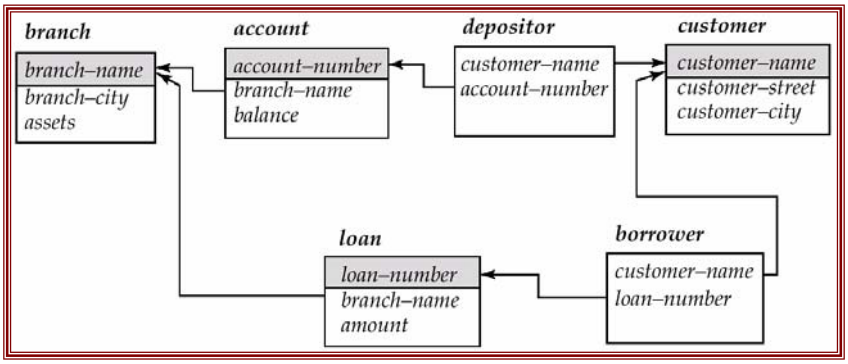
Οργάνωση Αρχείων & Ευρετήρια

- Οργάνωση Αρχείων
  - Αρχεία σωρού
  - Διατεταγμένα αρχεία
  - Αρχεία κατακερματισμού
- Ευρετήρια
  - Ταξινομημένα ευρετήρια
  - B^+-δένδρα
  - Ευρετήρια κατακερματισμού

Βασική πηγή διαφανειών: Silberschatz et al., "Database System Concepts", 4/e
Εργαστήριο Πληροφοριακών Συστημάτων, Παν/μιο Πειραιώς (http://infolab.cs.unipi.gr/)
έκδοση: Φεβ. 2010

Διάγραμμα Σχεσιακού σχήματος

- Βάση δεδομένων τράπεζας
  (πελάτες – υποκαταστήματα – καταθετικοί λογαριασμοί – δάνεια)

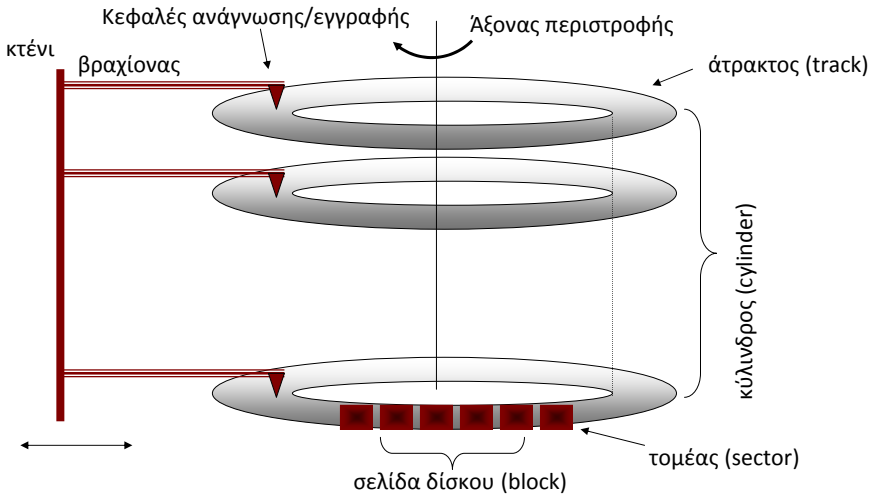


- **Οργάνωση Αρχείων**
  - **Αρχεία σωρού**
  - **Διατεταγμένα αρχεία**
  - **Αρχεία κατακερματισμού**
- Ευρετήρια
  - Ταξινομημένα ευρετήρια
  - B^+-δένδρα
  - Ευρετήρια κατακερματισμού

Οργάνωση αρχείων

- Η βάση δεδομένων αποθηκεύεται ως μια συλλογή από **αρχεία**. Κάθε αρχείο είναι μια ακολουθία από **εγγραφές**. Μια εγγραφή είναι μια ακολουθία από πεδία.
- Η βάση δεδομένων πρέπει να αποθηκευτεί σε κάποιο αποθηκευτικό μέσο
- Ιεραρχία μέσων αποθήκευσης
  - **πρωτεύουσα αποθήκευση** (primary storage): κύρια μνήμη, κρυφή μνήμη (cache) - γρήγορη προσπέλαση (nsecs) αλλά περιορισμένη χωρητικότητα (Mbytes)
  - **δευτερεύουσα αποθήκευση** (secondary storage): μαγνητικοί δίσκοι, ταινίες - αργή προσπέλαση (msecs) αλλά μεγάλη χωρητικότητα (TBytes)
  - Οι περισσότερες βάσεις δεδομένων αποθηκεύονται σε δευτερεύουσες αποθηκευτικές μονάδες (κυρίως σε δίσκους)
    - πολύ μεγάλες ΒΔ $\Rightarrow$ μεγάλο κόστος
    - μόνιμη αποθήκευση (non-volatile storage)

