Αποθηκεύοντας Δεδομένα: Δίσκοι και Αρχεία

Κεφάλαιο 9

"Yea, from the table of my memory
I'll wipe away all trivial fond records."
-- Shakespeare, *Hamlet*

Δίσκοι και Αρχεία

- Το DBMS αποθηκεύει την πληροφορία σε (“σκληρούς”) δίσκους
- Το γεγονός αυτό έχει επιπτώσεις στο σχεδιασμό των DBMS!
  - READ: μετάφερε δεδομένα από το δίσκο στην κύρια μνήμη (RAM)
  - WRITE: μετάφερε δεδομένα από τη RAM στο δίσκο
  - Και οι δυο παραπάνω λειτουργίες έχουν υψηλό κόστος σε σχέση με το κόστος των λειτουργιών στη RAM και πρέπει να οργανωθούν προσεκτικά!

Γιατί όχι όλα στη RAM;

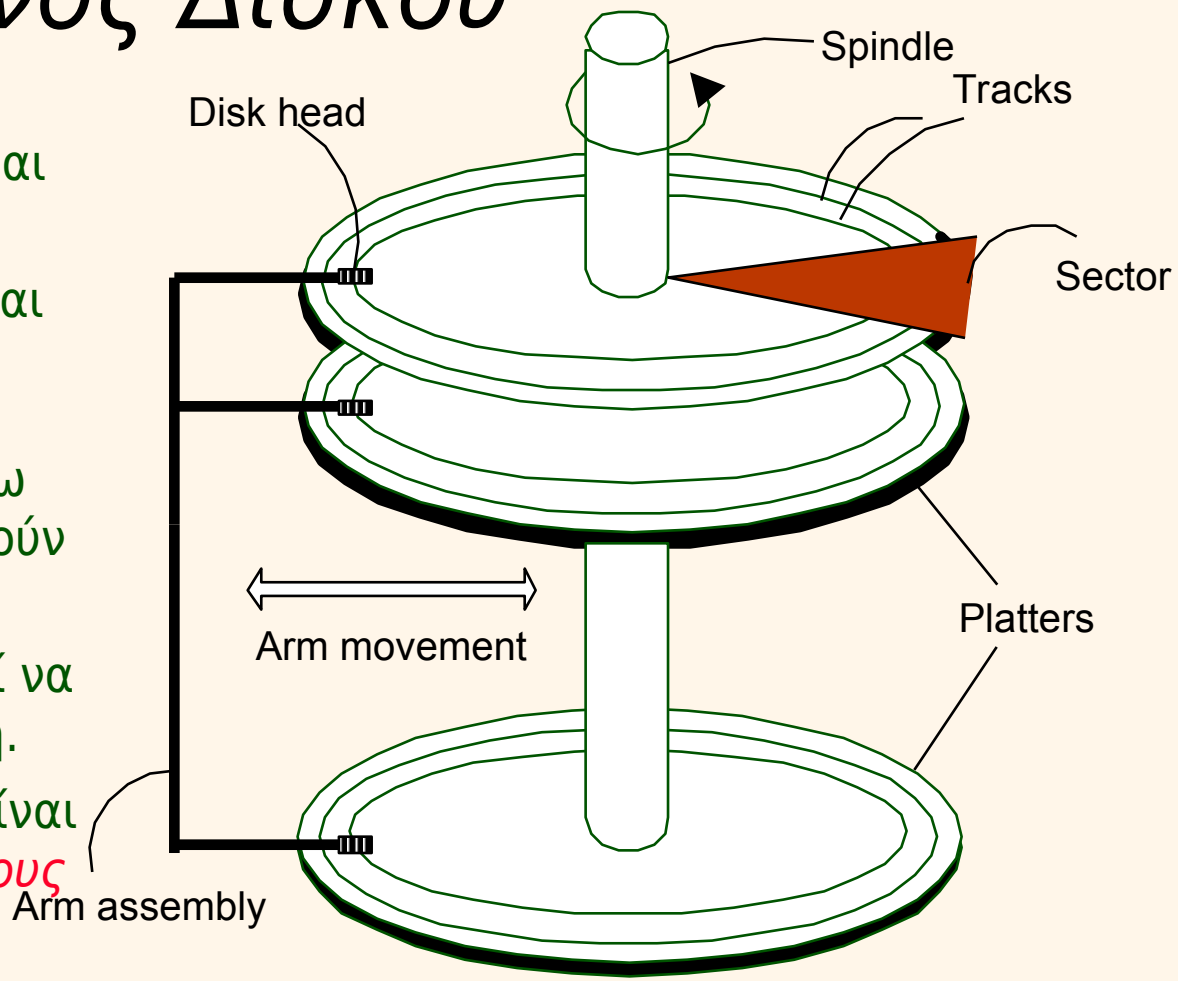
- *Υψηλό κόστος*.
  - κόστος δίσκων: 1981 $0,5M/GB 2011 ¢8/GB
  - κόστος RAM: 1982 $4M/GB 2009 $8/GB
- *Η RAM είναι μη-μόνιμη (volatile)*. Προφανώς, θέλουμε να "σώζονται" τα δεδομένα μας μεταξύ διαδοχικών εκτελέσεων του συστήματος
- Τυπική ιεραρχία αποθήκευσης:
  - Κύρια Μνήμη (RAM) για τα τρέχοντα δεδομένα (πρωτεύουσα)
  - Δίσκος για όλα τα δεδομένα της βάσης (δευτερεύουσα)
  - DVD/Ταινίες για αρχειοθέτηση παλιών εκδόσεων (τριτεύουσα)

