ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Βάσεις Δεδομένων ΙΙ

ISAM και Β-Δέντρα
Φυσικός Σχεδιασμός για Βάσεις Δεδομένων

Μανώλης Μαραγκουδάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

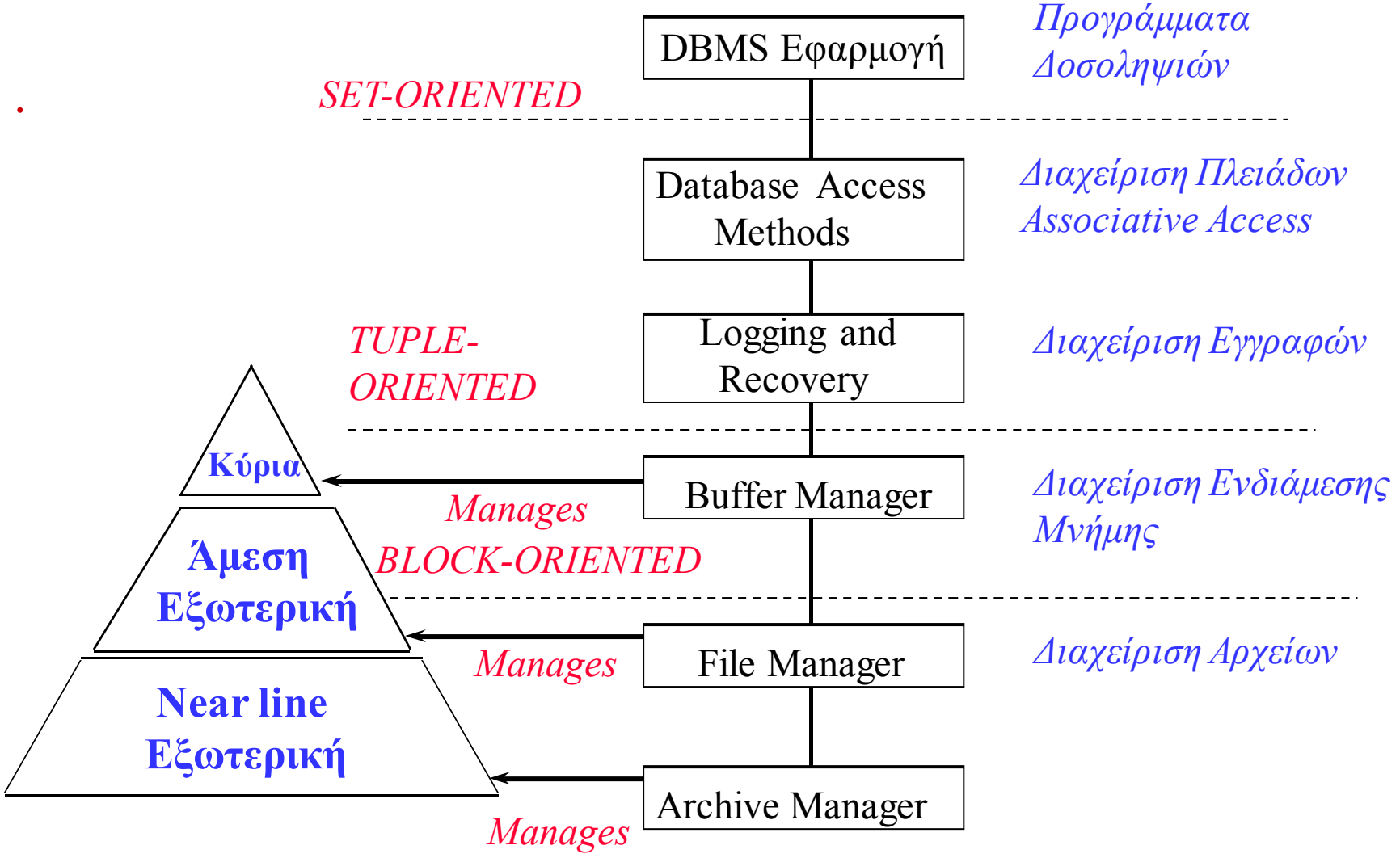
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Παράρτημα

ISAM και B-Δέντρα

Φυσικός Σχεδιασμός για Βάσεις Δεδομένων

Διαχείριση Μνήμης

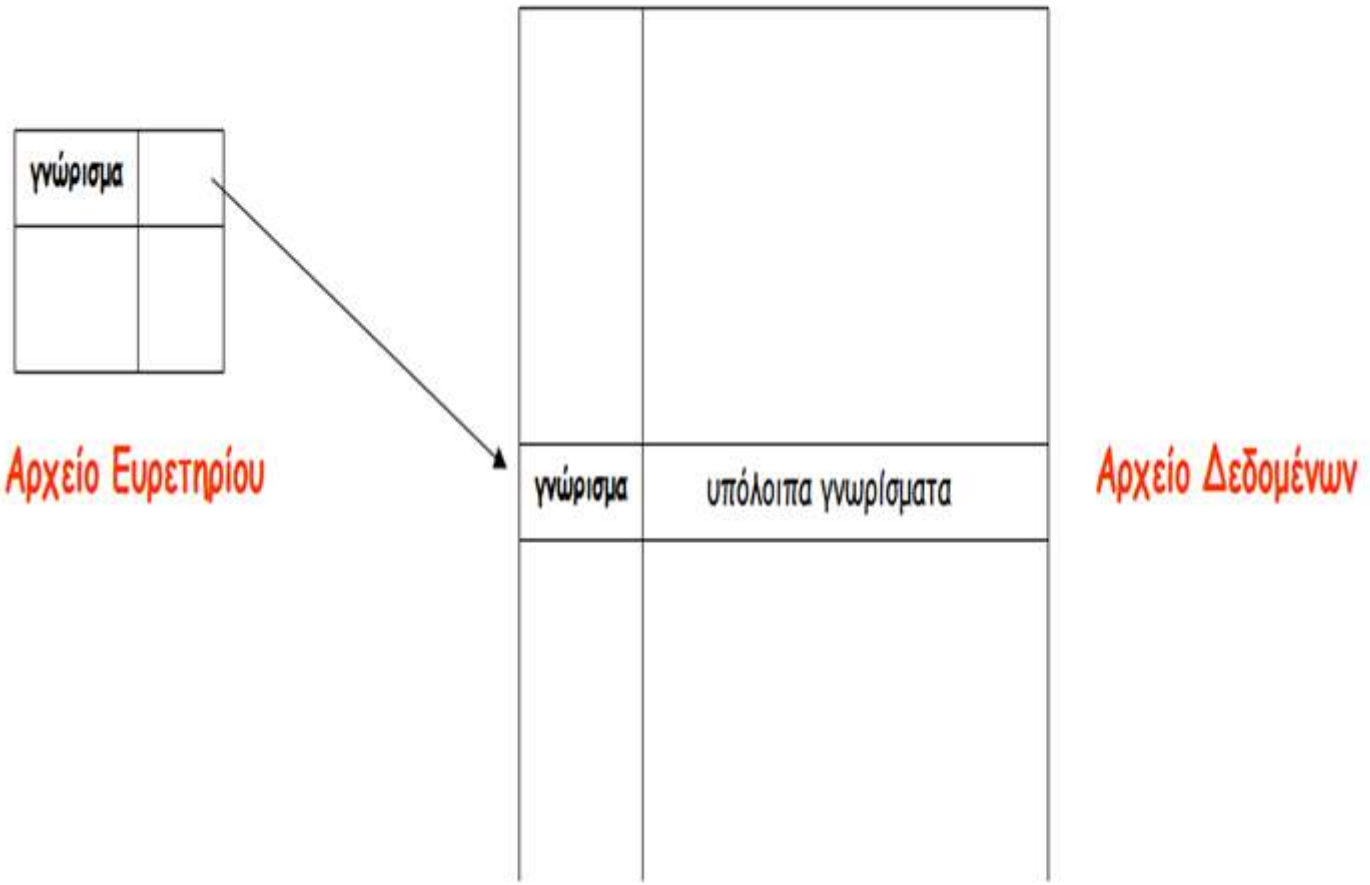


Δομές Ευρετηρίων

- 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***
- 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 Μπλοκ όπου αποθηκεύεται η εγγραφή που θέλουμε
- Μια καταχώρηση / εγγραφή στο Ευρετήριο έχει την μορφή:

Τιμή Κλειδιού	***Δείκτης στο Μπλοκ της εγγραφής***

Παράδειγμα


γνώρισμα
Αρχείο Ευρετηρίου
γνώρισμα
υπόλοιπα γνωρίσματα
Αρχείο Δεδομένων

Ευρετήρια

- ***Τα Ευρετήρια (ενός επιπέδου) χωρίζονται σε:***
  - ***Πρωτεύον Ευρετήριο:*** ορίζεται σε ένα αρχείο που είναι *διατεταγμένο στο (κύριο) κλειδί.* Περιλαμβάνει μια καταχώρηση ***για κάθε Μπλοκ***. Η καταχώρηση έχει την τιμή του κλειδιού της *πρώτης εγγραφής στο Μπλοκ*. (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 Μπλοκ που περιέχει εγγραφές με αυτή την τιμή γνωρίσματος
  - ***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***)

Ευρετήρια και Αρχεία - Στόχοι

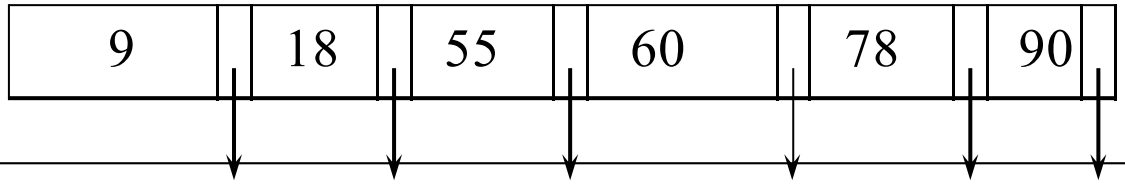
- Οι εγγραφές ενός αρχείου τοποθετούνται σε (κατάλληλα) Μπλοκ με την υποστήριξη μιας ***πρωτεύουσας δομής αρχείου / ευρετηρίου*** ***(ISAM Δέντρου, Β-Δέντρου ή Κατακερματισμού)*** ή ακόμη και σε ***Σωρούς.***
- Στη συνέχεια, οι εγγραφές μπορεί να ανακληθούν μέσω των τιμών Κλειδιών ***(TID – Tuple Identifier ή RID – Record Identifier)*** – με την υποστήριξη των δομών που χρησιμοποιήθηκαν για την αποθήκευσή τους
- Επιπλέον, δημιουργούνται ***δευτερεύουσες δομές ευρετηρίων*** ***(με ανάλογες τεχνικές υλοποίησης)*** ***ή / και ευρετήρια συστάδων*** ώστε οι εγγραφές να δύνανται να ανακληθούν με **αποδοτικό τρόπο** μέσω ***τιμών οιονδήποτε γνωρισμάτων.***

Ευρετήρια Πολλών Επιπέδων

- **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 (επικαιροποίηση του ευρετηρίου)
- Ένα πολύ-επίπεδο ευρετήριο αποτελεί ένα **Δέντρο Αναζήτησης** με την υπόθεση ότι το πρώτο επίπεδο **χωρά σε ένα Μπλοκ**