Μαγνητικοί δίσκοι (1)



Μαγνητικοί δίσκοι (2)

- **χρόνος εντοπισμού** (seek time): χρόνος για να τοποθετηθεί η κεφαλή στη σωστή άτρακτο
- **χρόνος περιστροφής** (rotational delay ή latency): χρόνος 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

Seagate Barracuda 180 hard disk
- size: 180 GBytes
- interface: Ultra 160 SCSI, Fibre Channel
- spindle rotation speed: 7200 RPM
- latency: 4.17 ms
- average read/write seek time: 7.4/8.2 ms
- internal formatted data rate: 26-47 Mbytes/s
- buffer size: 4 (optionally 16) Mb

Παραδοχές

- Η μονάδα μεταφοράς μεταξύ δίσκου και μνήμης είναι ένα block δίσκου (με μέγεθος b bytes)
- Κάθε αρχείο έχει εγγραφές ενός τύπου μόνο
- Το μέγεθος των εγγραφών είναι σταθερό (με μέγεθος r bytes)
- Οι εγγραφές δεν επιτρέπεται να διασχίζουν τα όρια ενός block

Παράγοντας ομαδοποίησης (blocking factor):

$$bfr = \lfloor (b / r) \rfloor$$

Αριθμός blocks B για την αποθήκευση ενός αρχείου R εγγραφών:

$$B = \lceil (R / bfr) \rceil$$

Εγγραφές σταθερού μήκους

- Απλή προσέγγιση:
  - Αποθηκεύουμε την i-οστή εγγραφή ξεκινώντας από το byte n∗(i–1), όπου n είναι το μέγεθος κάθε εγγραφής.
  - Υπέρ: Η προσπέλαση των εγγραφών είναι απλή
  - Κατά: Η διαγραφή μιας εγγραφής είναι προβληματική

record 0	A-102	Perryridge	400
record 1	A-305	Round Hill	350
record 2	A-215	Mianus	700
record 3	A-101	Downtown	500
record 4	A-222	Redwood	700
record 5	A-201	Perryridge	900
record 6	A-217	Brighton	750
record 7	A-110	Downtown	600
record 8	A-218	Perryridge	700

Ελεύθερες λίστες

- Αποθηκεύουμε τη διεύθυνση της πρώτης διαγραμμένης εγγραφής στην επικεφαλίδα του αρχείου.
- Χρησιμοποιούμε αυτήν την πρώτη εγγραφή για να αποθηκεύσουμε τη διεύθυνση της δεύτερης διαγραμμένης εγγραφής κ.ο.κ.
- Μπορούμε να θεωρήσουμε αυτές τις αποθηκευμένες διευθύνσεις ως δείκτες, αφού δείχνουν στη θέση που βρίσκεται μια εγγραφή.

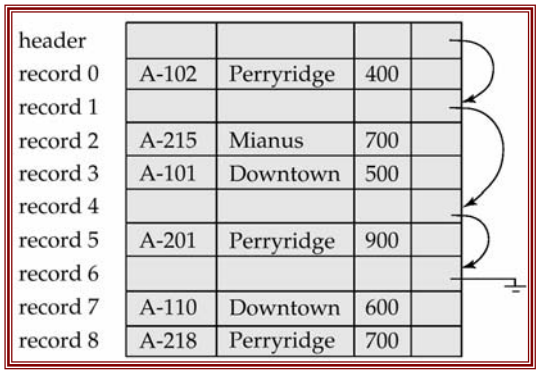
header				
record 0	A-102	Perryridge	400	
record 1				
record 2	A-215	Mianus	700	
record 3	A-101	Downtown	500	
record 4				
record 5	A-201	Perryridge	900	
record 6				
record 7	A-110	Downtown	600	
record 8	A-218	Perryridge	700	

Οργάνωση των εγγραφών σε αρχεία

- **Αρχεία σωρού** – μια εγγραφή μπορεί να τοποθετηθεί οπουδήποτε υπάρχει χώρος στο αρχείο.
- **Διατεταγμένα αρχεία** – αποθήκευση των εγγραφών βάσει της τιμής του κλειδιού διάταξης κάθε εγγραφής.
- **Αρχεία κατακερματισμού** – μια συνάρτηση κατακερματισμού υπολογίζεται σε κάποιο χαρακτηριστικό κάθε εγγραφής. Το αποτέλεσμα καθορίζει σε ποιο block του αρχείου πρέπει να αποθηκευθεί η εγγραφή.