Δίσκοι

- Η καλύτερη επιλογή δευτερεύουσας μνήμης
- Τα δεδομένα αποθηκεύονται και ανακτώνται σε μονάδες που ονομάζονται *μπλοκ δίσκου* ή *σελίδες*
- Αντίθετα με τη RAM, ο χρόνος ανάκτησης μιας σελίδας ποικίλει ανάλογα με την τοποθεσία της στο δίσκο
  - Συνεπώς, η σχετική τοποθέτηση των σελίδων στο δίσκο έχει μεγάλη σημασία στις επιδόσεις του DBMS!

Συστατικά ενός Δίσκου

- Τα *πλατώ* περιστρέφονται (έστω, 90rps).
- Ο *βραχίονας* μετακινείται για να τοποθετηθεί μια *κεφαλή* στο επιθυμητό *αυλάκι*. Τα αυλάκια κάτω από τις κεφαλές συνιστούν έναν (νοητό) *κύλινδρο*.
- Μόνο μια κεφαλή μπορεί να γράφει ανά πάσα στιγμή.
- Το *μέγεθος του μπλοκ* είναι πολλαπλάσιο του *μεγέθους του τομέα* (που είναι προκαθορισμένο).



Προσπέλαση σελίδας του δίσκου

- Χρόνος προσπέλασης (read/write) σε μπλοκ του δίσκου:
  - *χρόνος αναζήτησης (seek time)* (μετακίνηση του βραχίονα για την τοποθέτηση της κεφαλής στο κατάλληλο αυλάκι)
  - *καθυστέρηση περιστροφής (rotational delay)* (αναμονή ώστε να περάσει το μπλοκ κάτω από την κεφαλή)
  - *χρόνος μεταφοράς (transfer time)* (για τη μεταφορά των δεδομένων από/προς την επιφάνεια του δίσκου)
- Επικρατούν οι δυο πρώτοι χρόνοι:
  - Χρόνοι αναζήτησης: από 1 ως 20msec
  - Καθυστέρηση περιστροφής: από 0 ως 10msec
  - Χρόνοι μεταφοράς: από 1msec ανά σελίδα των 4KB
- Για χαμηλότερο κόστος I/O: μείωση των δύο χρόνων! Λύσεις μέσω Υλικού ή Λογισμικού;