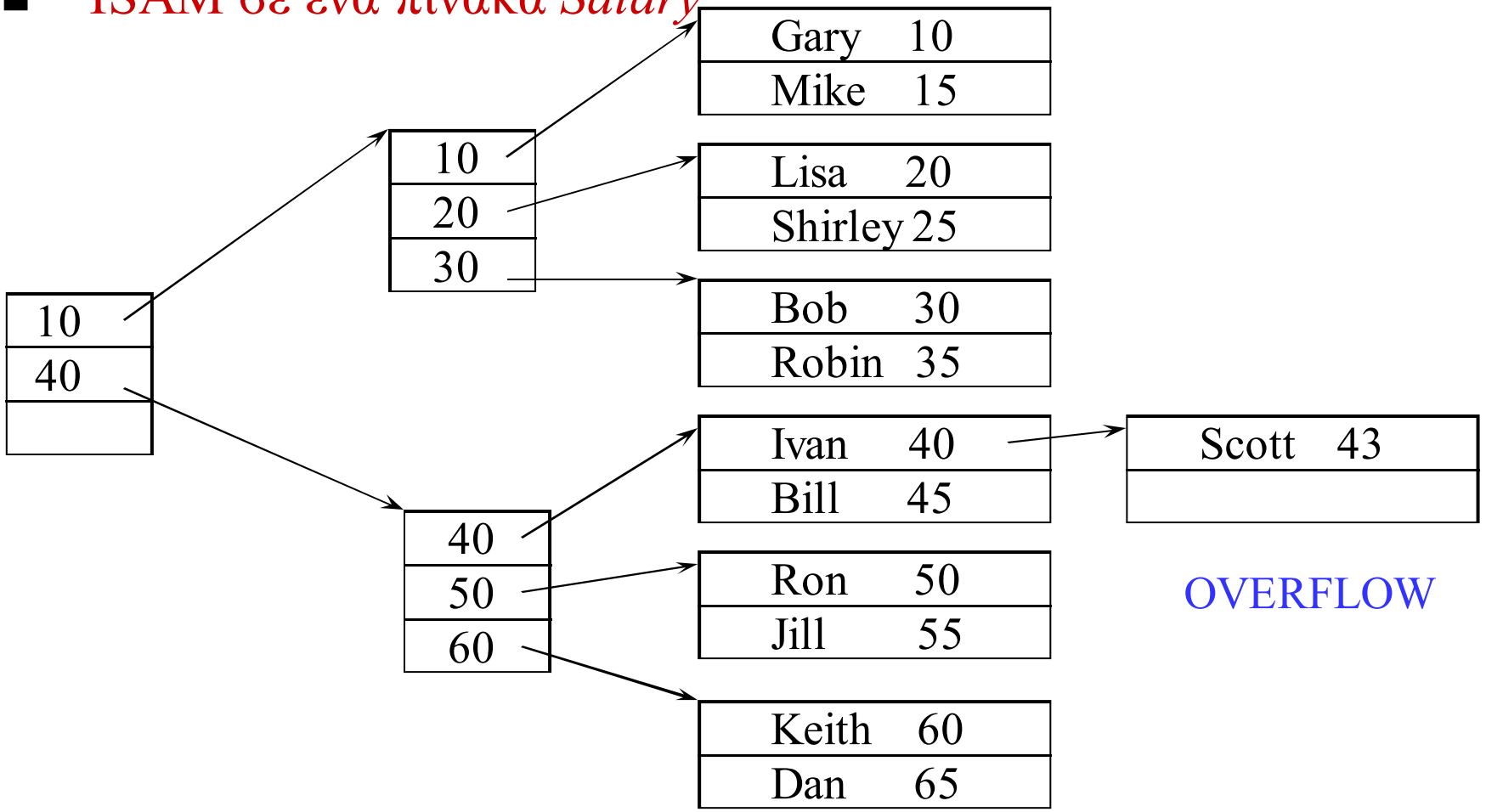
Indexed Sequential Access Method (ISAM)

- Το ISAM είναι μια πολύ-επίπεδη **Δομή Ευρετηρίου (Δέντρο)** για ***άμεση πρόσβαση*** σε εγγραφές αρχείου **διατεταγμένου στο Κλειδί**
- Κάθε κόμβος του Δέντρου είναι **ένα Μπλοκ** στο Δίσκο.
- 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



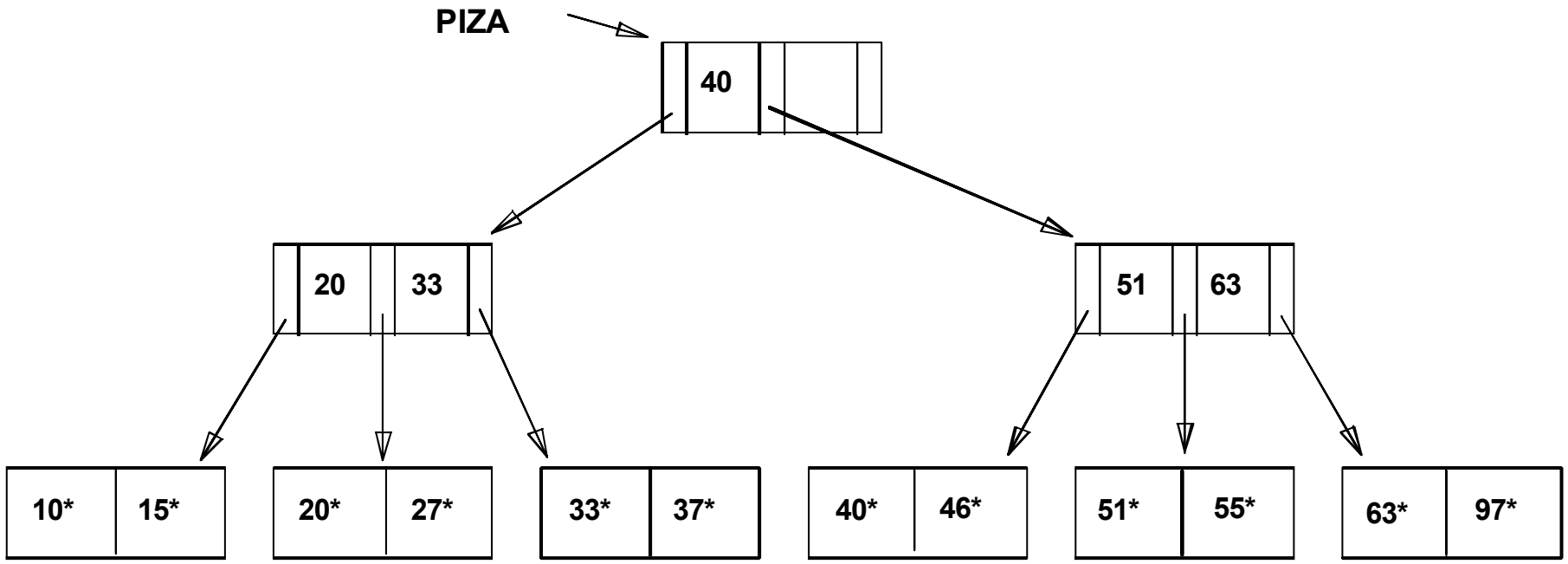
Παράδειγμα ISAM (Δευτερεύον Ευρετήριο)

- ISAM σε ένα πίνακα *Salary*

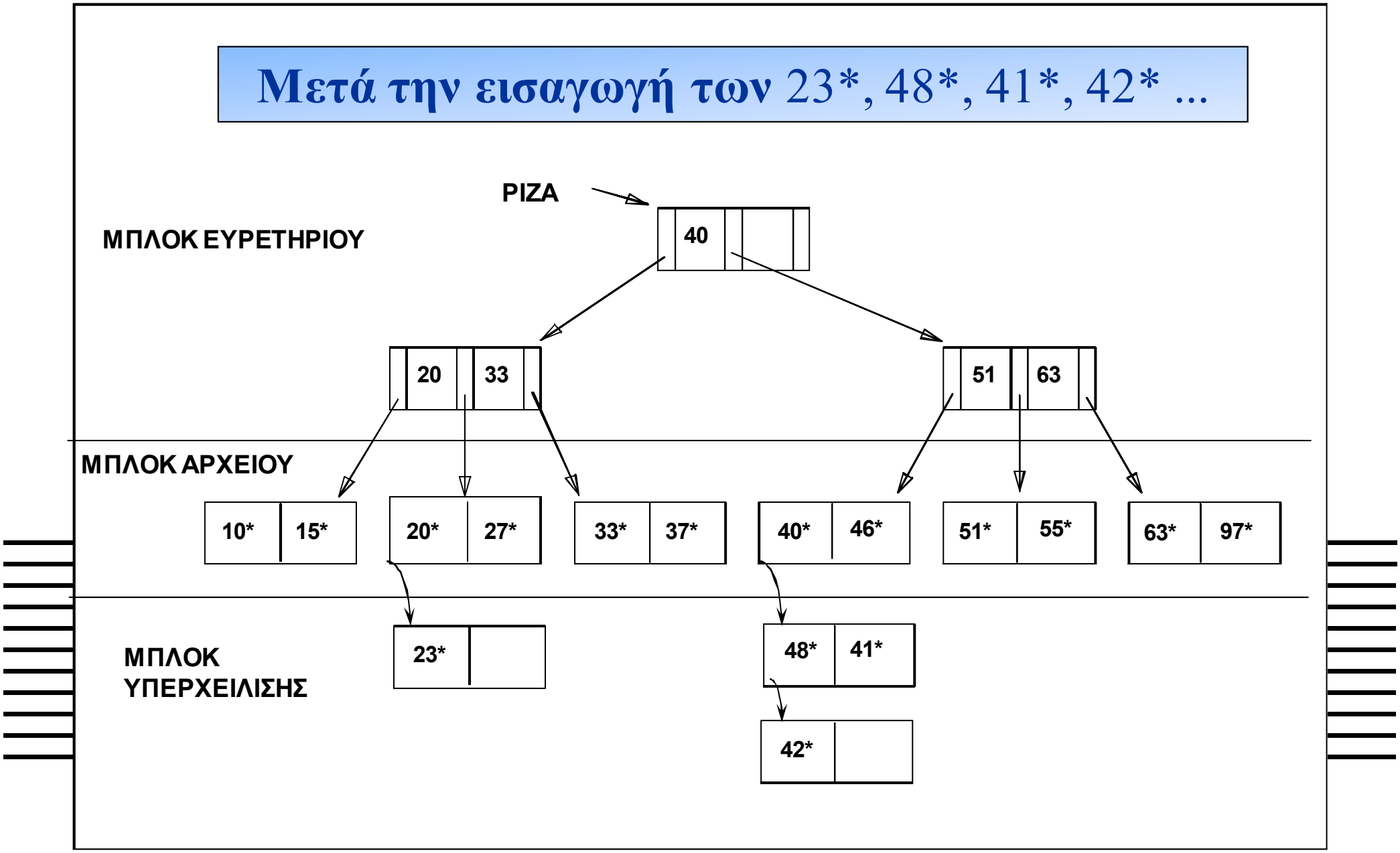


ISAM Δέντρο -- Παράδειγμα (2)

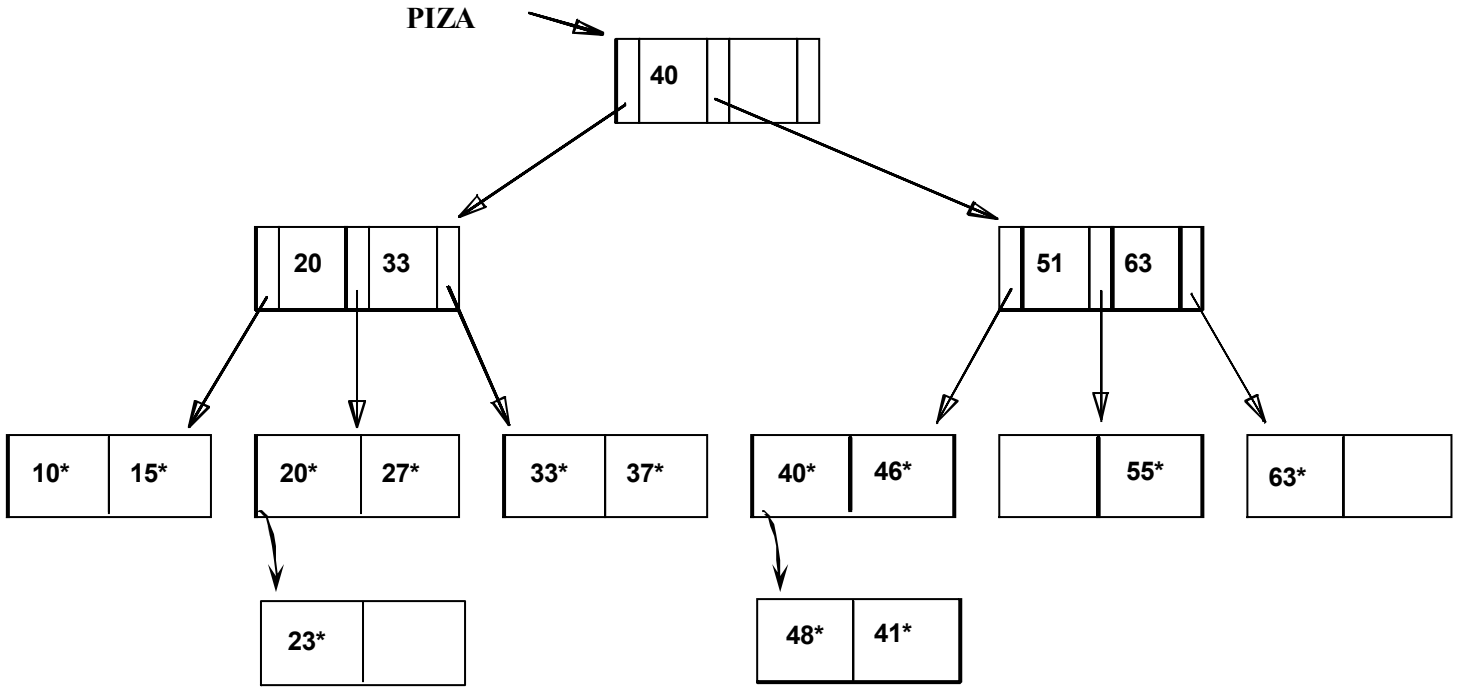
- Κάθε κόμβος δύναται να έχει 2 καταχωρήσεις