Αρχεία σωρού

- **Αρχείο Σωρού** (heap file ή pile file): Οι εγγραφές τοποθετούνται στο αρχείο με τη σειρά που εισάγονται
- Κόστος εισαγωγής εγγραφής:
  - 2 blocks (1 ανάγνωση + 1 εγγραφή)
- (μέσο) κόστος αναζήτησης εγγραφών (ερώτηση ταυτότητας)
  - B/2 blocks (κλειδί)
  - B blocks (όχι κλειδί)



Διατεταγμένα αρχεία (1)

- Οι εγγραφές στο αρχείο είναι διατεταγμένες βάσει ενός κλειδιού διάταξης.
  - **Κλειδί διάταξης** (ordering key): χαρακτηριστικό (ή σύνολο χαρακτηριστικών) που χρησιμοποιείται για την αναζήτηση εγγραφών σε ένα αρχείο.
    - Π.χ., *<branch-name, account-number>*
    - Δεν ταυτίζεται απαραίτητα με το πρωτεύον κλειδί

Διατεταγμένα αρχεία (2)

- Αναζήτηση:
  - Ερώτηση ταυτότητας σε πεδίο άλλο από το κλειδί διάταξης: **σειριακή αναζήτηση**
  - Ερώτηση ταυτότητας στο κλειδί διάταξης: **δυαδική αναζήτηση**
- Κόστος αναζήτησης (ερώτηση ταυτότητας στο κλειδί διάταξης) =

  κόστος αναζήτησης στο κύριο αρχείο ($\log_2 B$ blocks) + κόστος αναζήτησης στο αρχείο υπερχείλισης, μεγέθους B' (B'/2 blocks)
  - παραδοχή: B' << B

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

Διατεταγμένα αρχεία (3)

- Εισαγωγή – 2 εναλλακτικά:
  - Διατήρηση αχρησιμοποίητου χώρου σε κάθε block
  - Δημιουργία προσωρινού μη διατεταγμένου αρχείου (**αρχείο υπερχείλισης**)
- Και στις 2 περιπτώσεις προκύπτει η ανάγκη περιοδικής αναδιοργάνωσης
- Κόστος εισαγωγής εγγραφής =

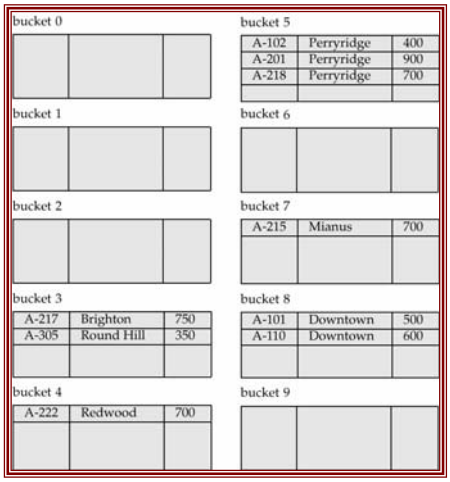
  Κόστος αναζήτησης (για εντοπισμό της κατάλληλης θέσης)

  + κόστος εισαγωγής στο αρχείο υπερχείλισης (2 blocks)

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	
A-888	North Town	800	

Αρχεία Κατακερματισμού

- Ένας **κάδος** (bucket) είναι μια μονάδα αποθήκευσης που περιέχει μια ή περισσότερες εγγραφές (συνήθως κάδος = block).
- Σε ένα **αρχείο κατακερματισμού** (hashing) βρίσκουμε τον κάδο στον οποίο περιέχεται μια εγγραφή απευθείας από την τιμή του κλειδιού διάταξης της εγγραφής χρησιμοποιώντας μια **συνάρτηση κατακερματισμού** (hashing function).



Στατικός κατακερματισμός

- Στο **στατικό κατακερματισμό** (static hashing) δεσμεύεται εξαρχής ένας αριθμός από κάδους.
- Η συνάρτηση κατακερματισμού **h(K)** είναι μια **συνάρτηση** από το σύνολο των τιμών του κλειδιού διάταξης K στο σύνολο των διευθύνσεων των κάδων B.
- Η συνάρτηση κατακερματισμού χρησιμοποιείται για την αναζήτηση, εισαγωγή αλλά και τη διαγραφή εγγραφών.
- Εγγραφές με διαφορετικές τιμές του κλειδιού διάταξης μπορεί να τοποθετηθούν στον ίδιο κάδο. Επομένως, όλος ο κάδος πρέπει να **σαρωθεί σειριακά** για την εύρεση μιας εγγραφής.