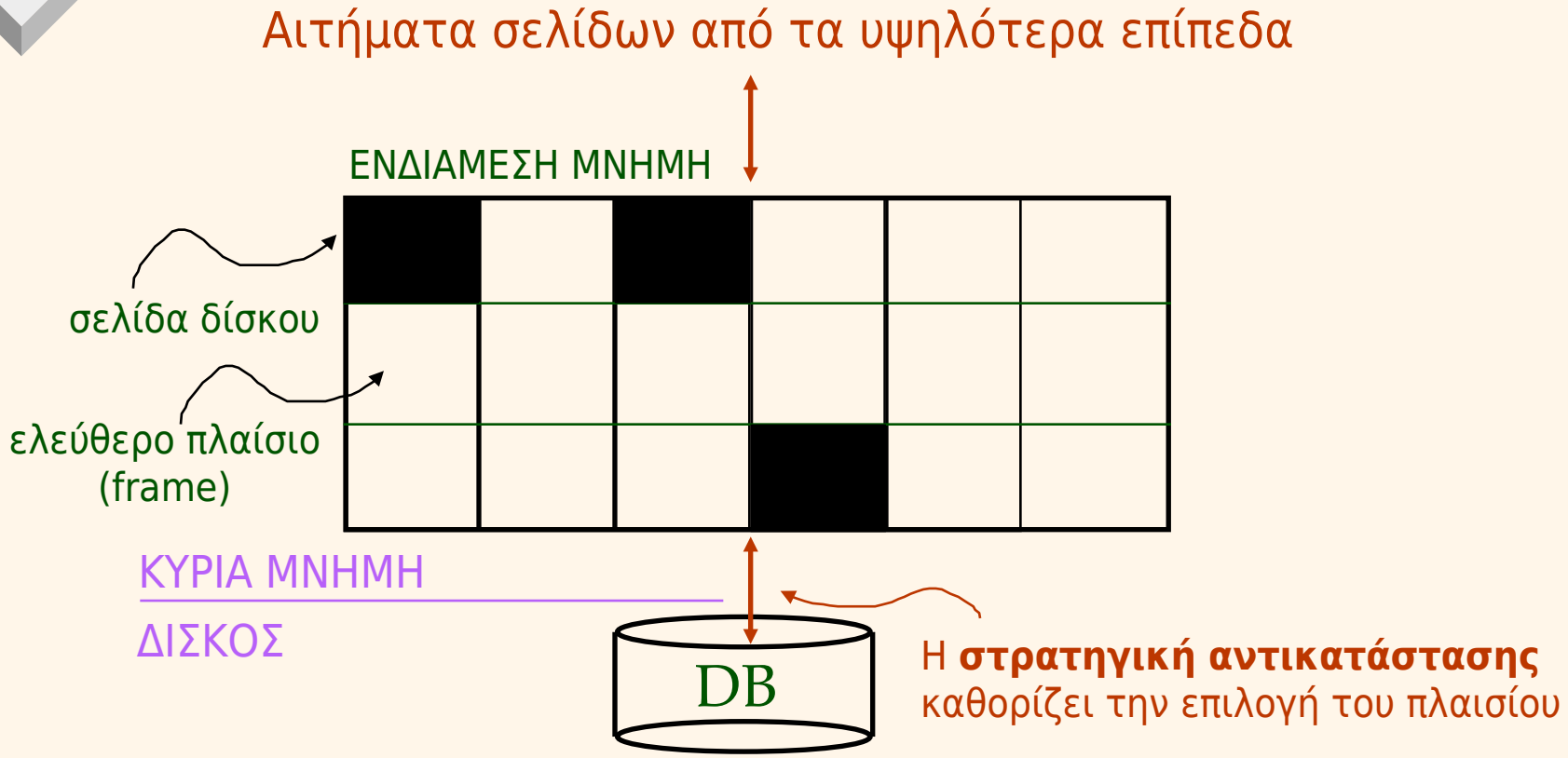
Τοποθέτηση των σελίδων στο δίσκο

- Η έννοια του '*επόμενου*' μπλοκ:
  - τα μπλοκ του ίδιου αυλακιού, ακολουθούμενα από
  - τα μπλοκ του ίδιου κυλίνδρου, ακολουθούμενα από
  - τα μπλοκ των γειτονικών κυλίνδρων
- Τα μπλοκ ενός αρχείου θα πρέπει να τοποθετούνται στο δίσκο με τη σειρά (κατά 'επόμενο') ώστε να ελαχιστοποιηθεί ο χρόνος αναζήτησης και η καθυστέρηση περιστροφής.
- Για μια σειριακή σάρωση (sequential scan), η *προφόρτωση (pre-fetching)* πολλών σελίδων τη φορά αποτελεί μεγάλο κέρδος!

Διαχείριση Δίσκου

- Το χαμηλότερο επίπεδο του λογισμικού DBMS, διαχειρίζεται το χώρο του δίσκου.
- Τα ανώτερα επίπεδα κάνουν κλήσεις στο επίπεδο αυτό:
  - για να δεσμεύσουν/αποδεσμεύσουν μια σελίδα
  - για να διαβάσουν/γράψουν μια σελίδα
- Ένα αίτημα για μια ακολουθία σελίδων πρέπει να ικανοποιείται με τη δέσμευση συνεχόμενων σελίδων στο δίσκο! Τα υψηλότερα επίπεδα δε χρειάζεται να γνωρίζουν πώς επιτυγχάνεται αυτό ή πώς γίνεται η διαχείριση του ελεύθερου χώρου.

Διαχείριση Ενδιάμεσης Μνήμης (1)



- *Τα δεδομένα πρέπει να είναι στη RAM για να μπορέσει να τα χρησιμοποιήσει το DBMS!*
- *Υπάρχει ένας πίνακας με ζεύγη <frame#, pageid>*

Όταν υπάρχει αίτημα για σελίδα...

- Αν η σελίδα δεν υπάρχει στην ενδιάμεση μνήμη:
  - Επιλέγεται ένα πλαίσιο για *αντικατάσταση*
  - Αν το πλαίσιο έχει τροποποιηθεί (dirty), γράφεται στο δίσκο
  - Διαβάζεται η σελίδα στο επιλεγμένο πλαίσιο
- *Καρφιτσώνεται (Pin)* η σελίδα και επιστρέφεται η διεύθυνσή της.
- *Αν τα αιτήματα μπορούν να προβλεφθούν (π.χ. σειριακές σαρώσεις) μπορεί να γίνει προφόρτωση πολλών σελίδων τη φορά!*

Διαχείριση Ενδιάμεσης Μνήμης (2)

- Αυτός που αιτήθηκε τη σελίδα πρέπει να την 'ξεκαρφιτσώσει' (unpin), και να δηλώσει αν η σελίδα έχει τροποποιηθεί:
  - για το σκοπό αυτό χρησιμοποιείται το *dirty* bit.
- Μπορεί να γίνουν πολλαπλά αιτήματα για μια σελίδα στην ενδιάμεση μνήμη,
  - χρησιμοποιείται ο *pin count*. Μια σελίδα είναι υποψήφια για αντικατάσταση iff *pin count* = 0.
- Ο έλεγχος ταυτοχρονισμού και η επαναφορά του συστήματος μπορεί να προκαλέσουν επιπλέον I/O κατά την επιλογή ενός πλαισίου για αντικατάσταση (αυτά θα τα πούμε αργότερα).

