Επεξεργασία και Βελτιστοποίηση επερωτήσεων σε Σχεσιακές ΒΔ

Πολλές ευχαριστίες στους Πάνο Βασιλειάδη, Γ. Ιωαννίδη, Τ. Σελλή, Ε. Πιτουρά για την επαναχρησιμοποίηση κειμένων/διαφανειών τους

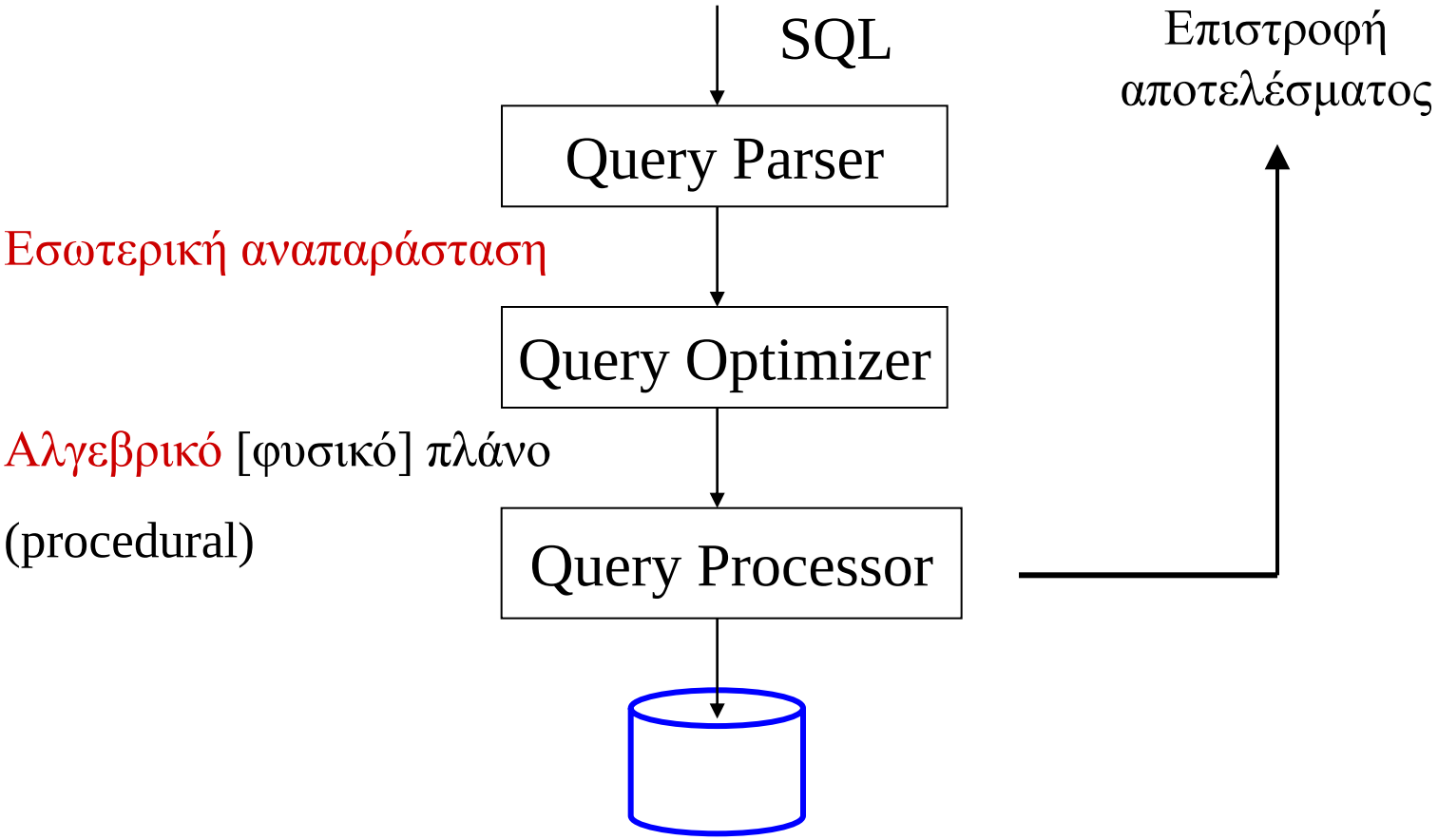
Θεματολόγιο

- Γενικές αρχές της αποτίμησης ερωτήσεων
- Επεξεργασία ερωτήσεων
  - Ερωτήσεις επιλογής
  - Ερωτήσεις σύνδεσης
  - Άλλες ερωτήσεις
- Βελτιστοποίηση ερωτήσεων:
  - Επανεγγραφή ερωτήσεων
  - Παραγωγή εναλλακτικών πλάνων
  - Αποτίμηση πλάνων
  - Πρόβλεψη μεγέθους