Παράδειγμα στατικού κατακερματισμού

- Κατακερματισμός πάνω στο branch-name
- 10 κάδοι (4 εγγραφές / κάδο)
- Η συνάρτηση κατακερ-ματισμού επιστρέφει μια τιμή στο [0..9]
- Παράδειγμα:

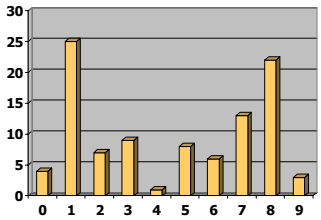
  h(Perryridge) = 5

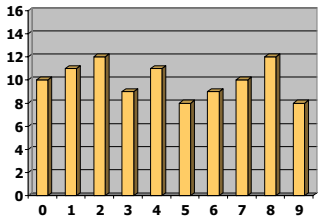
  h(Round Hill) = 3

bucket 0		
bucket 1		
bucket 2		
bucket 3		
A-217	Brighton	750
A-305	Round Hill	350
bucket 4		
A-222	Redwood	700
bucket 5		
A-102	Perryridge	400
A-201	Perryridge	900
A-218	Perryridge	700
bucket 6		
bucket 7		
A-215	Mianus	700
bucket 8		
A-101	Downtown	500
A-110	Downtown	600
bucket 9		

Συναρτήσεις κατακερματισμού

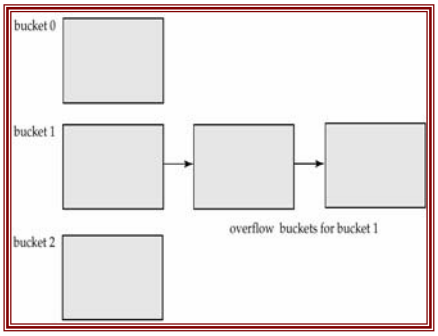
- Η **χειρότερη** συνάρτηση κατακερματισμού αντιστοιχεί όλες τις τιμές του κλειδιού διάταξης στον ίδιο κάδο.
- Μια **ιδανική** συνάρτηση κατακερματισμού είναι:
  - **ομοιόμορφη**, σε κάθε κάδο ανατίθεται ο ίδιος αριθμός από τιμές του κλειδιού διάταξης από το σύνολο όλων των πιθανών τιμών.
  - **τυχαία**, κάθε κάδος έχει τον ίδιο αριθμό από εγγραφές ανεξάρτητα από την **πραγματική κατανομή** των τιμών του κλειδιού διάταξης στο αρχείο.





Επίλυση συγκρούσεων

- **Σύγκρουση** (collision) συμβαίνει όταν μια νέα εγγραφή κατακερματίζεται σε έναν ήδη γεμάτο κάδο λόγω:
  - ανεπάρκειας των κάδων (λίγοι σε σχέση με τον όγκο της ΒΔ ή
  - ασυμμετρίας στην κατανομή των εγγραφών.
- Επίλυση συγκρούσεων:
  - **Ανοιχτή Διευθυνσιοδότηση** (open addressing): τοποθέτηση στην επόμενη κενή θέση
  - **Αλυσιδωτή Σύνδεση** (chaining): συνδεδεμένη λίστα με κάδους υπερχείλισης (overflow buckets)
  - **Πολλαπλός Κατακερματισμός** (multiple hashing): εφαρμογή δεύτερης συνάρτησης κατακ/μού



Μειονεκτήματα στατικού κατακερματισμού

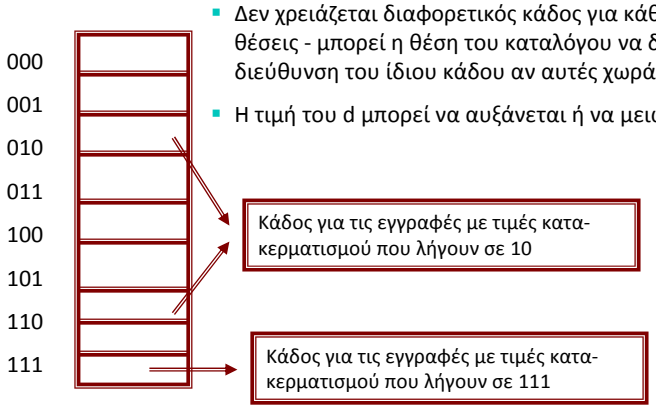
- Το σύνολο των διευθύνσεων κάδων είναι προκαθορισμένο.
  - Οι βάσεις δεδομένων μεγαλώνουν με το χρόνο. Αν ο αρχικός αριθμός κάδων είναι πολύ μικρός, η απόδοση θα μειωθεί λόγω των αλυσίδων υπερχείλισης.
  - Έστω και εάν είναι προβλέψιμο το μελλοντικό μέγεθος των αρχείων και προσδιοριστεί ανάλογα ο αριθμός των κάδων, στην αρχή θα σπαταλιέται άσκοπα σημαντικό ποσό χώρου.
  - Όταν συρρικνώνεται η βάση δεδομένων, πάλι υπάρχει σπατάλη χώρου.
  - Μια λύση είναι η **περιοδική αναδιοργάνωση** του αρχείου με μια νέα συνάρτηση κατακερματισμού, αλλά αυτή η διαδικασία είναι πολύ ακριβή (σε χώρο και χρόνο).
- Αυτά τα προβλήματα αποφεύγονται με τεχνικές που επιτρέπουν να τροποποιείται δυναμικά ο αριθμός των κάδων
  - **επεκτάσιμος κατακερματισμός** (extensible hashing).