Στρατηγικές Αντικατάστασης

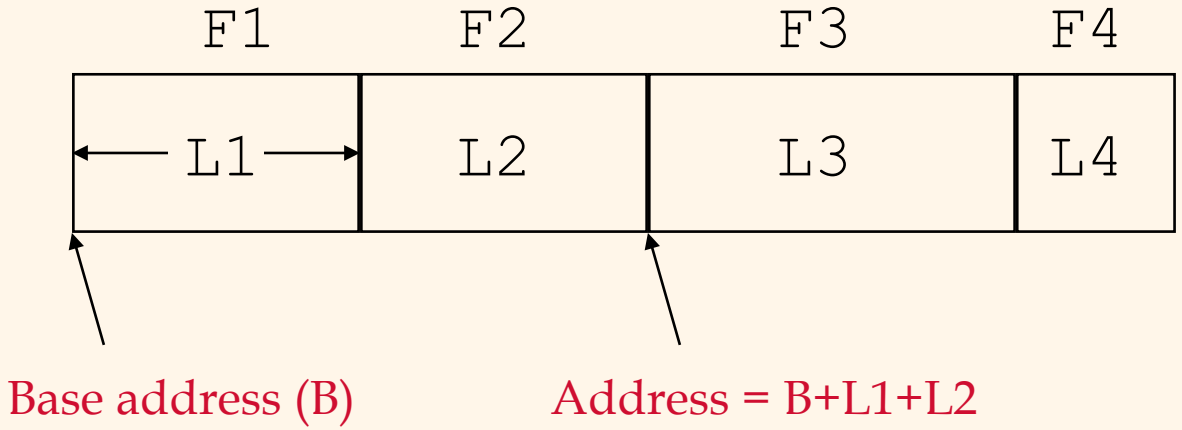
- Ένα πλαίσιο επιλέγεται για αντικατάσταση σύμφωνα με κάποια *στρατηγική αντικατάστασης:*
  - Least-recently-used (LRU), Clock, MRU etc.
- Η στρατηγική μπορεί να έχει μεγάλη επίδραση στο πλήθος των I/O – αυτό εξαρτάται από το *access pattern*.
- *Σειριακή πλημμύρα*: δυσάρεστη κατάσταση προκαλούμενη από LRU + επαναλαμβανόμενες σειριακές σαρώσεις.
  - # buffer frames < # pages in file σημαίνει πως κάθε αίτημα σελίδας προκαλεί μια I/O. Στην περίπτωση αυτή η MRU είναι προτιμότερη (αλλά, βέβαια, όχι σε όλες τις περιπτώσεις).

DBMS vs. OS File System

Το OS διαχειρίζεται το δίσκο και την ενδιάμεση μνήμη: γιατί να μην αφήσουμε το OS να κάνει τη διαχείριση;

- Θέματα μεταφερσιμότητας: τα OS διαφέρουν
- Διάφοροι περιορισμοί: π.χ., τα αρχεία δεν μπορούν να επικαλύπτουν δίσκους.
- Η διαχείριση της ενδιάμεσης μνήμης σε ένα DBMS απαιτεί:
  - καρφίτσωμα (pin) μιας σελίδας στην ενδιάμεση μνήμη, εξαναγκασμό μιας σελίδας στο δίσκο (απαραίτητο για την υλοποίηση του ταυτοχρονισμού και της επαναφοράς του συστήματος),
  - τροποποίηση της *στρατηγικής αντικατάστασης*, και προφόρτωση σελίδων ανάλογα με τα μοτίβα πρόσβασης των τυπικών πράξεων.