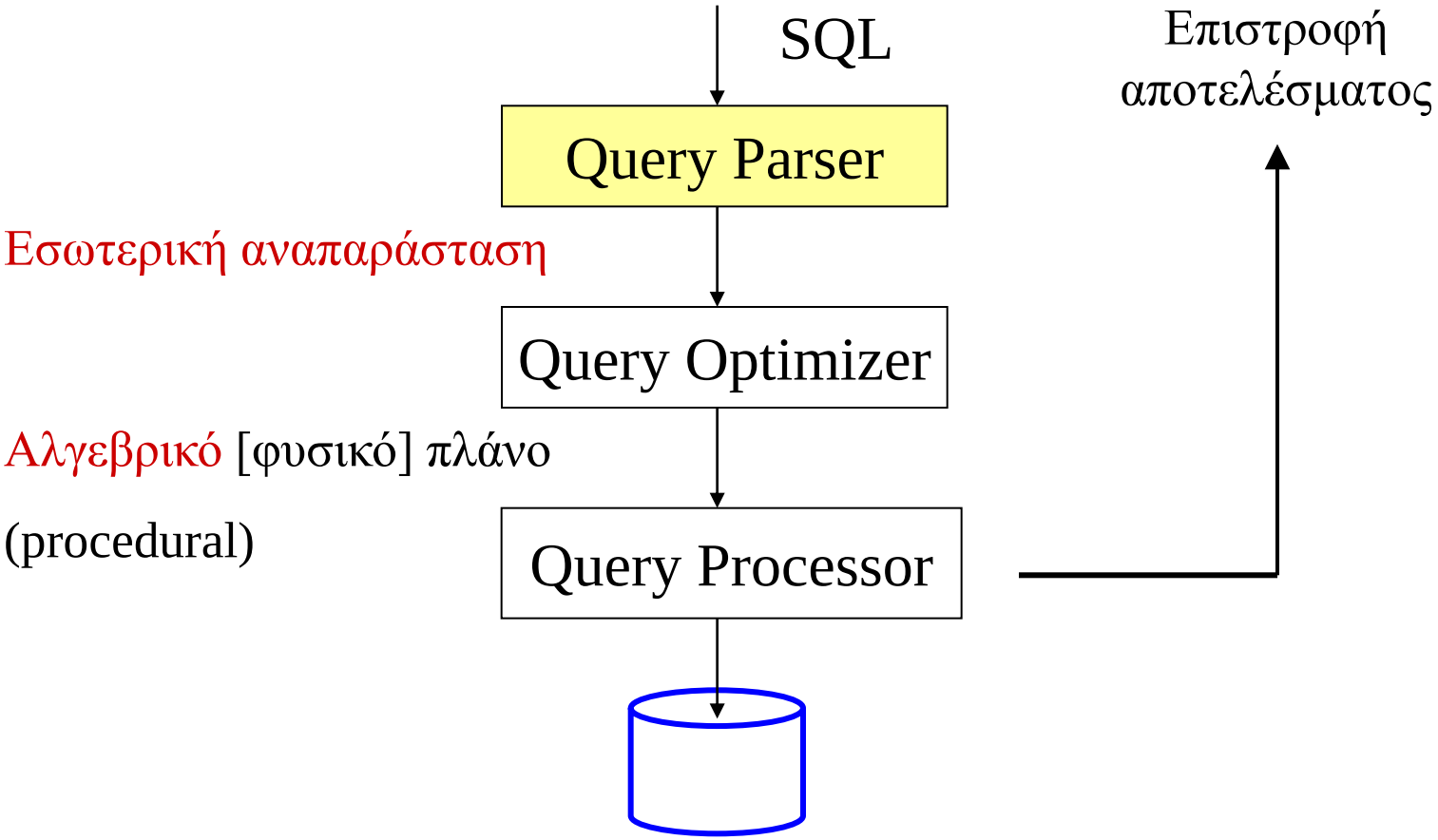
Επεξεργασία ερωτήσεων

- Οι clients θέτουν μια ερώτηση SQL στο server
- Χαρακτηριστικά: δηλωτική γλώσσα + το μόνο που πρέπει να ξέρει ο χρήστης είναι ονόματα πινάκων και γνωρισμάτων
- Το DBMS μέσω της server process κάνει τα εξής:
  - Parsing
  - Έλεγχο συντακτικής ορθότητας