Επεκτάσιμος κατακερματισμός (1)

- Δυαδική αναπαράσταση του αποτελέσματος της συνάρτησης κατακερματισμού, δηλαδή ως μια ακολουθία δυαδικών ψηφίων
- Κατανομή εγγραφών με βάση την τιμή των αρχικών ψηφίων
- Διαδικασία:
  - Το αρχείο ξεκινά με ένα μόνο κάδο
  - Μόλις γεμίσει ένας κάδος διασπάται σε δύο κάδους (0 και 1) με βάση την τιμή του λιγότερο σημαντικού δυαδικού ψηφίου (LSB) των τιμών κατακερματισμού
  - Νέα υπερχείλιση ενός κάδου οδηγεί σε διάσπασή του με βάση το αμέσως μεγαλύτερο σε σημασία δυαδικό ψηφίο, κοκ.
- Έτσι δημιουργείται ένας κατάλογος (directory) με τιμές κατακερματισμού (σε δυαδική μορφή) και διευθύνσεις κάδων
  - ο κατάλογος διατηρείται στη μνήμη, εκτός αν είναι μεγάλος

Επεκτάσιμος κατακερματισμός (2)

- Τα τελευταία d ψηφία της τιμής κατακερματισμού χρησιμοποιούνται ως δείκτης στον κατάλογο



- Δεν χρειάζεται διαφορετικός κάδος για κάθε μία από τις 2^d θέσεις - μπορεί η θέση του καταλόγου να δείχνει στη διεύθυνση του ίδιου κάδου αν αυτές χωράνε σε ένα κάδο
- Η τιμή του d μπορεί να αυξάνεται ή να μειώνεται

- Οργάνωση Αρχείων
  - Αρχεία σωρού
  - Διατεταγμένα αρχεία
  - Αρχεία κατακερματισμού
- **Ευρετήρια**
  - **Ταξινομημένα ευρετήρια**
  - **B^+-δένδρα**
  - **Ευρετήρια κατακερματισμού**

Ευρετήρια - Βασικά θέματα

- Οι μηχανισμοί δεικτοδότησης – ευρετήρια χρησιμοποιούνται για να επιταχύνουν την πρόσβαση σε επιθυμητά δεδομένα.
  - Π.χ., κατάλογος συγγραφέων στη βιβλιοθήκη
- Ένα **ευρετήριο** (index) αποτελείται από εγγραφές της μορφής:

κλειδί διάταξης	δείκτης

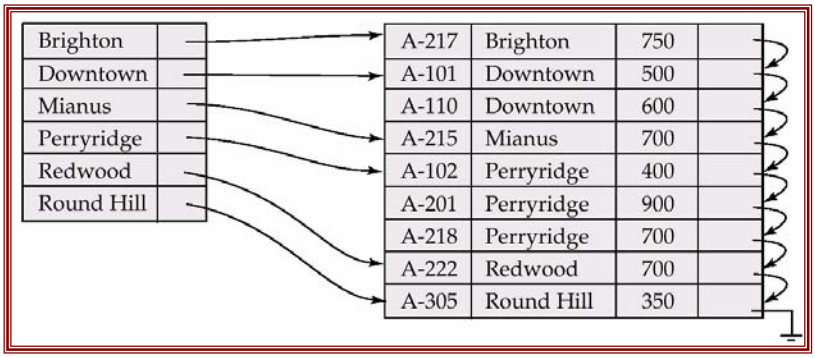
- Τα ευρετήρια συνήθως είναι αρκετά μικρότερα από το αρχείο που δεικτοδοτούν.
- Δυο βασικά είδη ευρετηρίων:
  - **διατεταγμένα ευρετήρια**: ευρετήρια που διατηρούν τη διάταξη των τιμών του πεδίου διάταξης
  - **ευρετήρια κατακερματισμού:** μηχανισμός αντίστοιχος με αυτόν στα αρχεία κατακερματισμού.