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



Μετά την διαγραφή των 42*, 51*, 97*



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

Η Επίδοση του ISAM

- **ΕΠΙΔΟΣΗ (PERFORMANCE).**

  Έστω ότι έχουμε D Σελίδες (Μπλοκ) για Δεδομένα και k δείκτες για κάθε κόμβο (υποθέτουμε ότι το $D = k^L$)

  ΔΙΑΔΙΚΑΣΙΑ ΑΝΑΖΗΤΗΣΗΣ:

Σειριακή Σάρωση (scan):	D
Δυαδική Αναζήτηση Σχέσης:	$\log_2 D + 1$
Δυαδική Αναζήτηση (μονό επίπεδο):	$(\log_2 (D/k) + 1) + 1$
Διάσχιση του ISAM Δέντρου:	$\log_k D + 1 = L + 1$

ISAM -- ΣΧΟΛΙΑ

- **ΠΛΕΟΝΕΚΤΗΜΑΤΑ:**
  - Παρέχει έναν **ταξινομημένο κατάλογο** για το Αρχείο (ή Σχέση)
  - Εξαιρετική δομή για **ακριβείς ερωτήσεις (exact queries)** π.χ., *Salary = 400000*
  - Το ISAM διευκολύνει την εκτέλεση των **ερωτήσεων διακύμανσης** (**range queries).** π.χ., *Salary μεταξύ 350.000 και 600.000*

- **ΜΕΙΟΝΕΚΤΗΜΑΤΑ**
  - Είναι μια **ΣΤΑΤΙΚΗ** δομή που εύκολα χάνει την ¨ισορροπία¨ της (**unbalanced) -** *ισορροπία ύψους και πληρότητας των κόμβων*
  - Αν στο Αρχείο γίνονται πολλές ενημερώσεις (volatile δεδομένα), τότε το Αρχείο μπορεί να χάσει την ταξινόμησή του.
  - Το Ευρετήριο απαιτεί μεγάλο χώρο το Δίσκο

B^+ - Δέντρα

- Το **B^+-Δέντρο** είναι μια πολύ-επίπεδη δομή ευρετηρίου για ένα διατεταγμένο αρχείο
- Οι κόμβοι-φύλλα **περιέχουν ταξινομημένες εγγραφές (πλειάδες)**, οι άλλοι (εσωτερικοί) κόμβοι έχουν **ειδική μορφή**
- Ένας κόμβος αντιστοιχεί σε ένα **Μπλοκ (Σελίδα)**
- Κάθε κόμβος κρατιέται ***κάτι μεταξύ γεμάτος και μισό-γεμάτος***
- Οι Εισαγωγές σε κόμβους, που **δεν είναι γεμάτοι, γίνονται αποδοτικά**; Αν ένας κόμβος είναι γεμάτος, τότε έχουμε **διάσπαση**
- Οι Διαγραφές γίνονται **πολύ αποδοτικά** αν ο κόμβος **δεν καθίσταται** λιγότερο από μισό-γεμάτος (ώστε να απαιτήσει διάσπαση)
- Η δομή Δέντρου παραμένει ανά πάσα στιγμή **ΙΣΟΡΡΟΠΗΜΕΝΗ**

B^+ - Δέντρα -- Σύνοψη

- Μορφή Εσωτερικών Κόμβων:

P_1	K_1	P_2	K_2	...	...	P_{n-1}	K_{n-1}	P_n

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

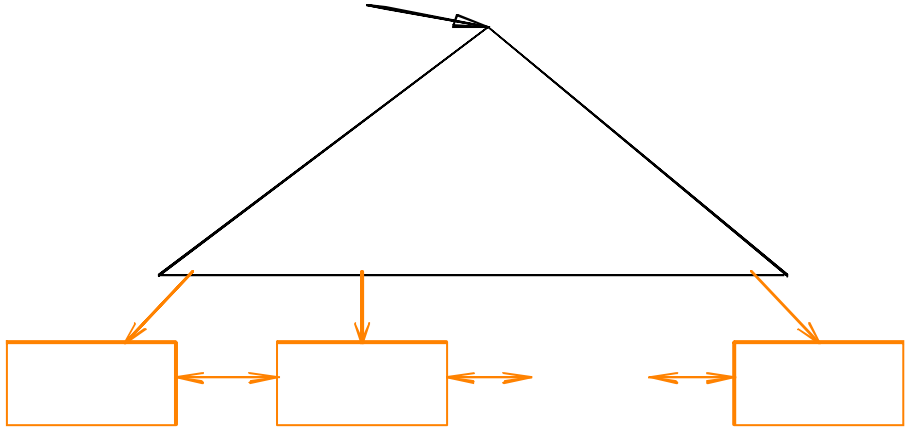
Το P_1 δείχνει ένα κόμβο που περιέχει τιμές κλειδιού n, $n < K_1$

Το P_2 δείχνει ένα κόμβο που περιέχει τιμές κλειδιού n, $K_1 <= n < K_2$

- **Παραλλαγές των B^+- Δέντρων:**
  - B-Δέντρα : *Σαν τα B^+-Δέντρα, αλλά οι εσωτερικοί κόμβοι περιέχουν επιπλέον δείκτες σε δεδομένα. Είναι συνήθως πιο μεγάλα και είναι δύσκολο να υλοποιηθούν, ενίοτε όμως, είναι ταχύτερα*
  - B*-Δέντρα : *Σαν τα B^+-Δέντρα, αλλά κρατούν κάθε κόμβο γεμάτο (τουλάχιστον) κατά τα 2/3. Μικρότερα και ταχύτερα δέντρα, αλλά πολύ χειρότερα για Εισαγωγές /Διαγραφές*

B^+ - Δέντρα – Τα πλέον δημοφιλή Ευρετήρια

- 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 **d <= *m*** **<= 2d** καταχωρήσεις. Το **d** ονομάζεται ***Τάξη*** *του Δέντρου*.
- Εξαιρετική δομή **ΚΑΙ** για exact queries **ΚΑΙ** για range queries.

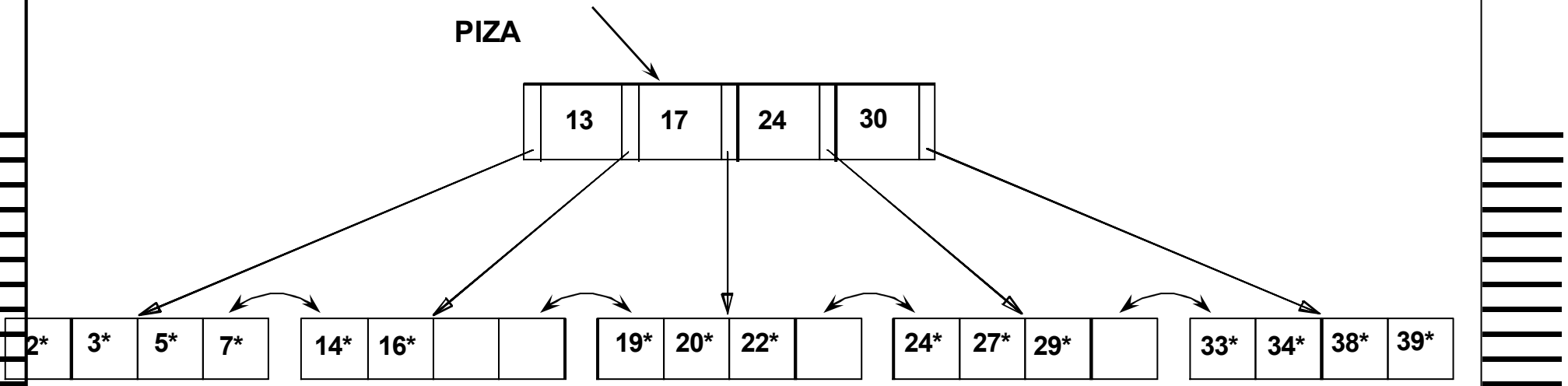


Καταχωρήσεις Ευρετηρίου (Άμεση Αναζήτηση)

Καταχωρήσεις Δεδομένων («Σύνολο ακολουθίας»)

Παράδειγμα B+ Δέντρου

- Η αναζήτηση ξεκινά από τη Ρίζα, και οι συγκρίσεις των κλειδιών μας οδηγούν στα φύλλα (όπως στο ISAM).
- Αναζήτηση για τα 5*, 15*, όλες οι καταχωρήσεις >= 24* ...



B^+ - Δέντρα – Πρακτικά Στοιχεία

- 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

Αναζήτηση στα B+ δέντρα

```
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
```

Η Εισαγωγή μιας καταχώρησης δεδομένων (εγγραφής)

Αν ο κόμβος-φύλλο είναι γεμάτος (έχει p_{leaf} εγγραφές, όπου $p_{leaf}=2d$,d η τάξη του κόμβου)

διάσπαση του κόμβου:

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

-- οι υπόλοιπες σε καινούργιο κόμβο

-- εισαγωγή (αντιγραφή) της k+1-οστής τιμής (K_{k+1}) στον γονέα

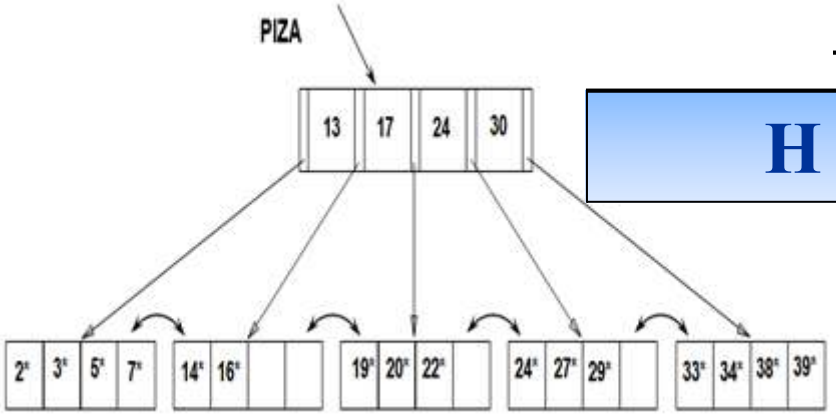
Αν ένας εσωτερικός κόμβος είναι γεμάτος (έχει ρ εγγραφές)

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

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

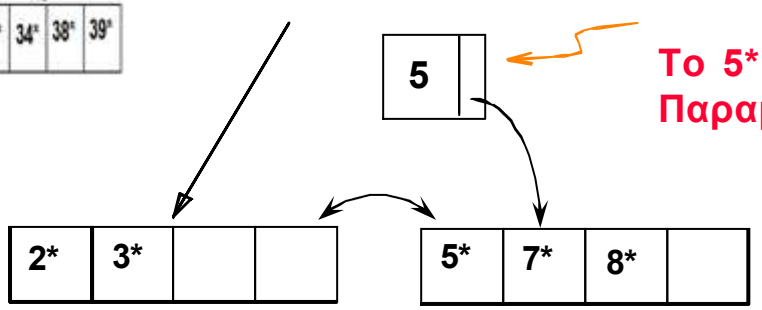
-- η k+1-οστή K_{k+1} τιμή μεταφέρεται (*δεν αντιγράφεται*) στον πατέρα

-- οι υπόλοιπες σε καινούργιο κόμβο

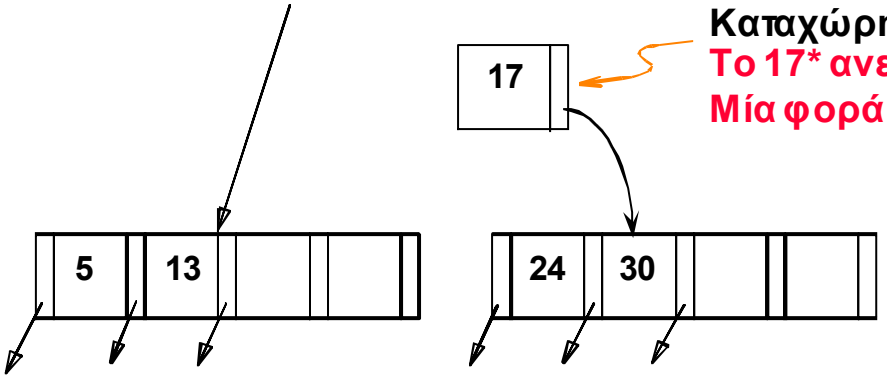


Η Εισαγωγή της καταχώρησης 8*

Καταχώρηση στον πατέρα κόμβο.