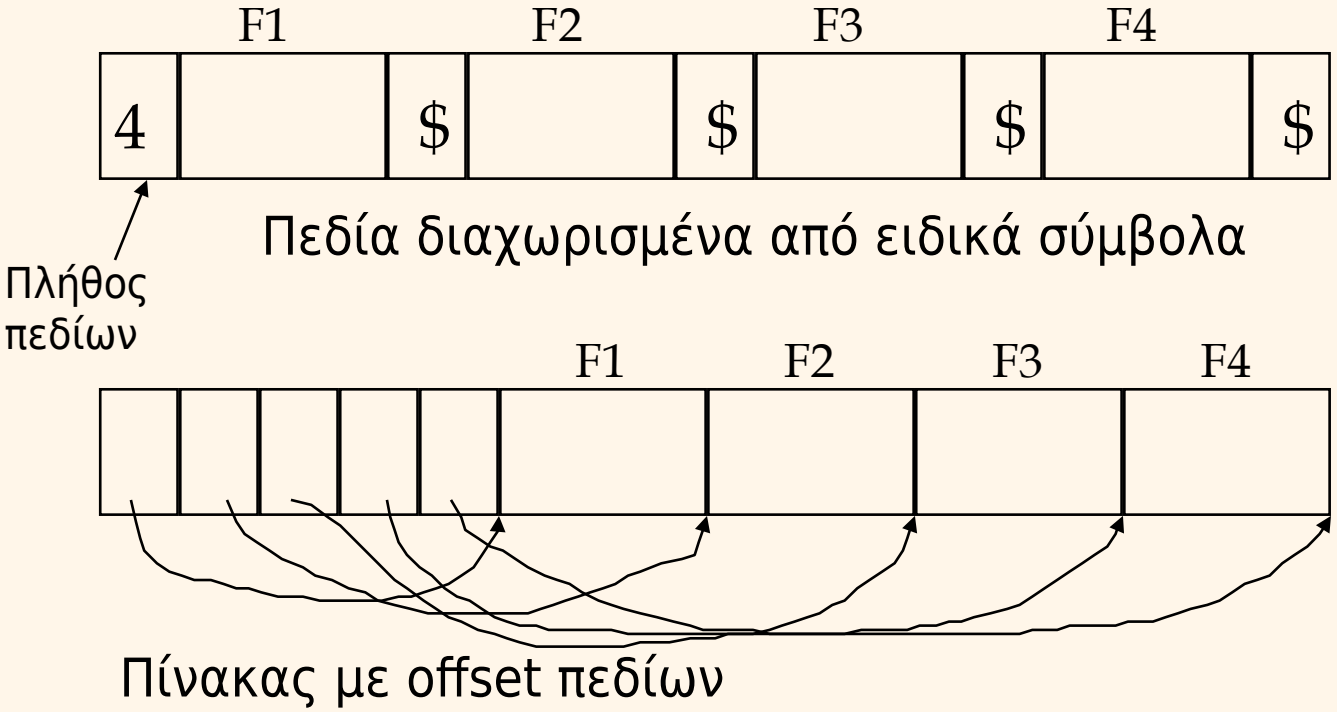
Εγγραφές σταθερού μήκους



- Η πληροφορία σχετικά με τους τύπους των πεδίων είναι η ίδια για όλες τις εγγραφές του αρχείου – αποθηκεύεται στους *καταλόγους του συστήματος.*
- Για αναζήτηση του *i'th* πεδίου απαιτείται σάρωση της εγγραφής.