Διατεταγμένα ευρετήρια

- Περιέχουν εγγραφές

  <τιμή κλειδιού διάταξης, δείκτης στο κύριο αρχείο>

  για κάθε τιμή του κλειδιού διάταξης που εμφανίζεται στο αρχείο.



Ενημέρωση ευρετηρίου

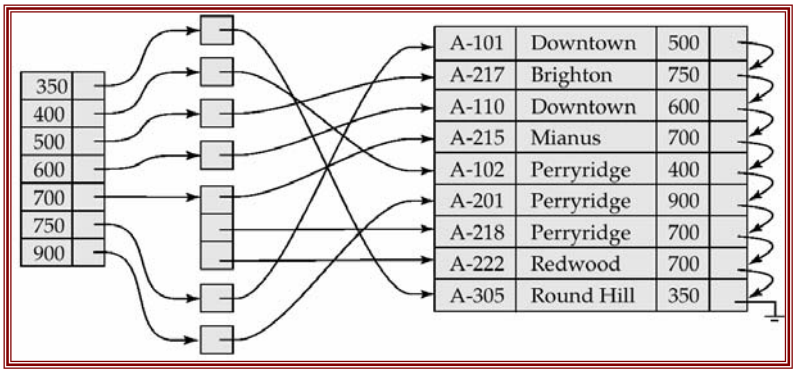
- **Εισαγωγή εγγραφής**: Πραγματοποιούμε μια αναζήτηση χρησιμοποιώντας την τιμή του κλειδιού διάταξης που έχει η εγγραφή που πρόκειται να εισαχθεί. Εάν το κλειδί διάταξης δεν υπάρχει στο ευρετήριο, το εισάγουμε.
- **Διαγραφή εγγραφής**: Εάν η διαγραφείσα εγγραφή ήταν η μοναδική στο αρχείο με τη συγκεκριμένη τιμή του κλειδιού διάταξης, το κλειδί διάταξης διαγράφεται επίσης και από το ευρετήριο.
  - η διαγραφή του κλειδιού διάταξης είναι παρόμοια με τη διαγραφή μιας εγγραφής του αρχείου.

Δευτερεύοντα ευρετήρια

- Πολλές φορές θέλουμε να βρούμε όλες τις εγγραφές, των οποίων οι τιμές σε ένα συγκεκριμένο πεδίο (το οποίο δεν είναι το κλειδί διάταξης) ικανοποιούν κάποια συνθήκη.
  - Παράδειγμα: Στη σχέση *account*, θέλουμε να βρούμε όλους τους λογαριασμούς με ένα συγκεκριμένο υπόλοιπο (ή εύρος υπολοίπων).
- Μπορούμε να έχουμε ένα **δευτερεύον ευρετήριο** (secondary index) με μια εγγραφή για κάθε τιμή του πεδίου. Κάθε εγγραφή του ευρετηρίου δείχνει σε ένα κάδο, ο οποίος περιέχει δείκτες σε όλες τις πραγματικές εγγραφές με τη συγκεκριμένη τιμή του πεδίου.

Παράδειγμα δευτερεύοντος ευρετηρίου

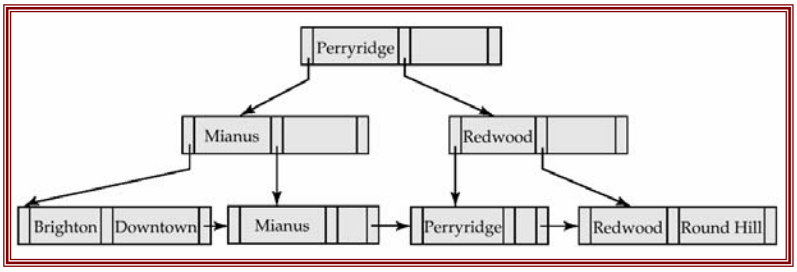
πάνω στο πεδίο *balance* της σχέσης *account*