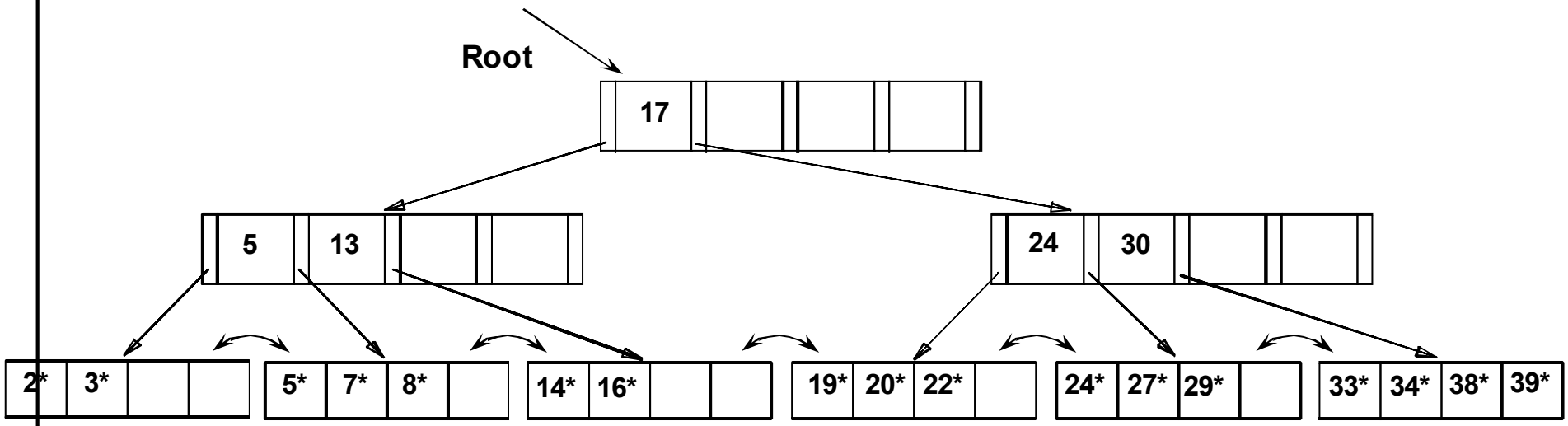


Το 5* ανεβαίνει απάνω, αλλά
Παραμένει και στο φύλλο



Καταχώρηση στον Πατέρα Κόμβο
Το 17* ανεβαίνει απάνω και παρουσιάζεται μόνο
Μία φορά στο Ευρετήριο (σε αντίθεση με Φύλλα)

Τελικό B+ Δέντρο Μετά την Εισαγωγή του 8*

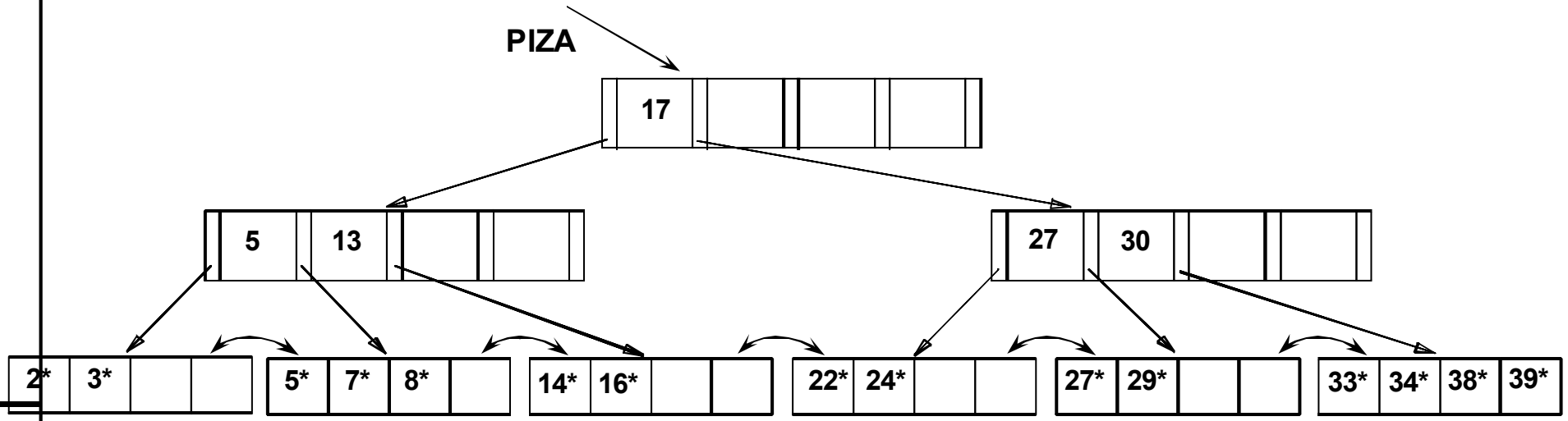


- Η PIZA διασπάστηκε οδηγώντας σε αύξηση του ύψους.

Η Διαγραφή μιας καταχώρησης δεδομένων (εγγραφής)

- Αρχίζοντας από τη Ρίζα, βρες το φύλλο *L* όπου ανήκει η καταχώρηση
- Διέγραψε την καταχώρηση.
  - Αν η L είναι τουλάχιστον μισό-γεμάτη, *τελείωσες!*
  - Αν η L έχει μόνο **d-1** καταχωρήσεις,
    - » Προσπάθησε να κάνεις ανακατανομή , δανειζόμενος το *sibling (γειτονικός κόμβος (αδερφός) με τον ίδιο πατέρα του L)*.
    - » Αν η ανακατανομή αποτύχει, συγχώνευσε το *L* και το sibling.
- Αν έγινε συγχώνευση, πρέπει να διαγραφεί η καταχώρηση (που δείχνει στο *L* ή το sibling) από τον πατέρα του *L*.
- Η συγχώνευση μπορεί να φτάσει στη Ρίζα, μειώνοντας το ύψος του Δέντρου.

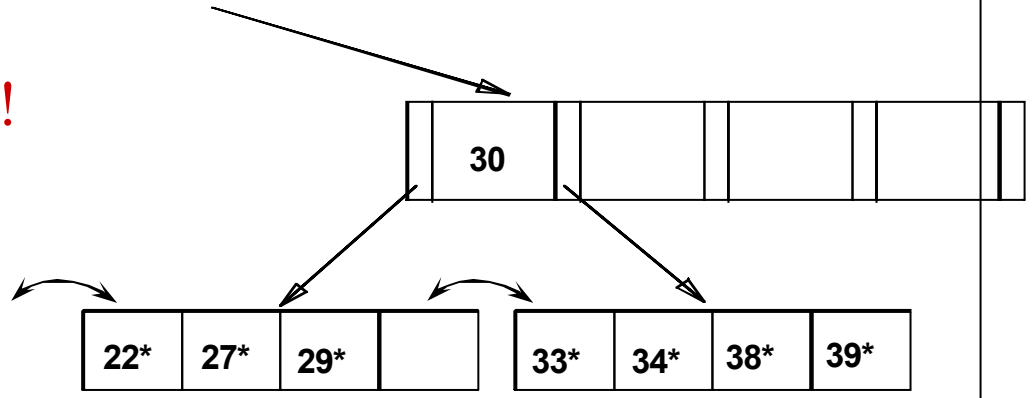
Το Παράδειγμα μετά την Διαγραφή του 19* και του 20*

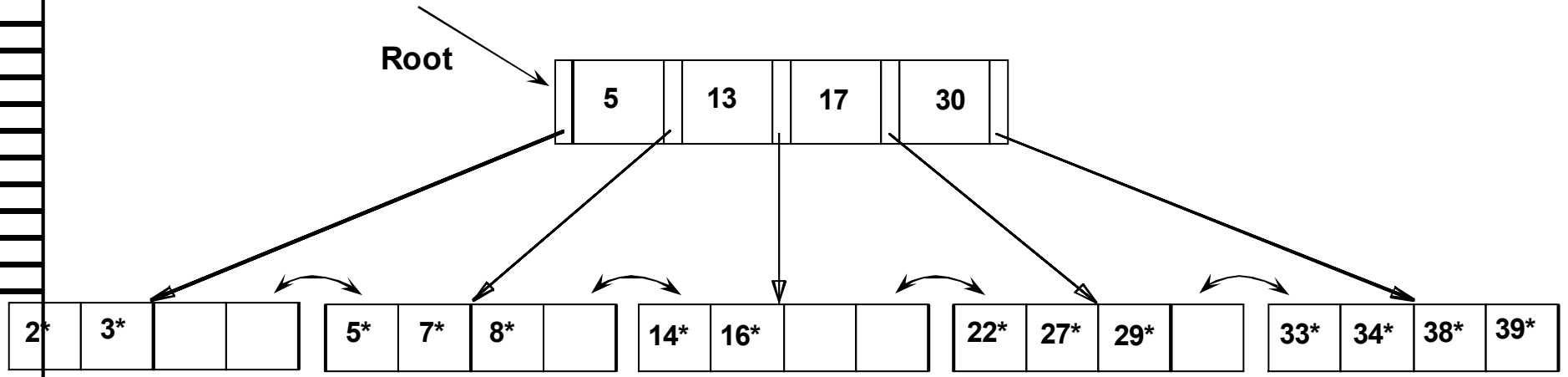


- Η Διαγραφή του 19* ήταν εύκολη
- Η Διαγραφή του 20* έγινε με ανακατανομή. Το μεσαίο κλειδί ανέβηκε.

Τέλος, η Διαγραφή του 24*

- Απαιτείται ΣΥΓΧΩΝΕΥΣΗ!





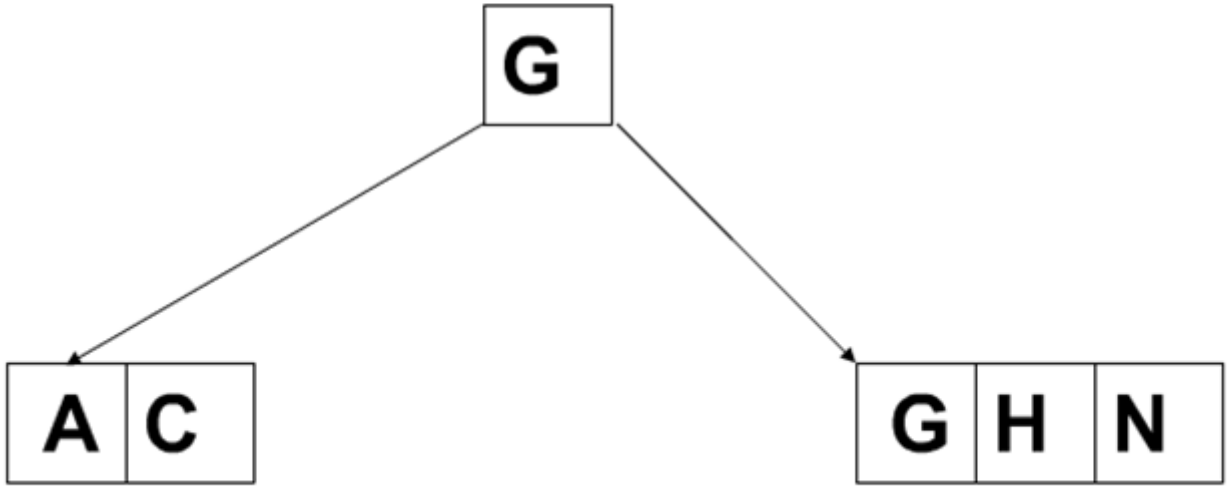
Παράδειγμα (έστω ρ=4)

