσεων (ελαχιστοποίηση συγκρούσεων και λίγες αχρησιμοποίητες θέσεις)

- Ευριστικοί:

  -- αν r εγγραφές, πρέπει να επιλέξουμε το M ώστε το r/M να είναι μεταξύ του 0.7 και 0.9

  -- όταν χρησιμοποιείται η mod τότε είναι καλύτερα το M να είναι πρώτος

Κατακερματισμός

Επίλυση Συγκρούσεων

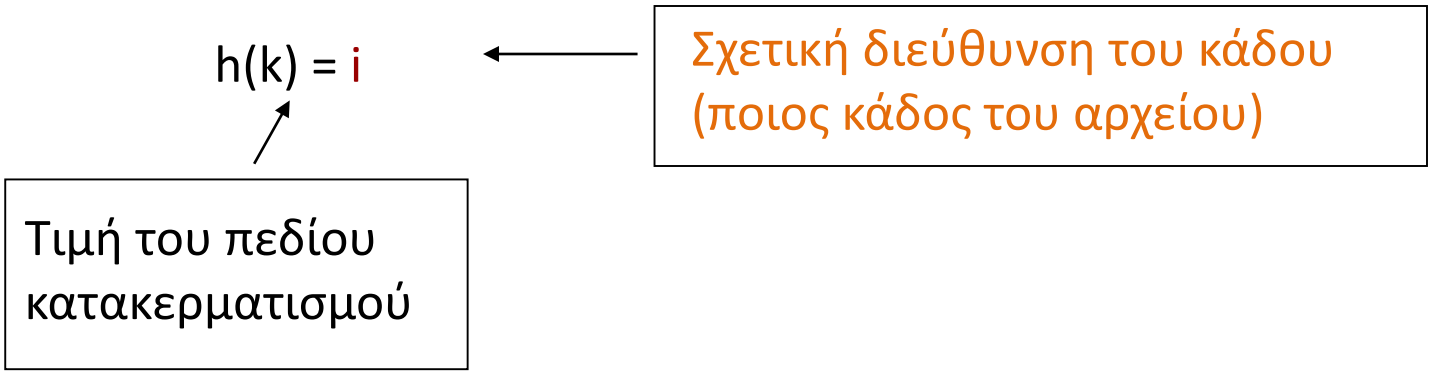
1. *Ανοιχτή Διευθυνσιοδότηση* (open addressing): χρησιμοποίησε την επόμενη κενή θέση

2. *Αλυσιδωτή Σύνδεση* (chaining): για κάθε θέση μια συνδεδεμένη λίστα με εγγραφές υπερχείλισης

3. *Πολλαπλός Κατακερματισμός* (multiple hashing): εφαρμογή μιας δεύτερης συνάρτησης κατακερματισμού

Εξωτερικός Κατακερματισμός

Κάδος: μια συστάδα από συνεχόμενα blocks του αρχείου



Ο κατακερματισμός είναι πολύ αποδοτικός για *επιλογές (αναζητήσεις) ισότητας*

Εξωτερικός Κατακερματισμός

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

Εξωτερικός Κατακερματισμός

Συγκρούσεις - αλυσιδωτή σύνδεση - εγγραφές υπερχείλισης ανά κάδο

1. Ανάγνωση όλου του αρχείου (scan)

Έστω ότι διατηρούμε κάθε κάδο γεμάτο κατά 80% άρα ένα αρχείο με μέγεθος B blocks χρειάζεται 1.25 B blocks

$1.25 * B * (T_D + R * T_C)$

2. Αναζήτηση

Συνθήκη ισότητας και μόνο ένα block ανά κάδο: $T_D + R * C$

Αν συνθήκη περιοχής (διαστήματος): scan!

Περίληψη

Κόστος: μεταφορά blocks (I/O)

	Σωρός	Ταξινομημένο	Κατακερματισμένο
Ανάγνωση του αρχείου	B	B	1.25B
Αναζήτηση με συνθήκη ισότητας	0.5 B	logB	1
Αναζήτηση με συνθήκη περιοχής	B	logB + ταιριάσματα	1.25 B
Εισαγωγή	2	αναζήτηση + B	2
Διαγραφή	αναζήτηση + 1	αναζήτηση + B	αναζήτηση + 1

Ερωτήσεις;