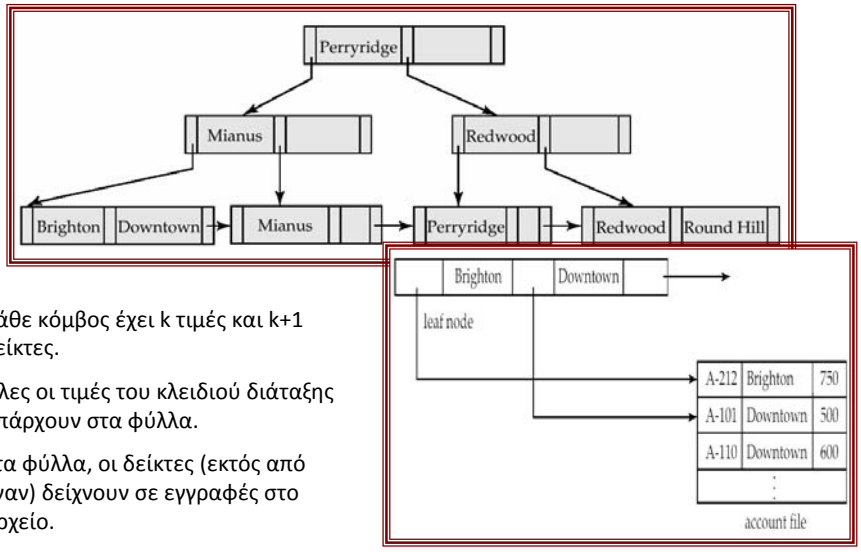
Βρείτε το λάθος !!

Ευρετήρια με χρήση B⁺-δένδρων

- **Μειονέκτημα των διατεταγμένων αρχείων**: η απόδοση μειώνεται καθώς το αρχείο μεγαλώνει, από τη στιγμή που μπορεί να δημιουργηθούν πολλά blocks υπερχείλισης. Απαιτείται περιοδική αναδιοργάνωση όλου του αρχείου.
- Λύση: χρήση **B⁺-δένδρων**



Παράδειγμα B⁺-δένδρου



- Κάθε κόμβος έχει k τιμές και k+1 δείκτες.
- Όλες οι τιμές του κλειδιού διάταξης υπάρχουν στα φύλλα.
- Στα φύλλα, οι δείκτες (εκτός από έναν) δείχνουν σε εγγραφές στο αρχείο.

Ορισμός B^+- δένδρου

Ένα B^+-δένδρο **τάξης n** (συνήθως, n περιττός αριθμός) είναι ένα δένδρο με ρίζα που ικανοποιεί τις παρακάτω ιδιότητες:

- Όλα τα μονοπάτια από τη ρίζα προς ένα φύλλο έχουν το ίδιο μήκος (ισοζυγισμένο δένδρο).
- Ένας εσωτερικός κόμβος με k τιμές έχει k+1 παιδιά.
- Ένας εσωτερικός κόμβος έχει από $\lceil n/2 \rceil$ έως n παιδιά.
  - Ειδικά η ρίζα (αν είναι εσωτερικός κόμβος και όχι φύλλο) έχει από 2 έως n παιδιά.
- Ένα φύλλο έχει από $\lceil (n-1)/2 \rceil$ έως n–1 τιμές.
  - Ειδικά η ρίζα (αν είναι φύλλο) έχει από 0 έως n–1 τιμές.

Οι κόμβοι του B^+-δένδρου

- **Τυπικός κόμβος**

P_1	K_1	P_2	...	P_{n-1}	K_{n-1}	P_n

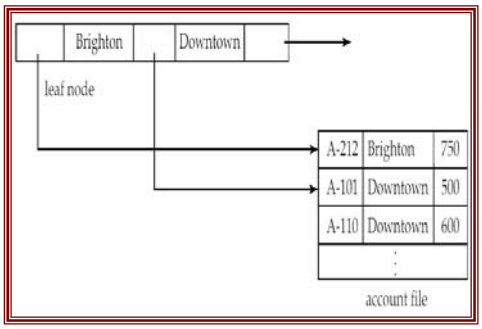
  - K_i είναι τιμές του κλειδιού διάταξης.
  - P_i είναι δείκτες σε παιδιά (για εσωτερικούς κόμβους) ή δείκτες σε blocks με εγγραφές (για φύλλα).

- **Τα κλειδιά διάταξης σε ένα κόμβο είναι διατεταγμένα**

$$K_1 < K_2 < K_3 < \ldots < K_{n-1}$$

Τα φύλλα του B^+-δένδρου

- Ιδιότητες ενός φύλλου
  - Για i = 1, 2, …, n–1, ο δείκτης P_i δείχνει σε μια εγγραφή (στο αρχείο) με τιμή κλειδιού διάταξης K_i
    - Δηλαδή, το τελευταίο επίπεδο (φύλλα) λειτουργεί ως διατεταγμένο ευρετήριο
  - Ο δείκτης P_n δείχνει στο επόμενο φύλλο βάσει της σειράς του κλειδιού διάταξης.
    - Άρα, υπάρχει διάταξη στα φύλλα ενός B^+-δένδρου.