C N G A H E K Q M F W L T Z D P R X Y S

A	C	G	N

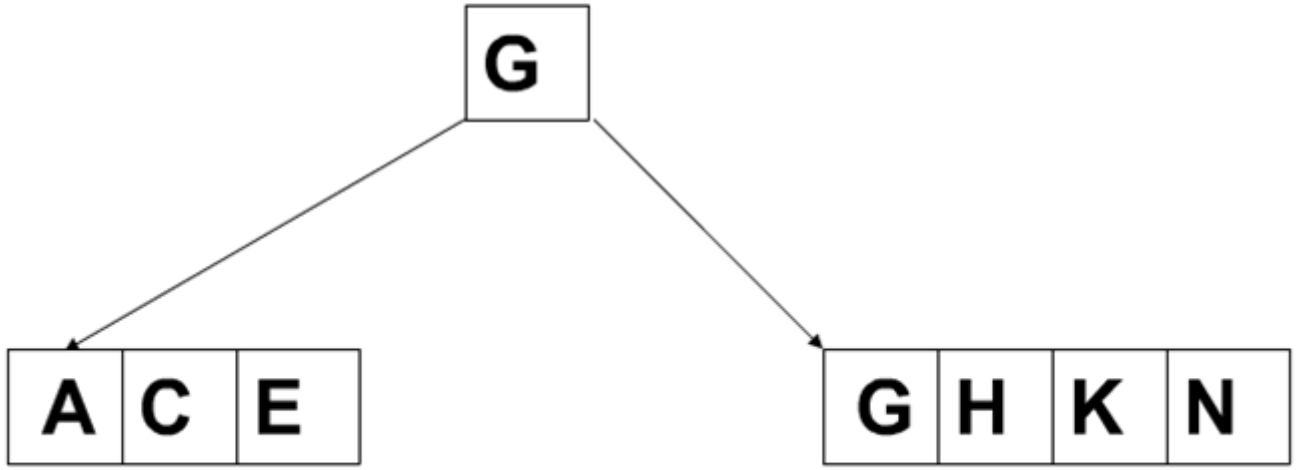
Παράδειγμα

C N G A *H* E K Q M F W L T Z D P R X Y S



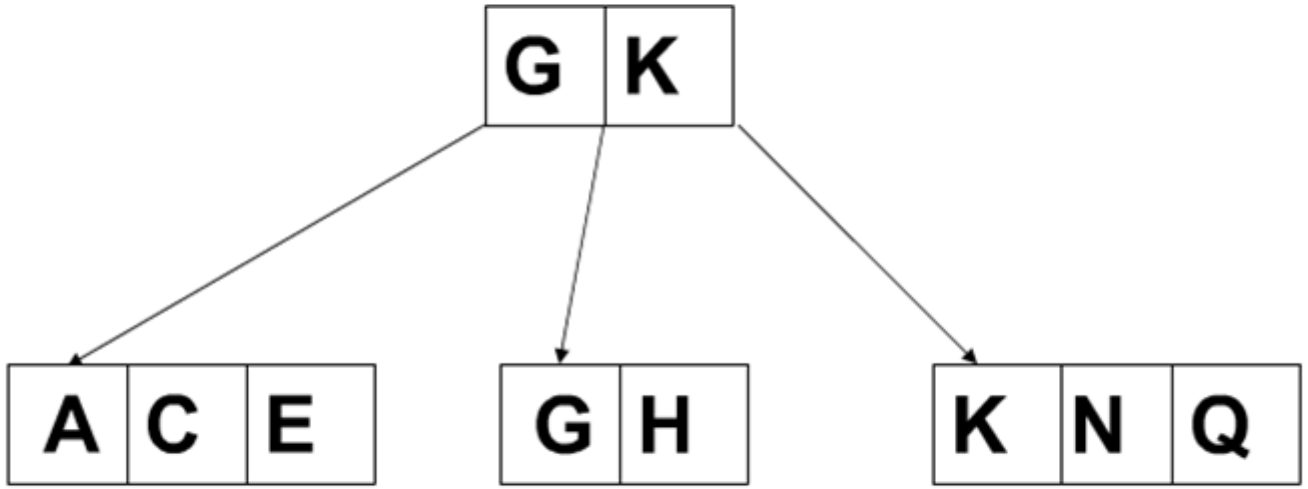
Παράδειγμα

C N G A H *E K* Q M F W L T Z D P R X Y S



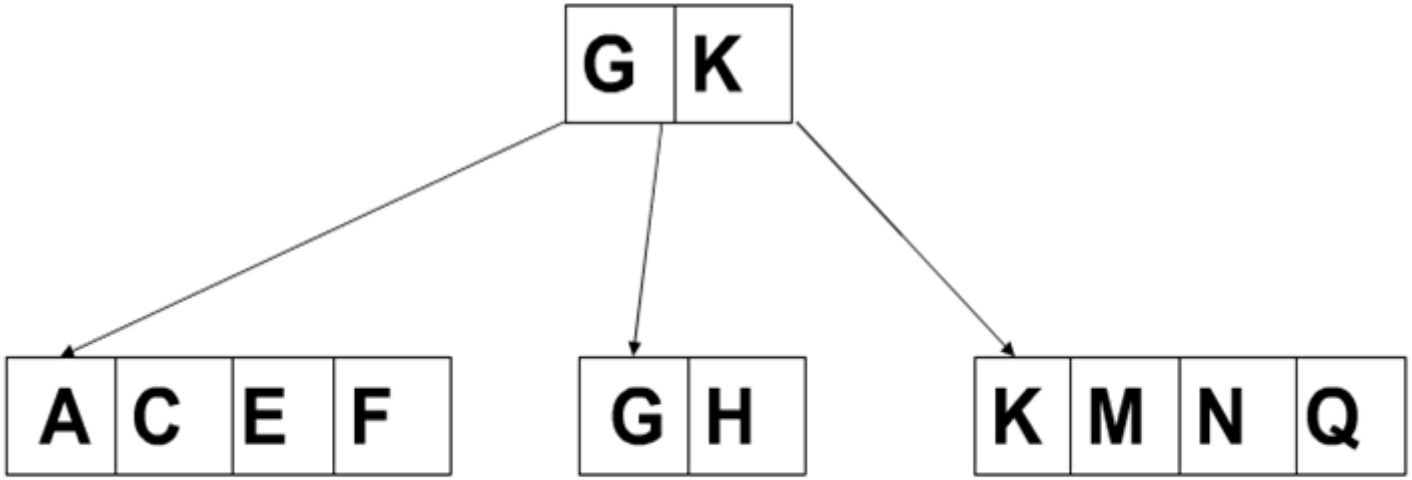
Παράδειγμα

C N G A H E K Q M F W L T Z D P R X Y S



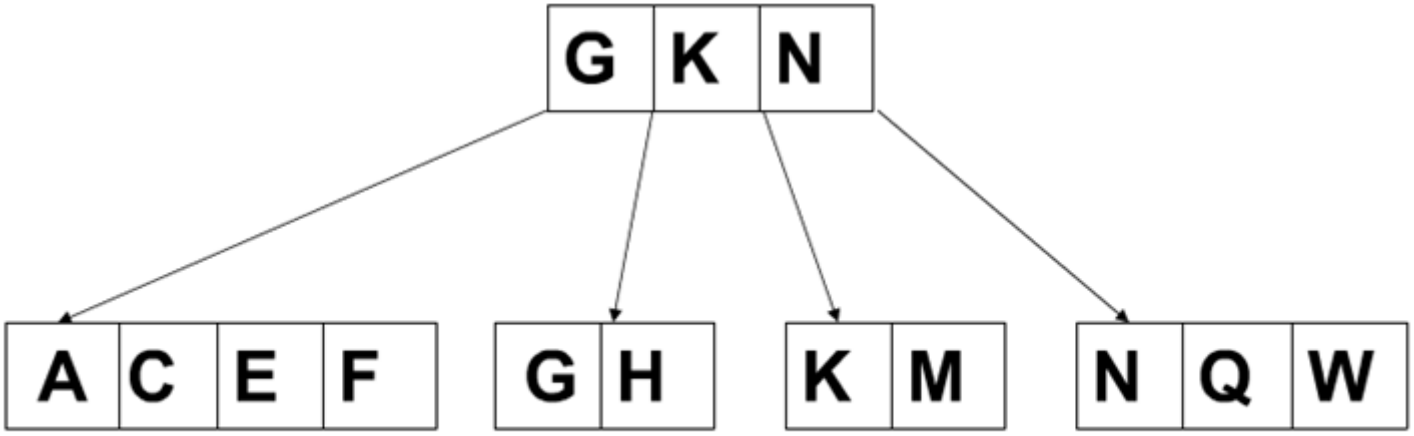
Παράδειγμα

C N G A H E K Q *M F* W L T Z D P R X Y S



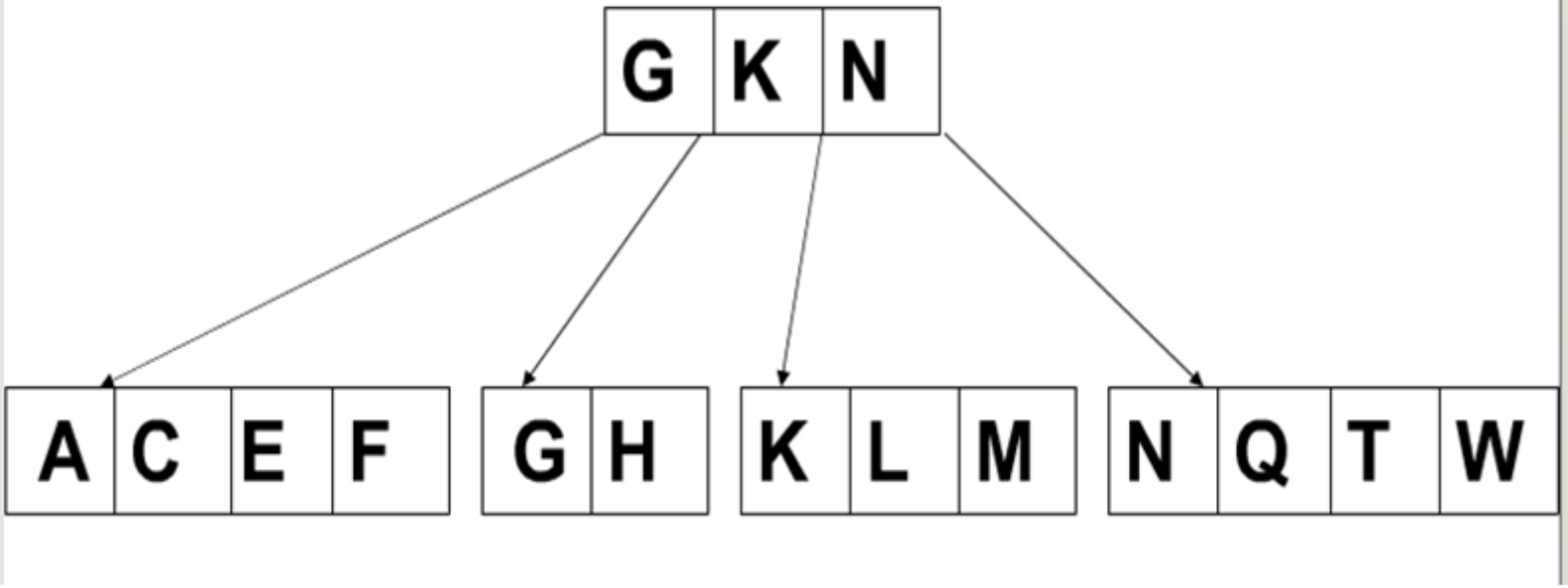
Παράδειγμα

C N G A H E K Q M F *W* L T Z D P R X Y S