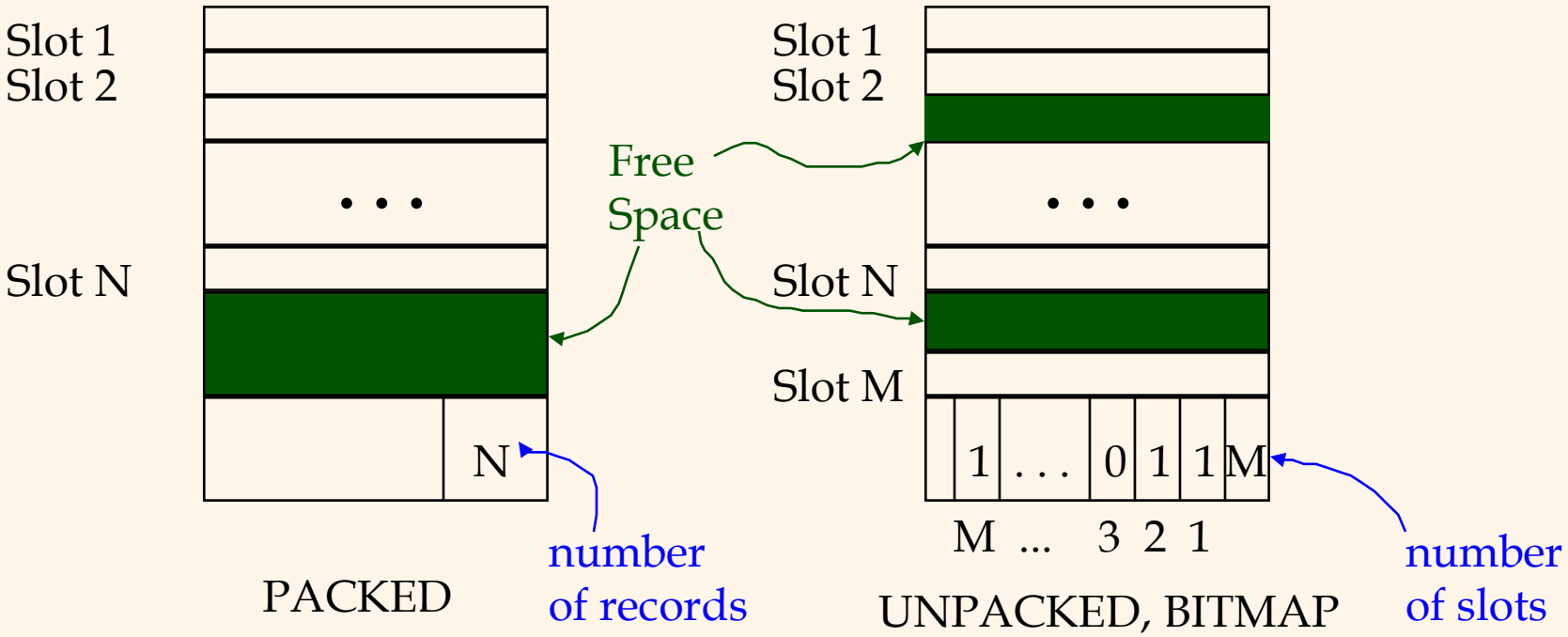
Εγγραφές μεταβλητού μήκους

- 2 εναλλακτικές μορφοποιήσεις (σταθερό πλήθος πεδίων):



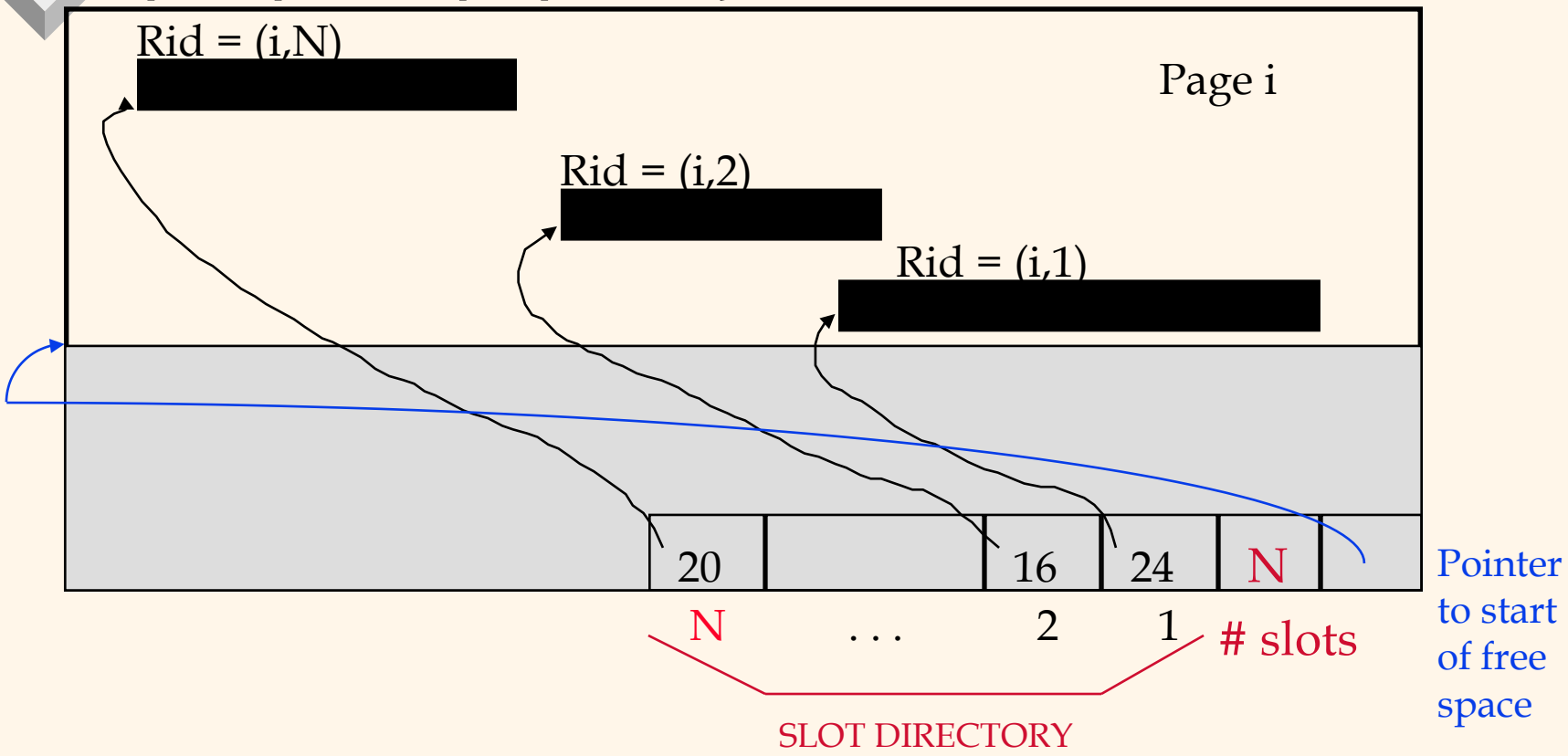
  - Η δεύτερη προσφέρει απευθείας πρόσβαση στο i'th πεδίο, αποτελεσματική αποθήκευση των *nulls,* μικρότερο κατάλογο.