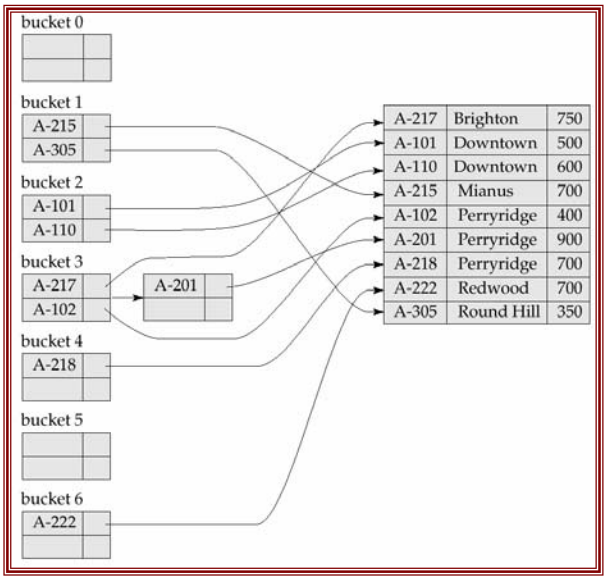
Υπέρ και κατά των B^+-δένδρων

- **Πλεονεκτήματα** από τη χρήση των B^+-δένδρων:
  - έχουν τη δυνατότητα αυτόματης αναδιοργάνωσης με μικρές, τοπικές αλλαγές κατά τη διάρκεια εισαγωγών και διαγραφών. Για τη διατήρηση της απόδοσης δεν απαιτείται αναδιοργάνωση όλου του αρχείου.
- **Μειονεκτήματα** από τη χρήση των B^+-δένδρων:
  - επιπλέον κόστος I/O για την εισαγωγή και διαγραφή καθώς και επιπλέον απαίτηση χώρου.
- Τα πλεονεκτήματα των B^+-δένδρων υπερκαλύπτουν τα μειονεκτήματα και αυτό δικαιολογεί την ευρεία χρήση τους.

Ευρετήρια κατακερματισμού

- Ο κατακερματισμός δεν χρησιμοποιείται μόνο για οργάνωση αρχείων, αλλά και για τη δημιουργία δομών ευρετηρίων.
- Ένα **ευρετήριο κατακερματισμού** (hash index) οργανώνει τα κλειδιά διάταξης και τους αντίστοιχους δείκτες στις εγγραφές σε μια δομή ενός αρχείου κατακερματισμού.
- Εάν το αρχείο είναι οργανωμένο με κατακερματισμό, είναι πλεονασμός να υπάρχει και ξεχωριστό ευρετήριο πάνω σε αυτό χρησιμοποιώντας το ίδιο κλειδί διάταξης.

Παράδειγμα ευρετηρίου κατακερματισμού



Διατεταγμένα ή κατακερματισμού ;

- Κόστος περιοδικής αναδιοργάνωσης.
- Σχετική συχνότητα εισαγωγών και διαγραφών.
- Αναμενόμενοι τύποι ερωτήσεων:
  - Ο κατακερματισμός είναι προτιμότερος για **ερωτήσεις ταυτότητας** πάνω στο κλειδί διάταξης.
  - Για **ερωτήσεις διαστήματος** πάνω στο κλειδί διάταξης επιλέγονται τα ταξινομημένα ευρετήρια.
    - Ο κατακερματισμός **δεν** μπορεί να υποστηρίξει ερωτήσεις διαστήματος

Ορισμός ευρετηρίων στην SQL

- Δημιουργία ενός ευρετηρίου

  create [unique] index <όνομα ευρετηρίου>
  on <όνομα σχέσης> (<λίστα χαρακτηριστικών>)

  Π.χ.: **create index** *b-index* **on** *branch(branch-name)*

  - Χρήση της παραμέτρου **unique** για να δηλώσουμε ότι το κλειδί διάταξης είναι υποψήφιο κλειδί.
- Για να διαγράψουμε ένα ευρετήριο

  drop index <όνομα ευρετηρίου>

Donald Knuth and the B^+-tree

- About the naming of B^+-trees:
  - A variation of B-trees, B*-Trees, was named by **Donald Knuth** in The Art of Computer Programing, Vol. 3, Searching and Sorting. Knuth proposed that B*-Trees had nodes that were at least 2/3 full rather than 1/2 full. Knuth also proposed another B-Tree variation having all keys in the leaves of the tree and the leaves are linked from left-to-right (making B^+-Tree ideal for sequential search) Comer says that Knuth did not name this variation, but Comer chooses to call it B^+-Tree. (source: http://mactech.com/articles/mactech/Vol.06/06.11/Nov90Letters/index.html)

Donald Knuth