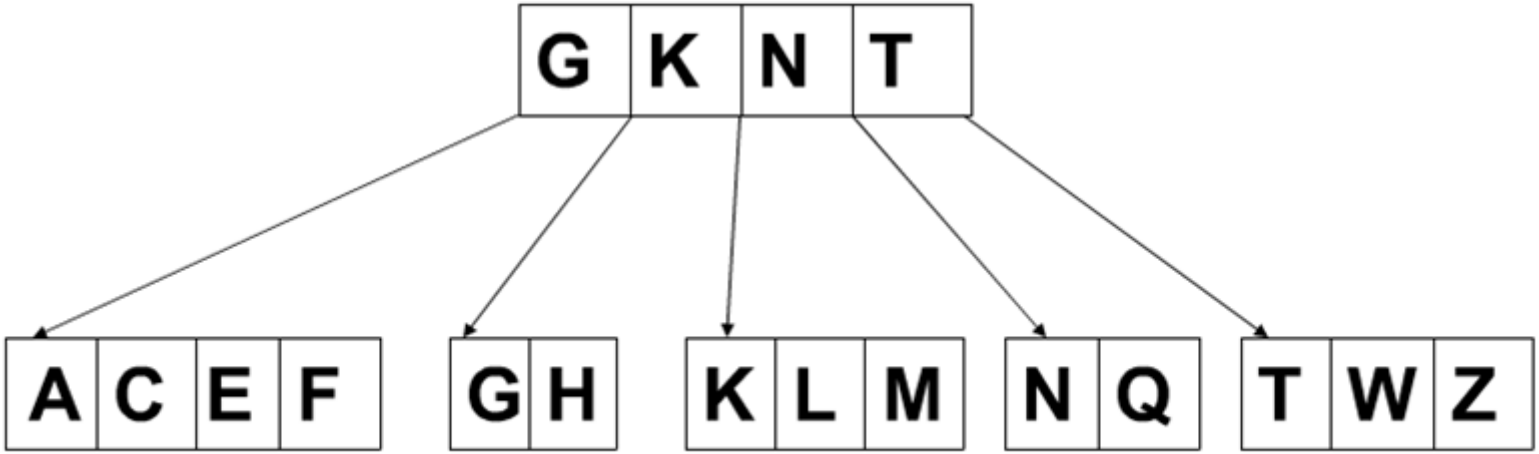
Παράδειγμα

C N G A H E K Q M F W L T Z D P R X Y S



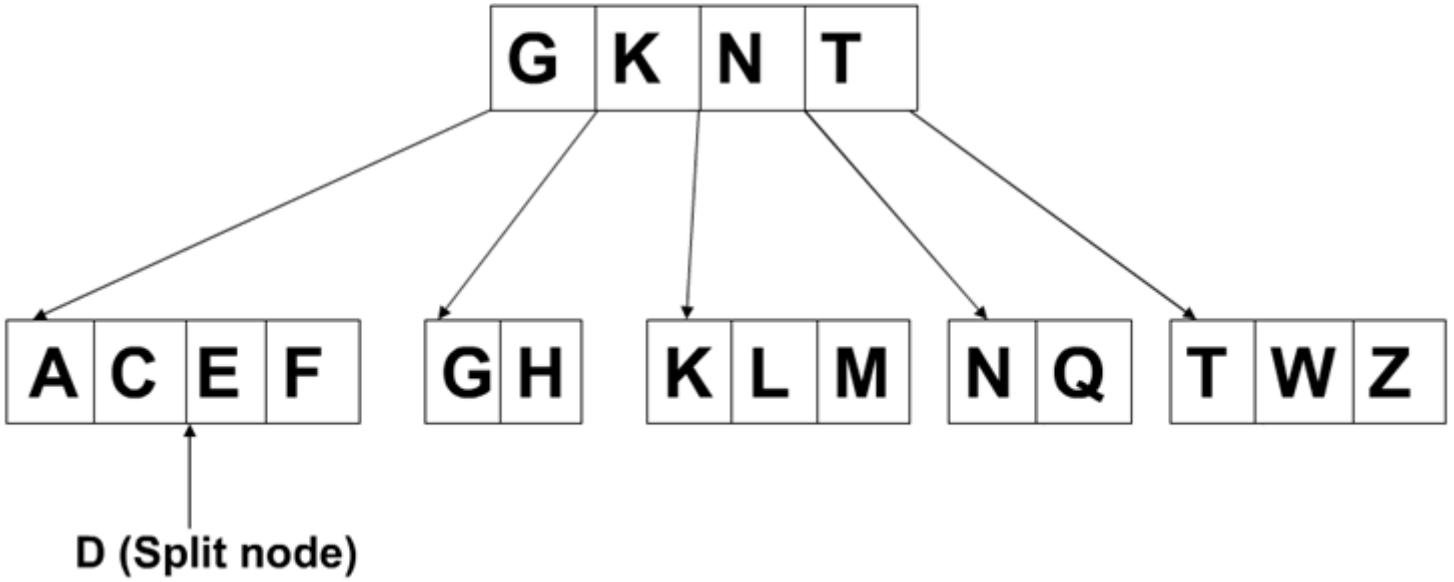
Παράδειγμα

C N G A H E K Q M F W L T Z D P R X Y S

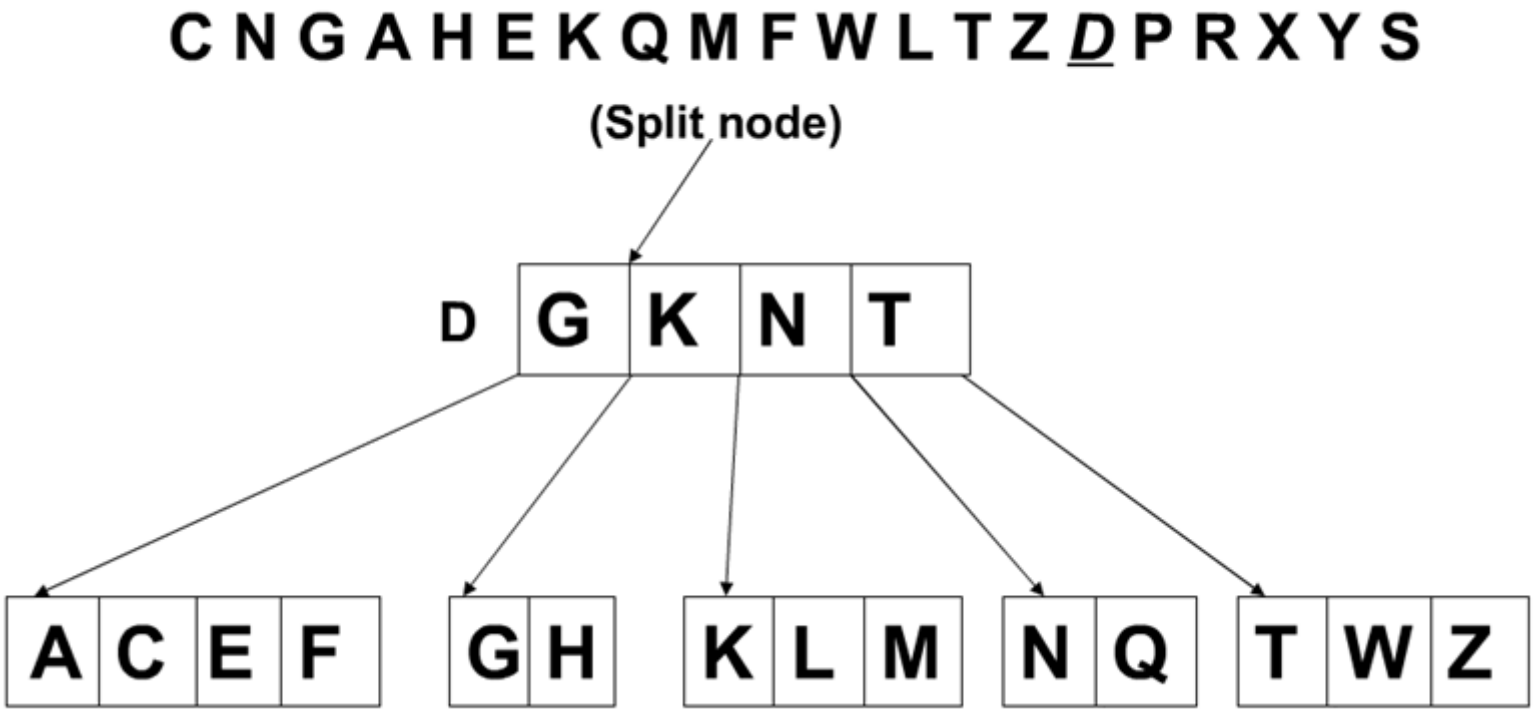


Παράδειγμα

C N G A H E K Q M F W L T Z *D* P R X Y S

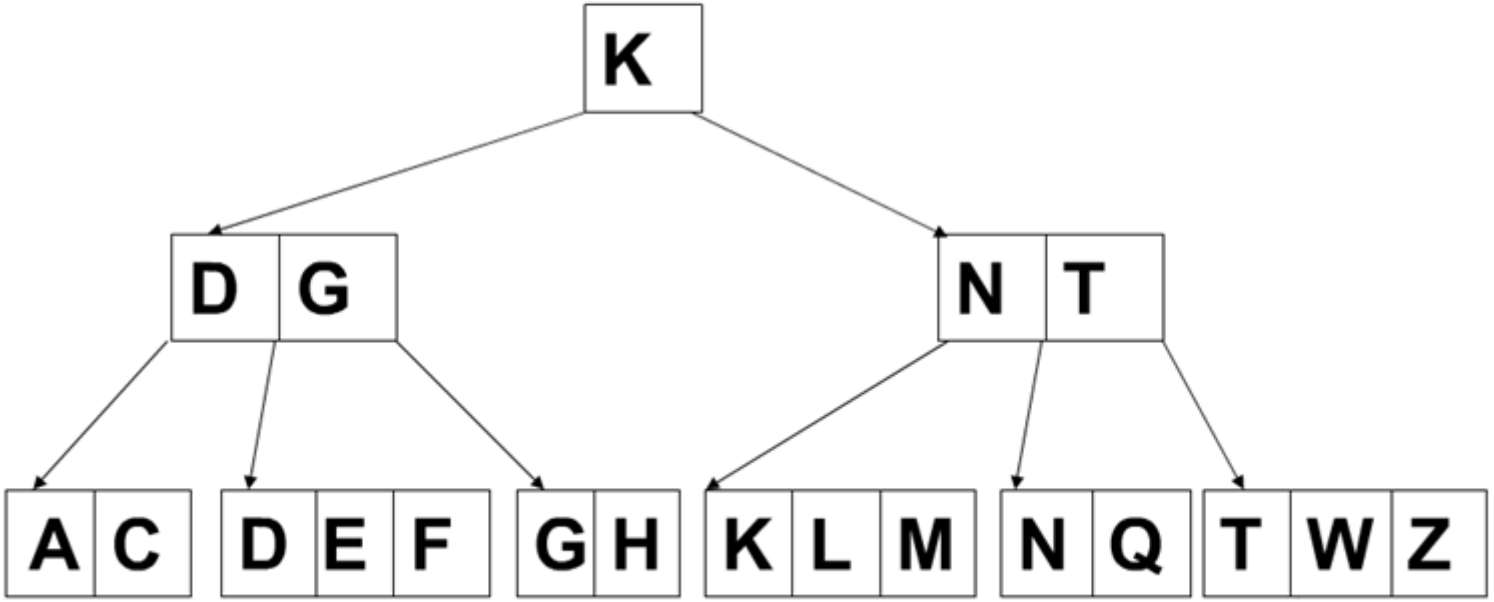


Παράδειγμα



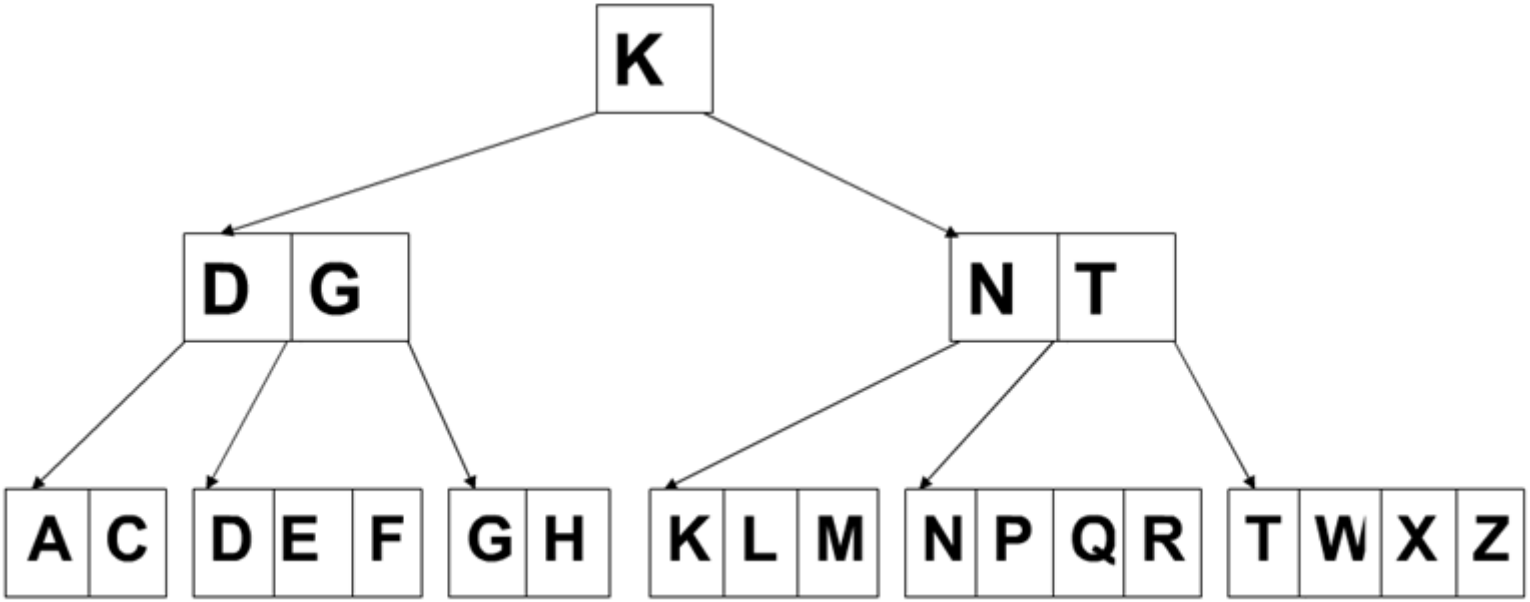
Παράδειγμα

C N G A H E K Q M F W L T Z *D* P R X Y S



Παράδειγμα

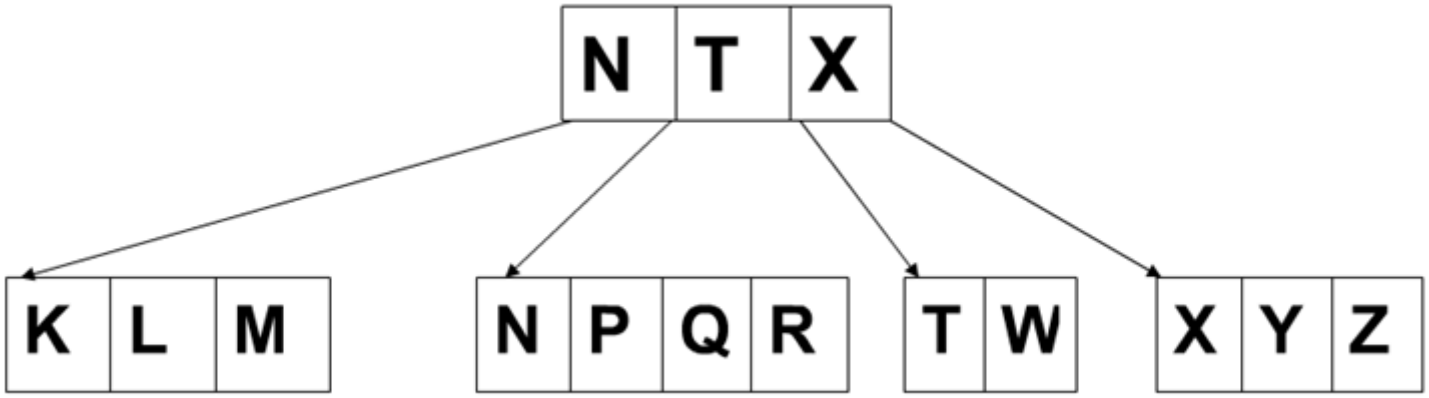
C N G A H E K Q M F W L T Z D ***P R X*** Y S