Μορφοποίηση σελίδων για εγγραφές σταθερού μήκους



- *rid = <page id, slot #>. Στην πρώτη εναλλακτική, η μετακίνηση εγγραφής για τη διαχείριση του ελεύθερου χώρου αλλάζει το rid – κάτι τέτοιο μπορεί να μην είναι αποδεκτό.*

Μορφοποίηση σελίδων για εγγραφές μεταβλητού μήκους



- *Οι εγγραφές μπορούν να μετακινηθούν χωρίς αλλαγή του rid – άρα η μορφοποίηση είναι κατάλληλη και για εγγραφές σταθερού μήκους.*

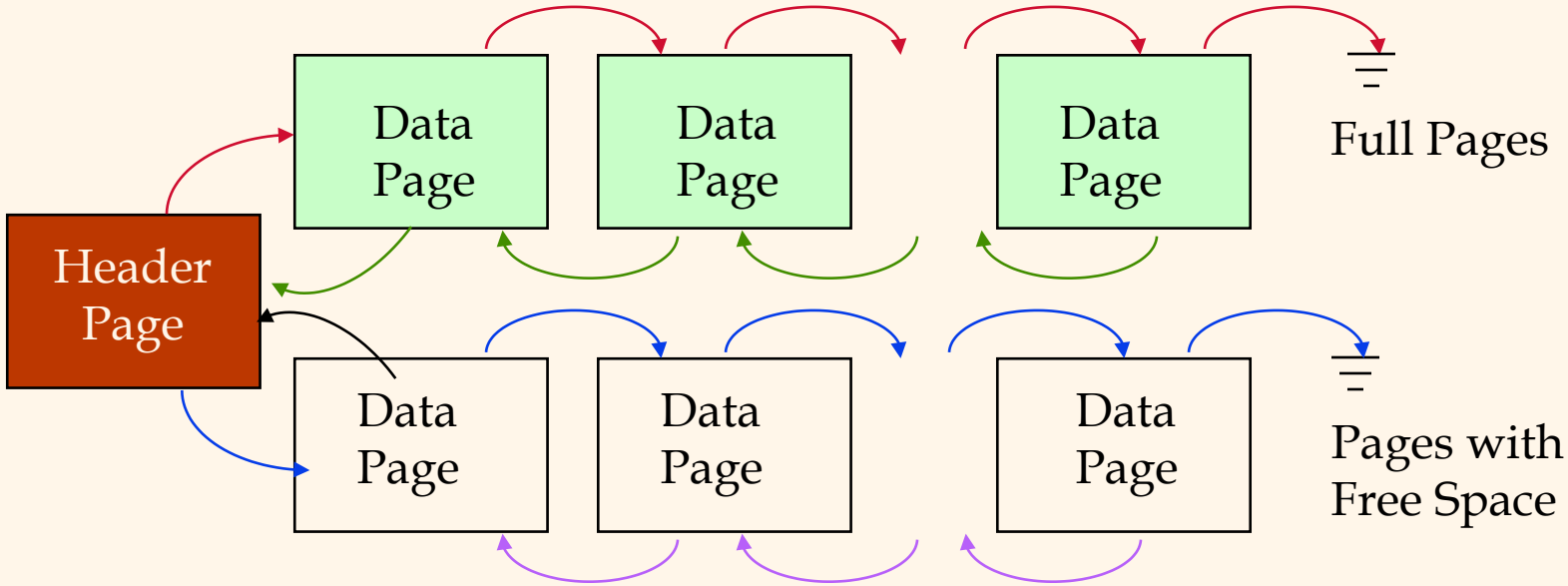
Αρχεία εγγραφών

- Οι σελίδες ή τα μπλοκ είναι OK για I/O, αλλά τα υψηλότερα επίπεδα του DBMS λειτουργούν πάνω σε *εγγραφές* και *αρχεία εγγραφών*.
- ΑΡΧΕΙΟ (FILE): Μια συλλογή σελίδων, όπου η κάθε σελίδα αποτελεί για συλλογή εγγραφών. Πρέπει να υποστηρίζει:
  - εισαγωγή/διαγραφή/τροποποίηση εγγραφής
  - ανάγνωση μιας συγκεκριμένης εγγραφής (που προσδιορίζεται από το *record id*)
  - σάρωση όλων των εγγραφών (πιθανώς με την εφαρμογή συνθηκών ως προς τις εγγραφές που πρέπει να ανακτηθούν)

Μη-ταξινομημένα Αρχεία (Σωρού)