Παράδειγμα

C N G A H E K Q M F W L T Z D P R X Y S

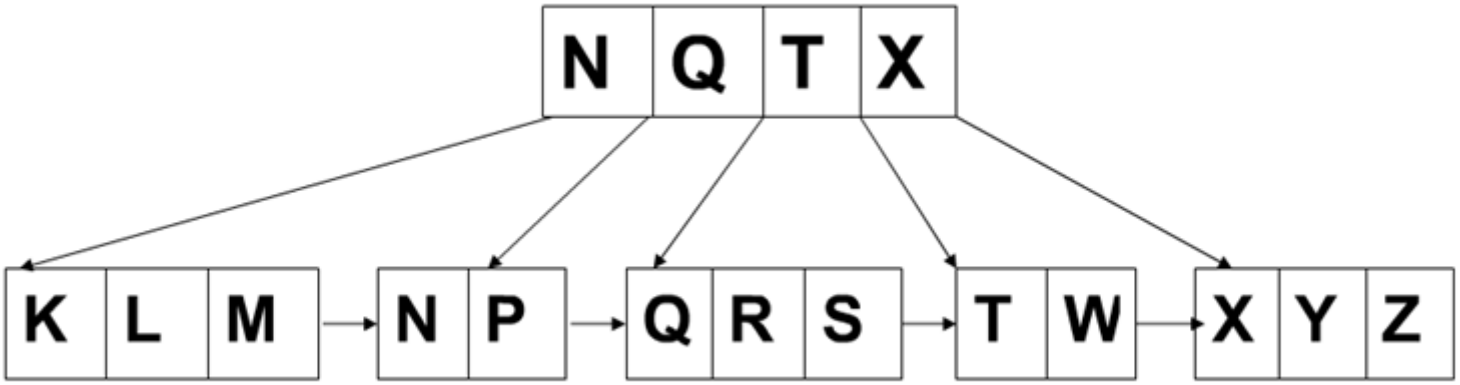
Right Sub-Tree



Παράδειγμα

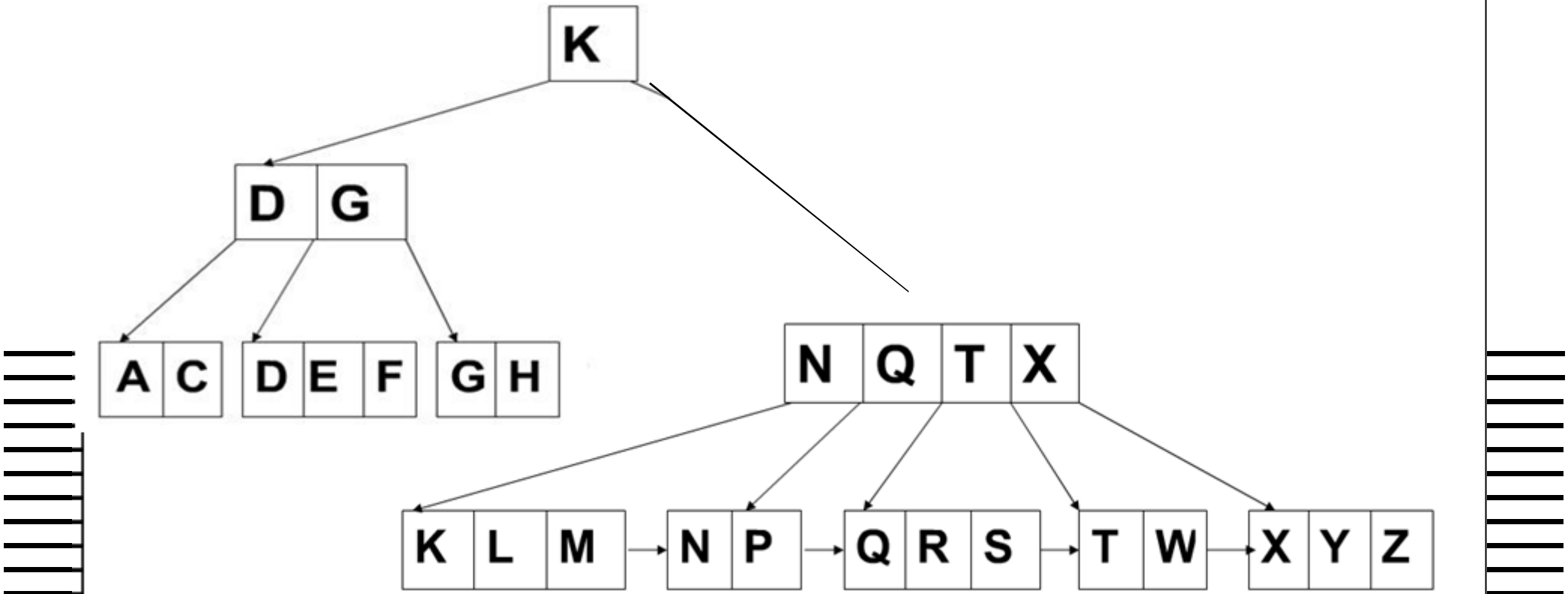
C N G A H E K Q M F W L T Z D P R X Y *S*

Right Sub-Tree



C N G A H E K Q M F W L T Z D P R X Y S

Παράδειγμα-Τελικό Δέντρο

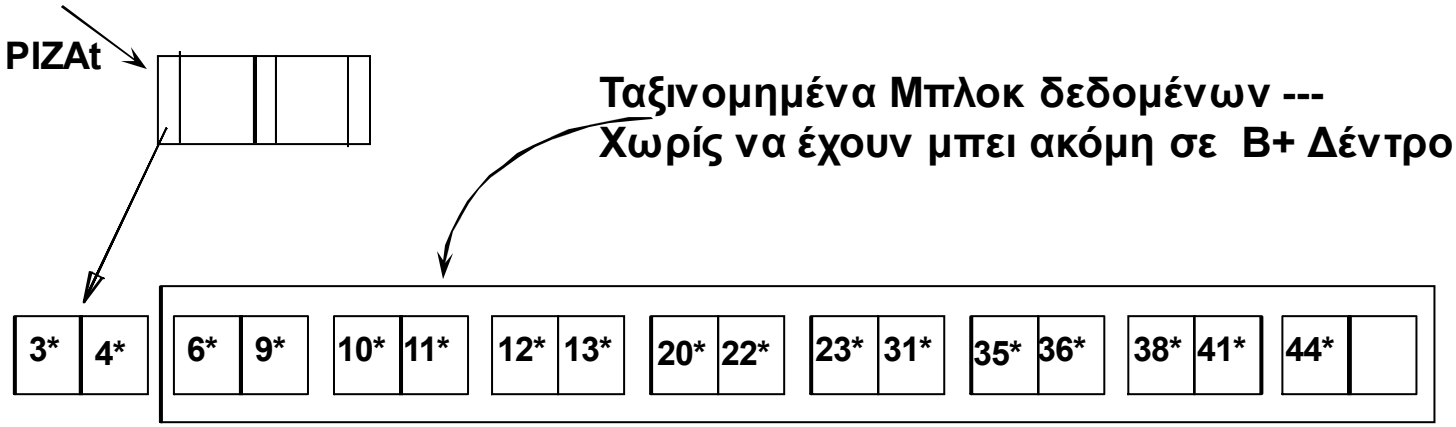


Ταχύτητα σε ένα B+ δέντρο

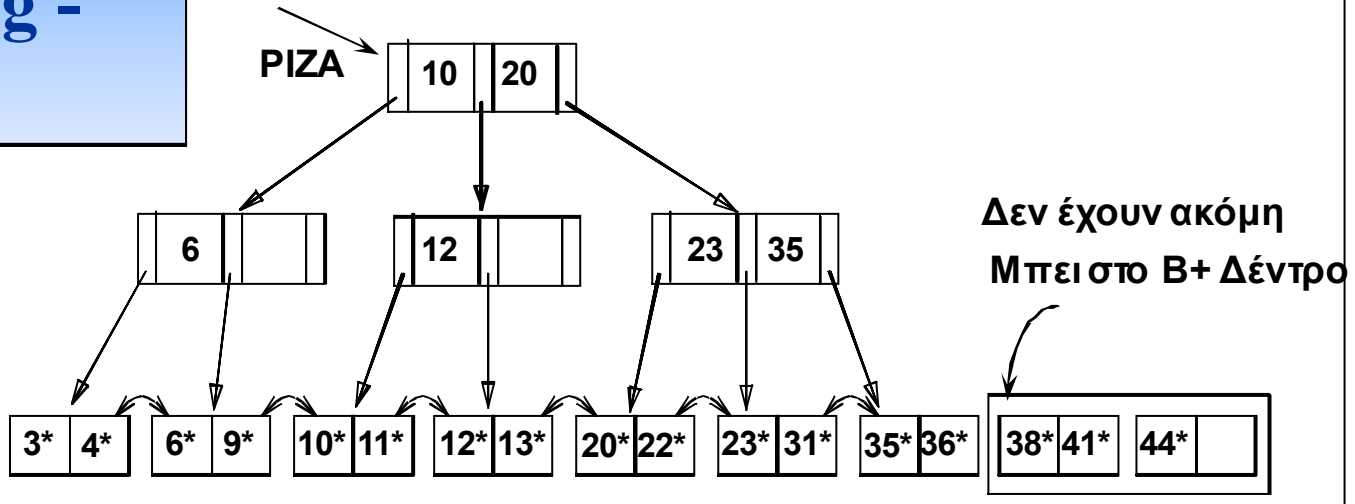
- Για K τιμές αναζήτησης
  - μήκος μονοπατιού από τη ρίζα μέχρι το φύλλο=$\text{Log}_{(n/2)}(K)$
    - » n=αριθμός των συνδέσμων από οποιονδήποτε κόμβο
  - Π.χ. για n=100, τότε μήκος(1.000.000 αναζητήσεις)= $\text{Log}_{(100/2)}(1.000.000)=4$
  - Αφού συνήθως η ρίζα είναι ήδη στο Buffer, τυπικά απαιτούνται <3 αναγνώσεις στο δίσκο (περίπου 90 msec!)