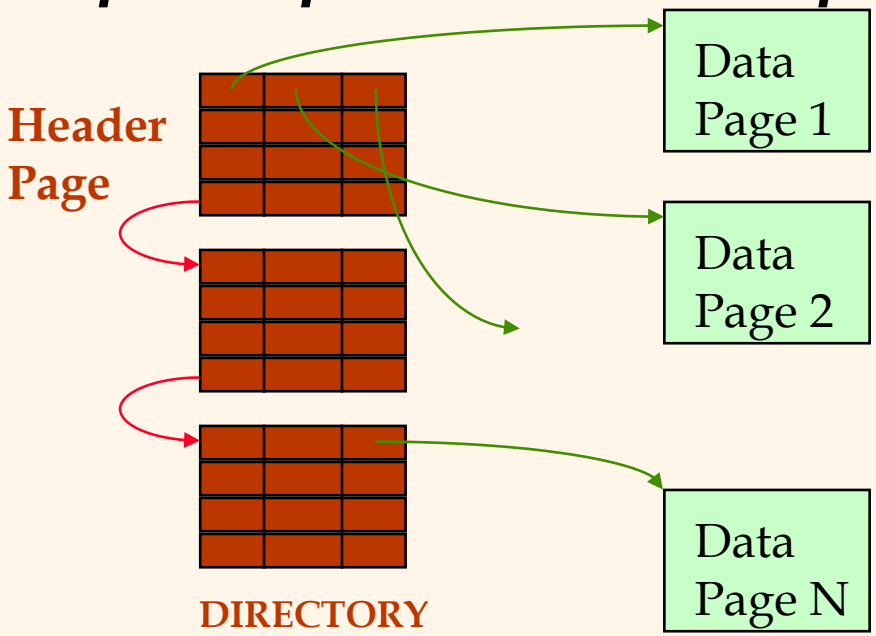
- Η απλούστερη δομή αρχείου – περιέχει εγγραφές χωρίς κάποια συγκεκριμένη ταξινόμηση.
- Σελίδες δεσμεύονται ή αποδεσμεύονται καθώς το αρχείο επεκτείνεται ή συρρικνώνεται.
- Για την υποστήριξη των λειτουργιών σε επίπεδο εγγραφής, πρέπει να:
  - καταγράφουμε τις *σελίδες* ενός αρχείου
  - καταγράφουμε τον *ελεύθερο χώρο* μέσα στις σελίδες
  - καταγράφουμε τις *εγγραφές* μέσα σε μια σελίδα
- Υπάρχουν πολλές εναλλακτικές προσεγγίσεις.

Αρχείο σωρού ως λίστα



- Απαιτείται η αποθήκευση του header page id και του ονόματος του αρχείου σωρού.
- Κάθε σελίδα περιέχει 2 'pointers' και δεδομένα.

Αρχείο σωρού με κατάλογο σελίδων



- Η καταχώρηση για κάθε σελίδα μπορεί να περιέχει το πλήθος των ελεύθερων bytes στη σελίδα.
- Ο κατάλογος είναι μια συλλογή σελίδων – η υλοποίηση ως συνδεδεμένη λίστα είναι ένας τρόπος υλοποίησης.
  - *Είναι πολύ μικρότερος από τη συνδεδεμένη λίστα όλων των σελίδων του αρχείου σωρού!*

Ευρετήρια

- Ένα αρχείο σωρού επιτρέπει την ανάκτηση εγγραφών:
  - δίνοντας το *rid*, *ή*
  - *σαρώνοντας σειριακά όλες τις εγγραφές*
- Μερικές φορές, θέλουμε να ανακτούμε εγγραφές προσιορίζοντας τις τιμές ενός ή περισσοτέρων πεδίων:
  - Βρες όλους τους φοιτητές του τμήματος Εφ. Πληροφορικής
  - Βρες όλους τους φοιτητές με βαθμό > 8
- Τα Ευρετήρια (Indexes) είναι δομές αρχείων που επιτρέπουν την αποτελεσματική απάντηση τέτοιου είδους αιτημάτων βάσει τιμών.

Κατάλογοι του Συστήματος

- Για κάθε ευρετήριο:
  - Δομή (π.χ., B+ tree) και πεδία αναζήτησης
- Για κάθε πίνακα (σχέση):
  - όνομα, όνομα αρχείου, δομή αρχείου (π.χ., αρχείο σωρού)
  - όνομα πεδίου και τύπος, για κάθε πεδίο
  - όνομα ευρετηρίου, για κάθε ευρετήριο σε πεδία του πίνακα
  - περιορισμοί ακεραιότητας
- Για κάθε όψη:
  - όνομα και ορισμός
- Και επιπλέον, στατιστικά, δικαιώματα πρόσβασης, μέγεθος ενδιάμεσης μνήμης, κλπ.
- *Οι κατάλογοι αποθηκεύονται επίσης ως πίνακες*!

Attr_Cat(attr_name, rel_name, type, position)

attr_name	rel_name	type	position
attr_name	Attribute_Cat	string	1
rel_name	Attribute_Cat	string	2
type	Attribute_Cat	string	3
position	Attribute_Cat	integer	4
sid	Students	string	1
name	Students	string	2
login	Students	string	3
age	Students	integer	4
gpa	Students	real	5
fid	Faculty	string	1
fname	Faculty	string	2
sal	Faculty	real	3