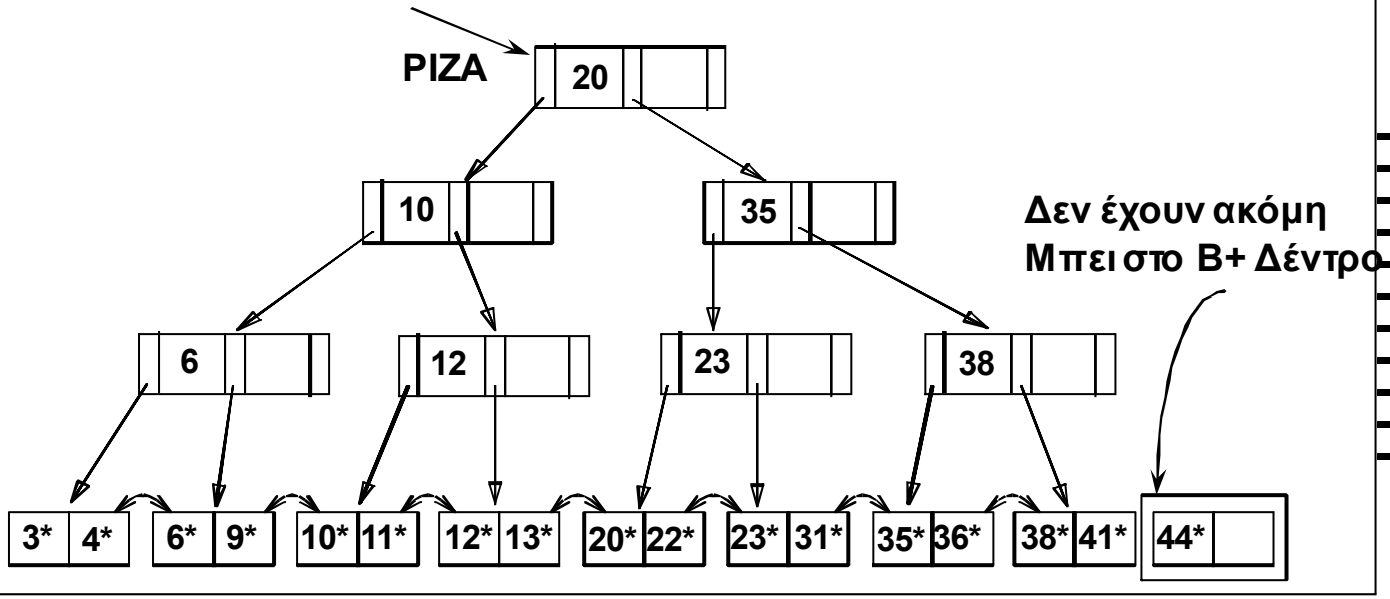
Μαζική Εισαγωγή B+ Δέντρου (Bulk Loading)

- Αν έχουμε πολλές εγγραφές και θέλουμε να δημιουργήσουμε ένα B+ Δέντρο σε κάποιο γνώρισμα, χρησιμοποιώντας τις παραπάνω μεθόδους (εγγραφή προς εγγραφή) θα έχουμε μεγάλη καθυστέρηση
- *Το Bulk Loading* είναι εξαιρετικά πιο αποδοτικό
- *Αρχικοποίηση*: Ταξινόμησε όλες τις καταχωρήσεις (δεδομένα), βάλε δείκτη στον πρώτο κόμβο φύλλο (Μπλοκ) σε μια νέα (Ρίζα) Σελίδα (Μπλοκ)



Bulk Loading - Συνέχεια





Σύνοψη του Bulk Loading

- Εναλλακτικός Τρόπος 1: Πολλαπλές Εισαγωγές
  - Αργός
  - Δεν καταλήγει σε σειριακή τοποθέτηση των φύλλων
- Εναλλακτικός Τρόπος 1 : *Bulk Loading*
  - Πλεονεκτήματα στη Λειτουργία (π.χ., έλεγχο συνδρομικότητας)
  - Μικρότερος αριθμός από I/Os κατά την διάρκεια εισαγωγής
  - Σειριακή τοποθέτηση των φύλλων στο Δίσκο
  - Ελέγχει καλύτερα τον παράγοντα πληρότητας

ΣΥΝΟΨΗ για Δομές Ευρετηρίων

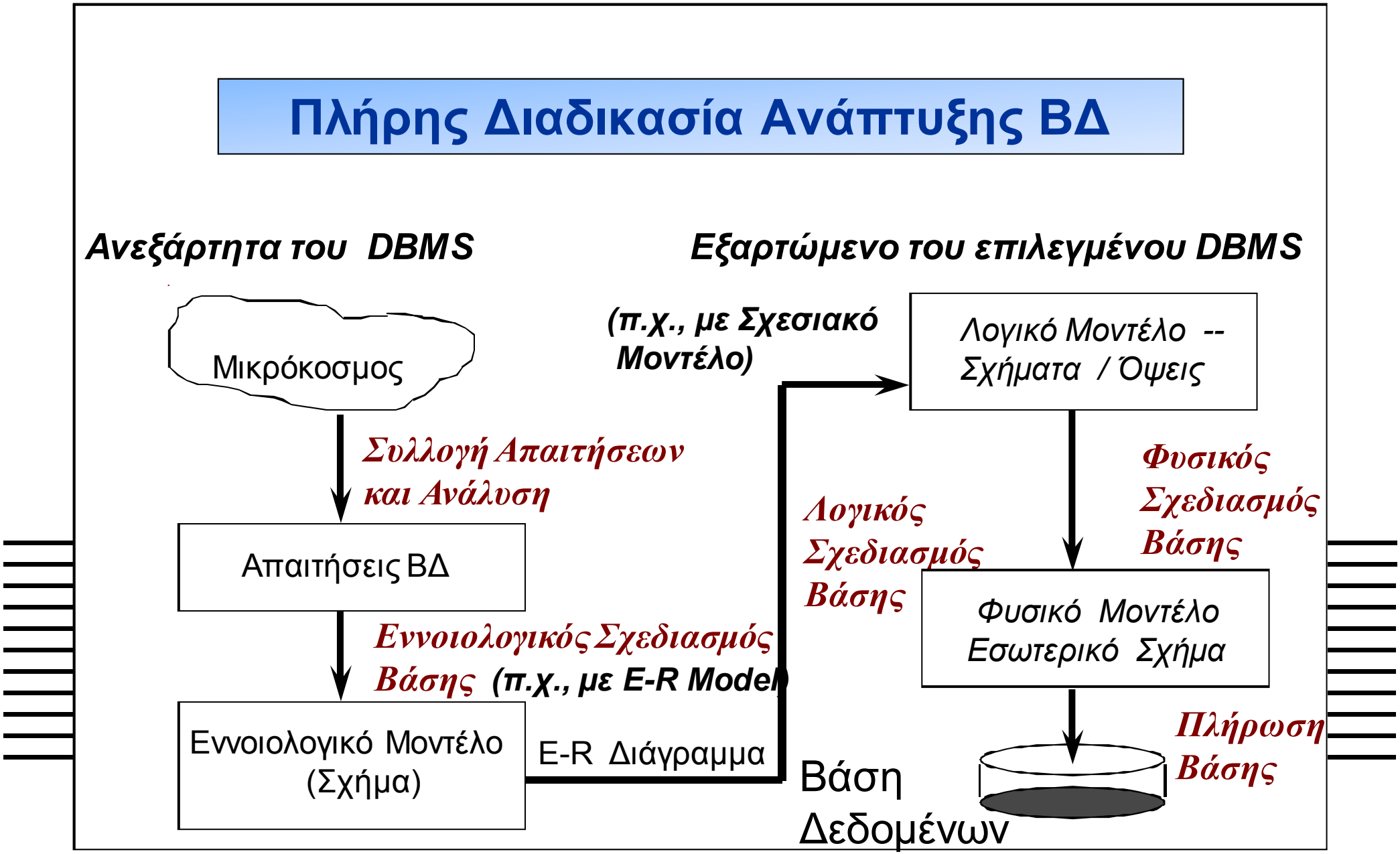
- Η υλοποίηση ευρετηρίων με Δομές Δέντρων είναι ιδανική για RANGE QUERIES και πολύ καλή για EXACT (equality) QUERIES.
- Το ISAM είναι μια Στατική Δομή.
  - Μόνο φύλλα κόμβοι αλλάζουν – απαιτούνται κόμβοι υπερχείλισης
  - Οι αλυσίδες των κόμβων υπερχείλισης επηρεάζουν αρνητικά την επίδοση των αναζητήσεων / ανακλήσεων εγγραφών.
- Το B+ tree είναι μια Δυναμική Δομή.
  - Οι Εισαγωγές / Διαγραφές δεν αλλάζουν την ισορροπία του Δέντρου; $\log_F N$ είναι το κόστος αναζήτησης.
  - Υψηλή εξάπλωση (**F**) σημαίνει ότι το βάθος δεν ξεπερνά το 3 ή 4.
  - Σχεδόν πάντα, η καλύτερη δομή (λαμβάνοντας υπόψη την πρακτική χρήση)

ΣΥΝΟΨΗ για Δομές Ευρετηρίων (2)

- Στην πράξη, 67% πληρότητα είναι ο Μέσος όρος για Αρχεία
- Προτιμάται το B+ - Δέντρο από το ISAM, διότι προσαρμόζεται στην αύξηση του αρχείου πολύ καλύτερα

- Η Συμπίεση Κλειδιών αυξάνει την εξάπλωση και μειώνει το ύψος
- Το Bulk loading είναι πολύ ταχύτερο από τις επαναλαμβανόμενες εισαγωγές όταν δημιουργείται ένα B+ δέντρο για πολλά δεδομένα
- Ένα από τα πλέον βελτιστοποιημένα τμήματα του DBMS.

Πλήρης Διαδικασία Ανάπτυξης ΒΔ



Επισκόπηση του Φυσικού Σχεδιασμού

- 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)

ΚΑΤΑΛΟΓΟΙ ΣΥΣΤΗΜΑΤΟΣ

- 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 ίδιο 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.

Η κατανόηση του φόρτου εργασίας

- Για κάθε Ερώτηση (query) στο workload:
  - 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) …

ΕΠΙΛΟΓΗ ΕΥΡΕΤΗΡΙΩΝ

- ***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

Θέματα για Επιλογή των Ευρετηρίων

- 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 MONO αν υπάρχει σημαντικό query που επωφελείται.

Θέματα για Επιλογή των Ευρετηρίων (2)

- 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.

Παραδείγματα Clustering

B+ tree index στο E.age μπορεί να χρησιμοποιηθεί για την ανάκληση των πλειάδων

```
SELECT  E.dno
FROM    Emp E
WHERE  E.age>40
```

Θεωρήστε το GROUP BY query.

- Αν για πολλές πλειάδες το *E.age* > 10, η χρήση του *E.age* index και μετά η ταξινόμηση του αποτελέσματος είναι αργή - Clustered *E.dno* index ίσως είναι καλύτερη

```
SELECT E.dno, COUNT (*)
FROM   Emp E
WHERE E.age> 10
GROUP BY E.dno
```

Ερωτήσεις με Ισότητα

- Clustering στο *E.hobby* βοηθά πολύ!

```
SELECT  E.dno
FROM  Emp E
WHERE  E.hobby=Stamps
```

Ευρετήρια για Πολλαπλά Γνωρίσματα

- Για ανάκληση Emp εγγραφών με *age*=30 AND *sal*=4000, ένα ευρετήριο στο <*age,sal*> είναι καλύτερο από ένα ευρετήριο στο *age* ή /και ένα ευρετήριο στο *sal*.
  - Αυτά τα ευρετήρια αποκαλούνται *ΣΥΝΘΕΤΑ* ευρετήρια
- Αν η συνθήκη είναι: 20<*age*<30 AND 3000<*sal*<5000:
  - Clustered tree index στο <*age,sal*> ή στο <*sal,age*> συνίσταται.
- Αν η συνθήκη είναι: *age*=30 AND 3000<*sal*<5000:
  - Clustered <*age,sal*> index πολύ καλύτερο από το <*sal,age*> index!
- Τα σύνθετα ευρετήρια απαιτούν πολύ χώρο στο Δίσκο.

ΣΥΝΟΨΗ

- Η Ανάπτυξη μιας Βάσης Δεδομένων περιλαμβάνει πολλές φάσεις: *ανάλυση απαιτήσεων, εννοιολογικό σχεδιασμό, λογικό σχεδιασμό, φυσικό σχεδιασμό* και *tuning (ρύθμιση).*
  - Εν γένει, πρέπει να πηγαίνουμε μπρος-πίσω από φάση σε φάση για τον βέλτιστο σχεδιασμό, και αποφάσεις σε κάποια φάση επηρεάζουν τις δυνατότητες στις άλλες φάσεις.
- Κατανοώντας τον τύπο του *φόρτου εργασίας* για την εφαρμογή και τις προσδοκώμενες επιδόσεις, είναι σημαντικό ***προαπαιτούμενο*** για τον καλύτερο φυσικό σχεδιασμό
  - Ποια είναι τα πλέον συχνά /σημαντικά queries? Ποια γνωρίσματα / σχέσεις εμπλέκονται? κλπ.

ΣΥΝΟΨΗ (2)

- Τα ευρετήρια χρησιμοποιούνται για την ταχύτερη εκτέλεση των πράξεων
  - Τα ευρετήρια πρέπει επίσης να ενημερώνονται στις Ενημερώσεις του Αρχείου.
  - Επέλεξε ευρετήρια για να εξυπηρετηθούν όσες δυνατόν περισσότερες Σχέσεις / Αρχεία..
  - Η δημιουργία Συστάδας (Cluster) αποτελεί πολύ σημαντική απόφαση; Μια και ΜΟΝΟ ένα Γνώρισμα ανά Σχέση μπορεί να είναι clustered!.
- Τα στατικά ευρετήρια πρέπει περιοδικά να ξανά-δημιουργούνται
- Τα Στατιστικά στοιχεία στους Καταλόγους Συστήματος πρέπει περιοδικά να ανανεώνονται / ενημερώνονται