  - Σύνθεση ενός αλγεβρικού πλάνου εκτέλεσης (μέσω ενός δέντρου τελεστών)
  - Εκτέλεση του πλάνου και επιστροφή του αποτελέσματος

Επεξεργασία ερωτήσεων



Επεξεργασία ερωτήσεων



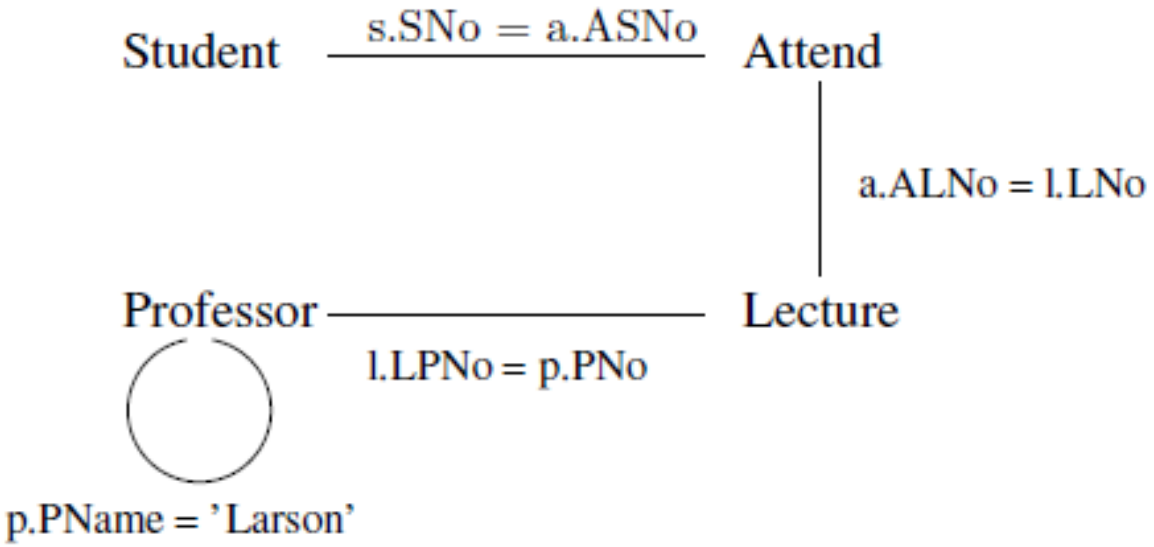
Παράδειγμα Ερώτησης

Student(SNo, SName, SAge, SYear)
Attend(ASNo, ALNo, AGrade)
Lecture(LNo, LTitle, LPNo)
Professor(PNo, PName)

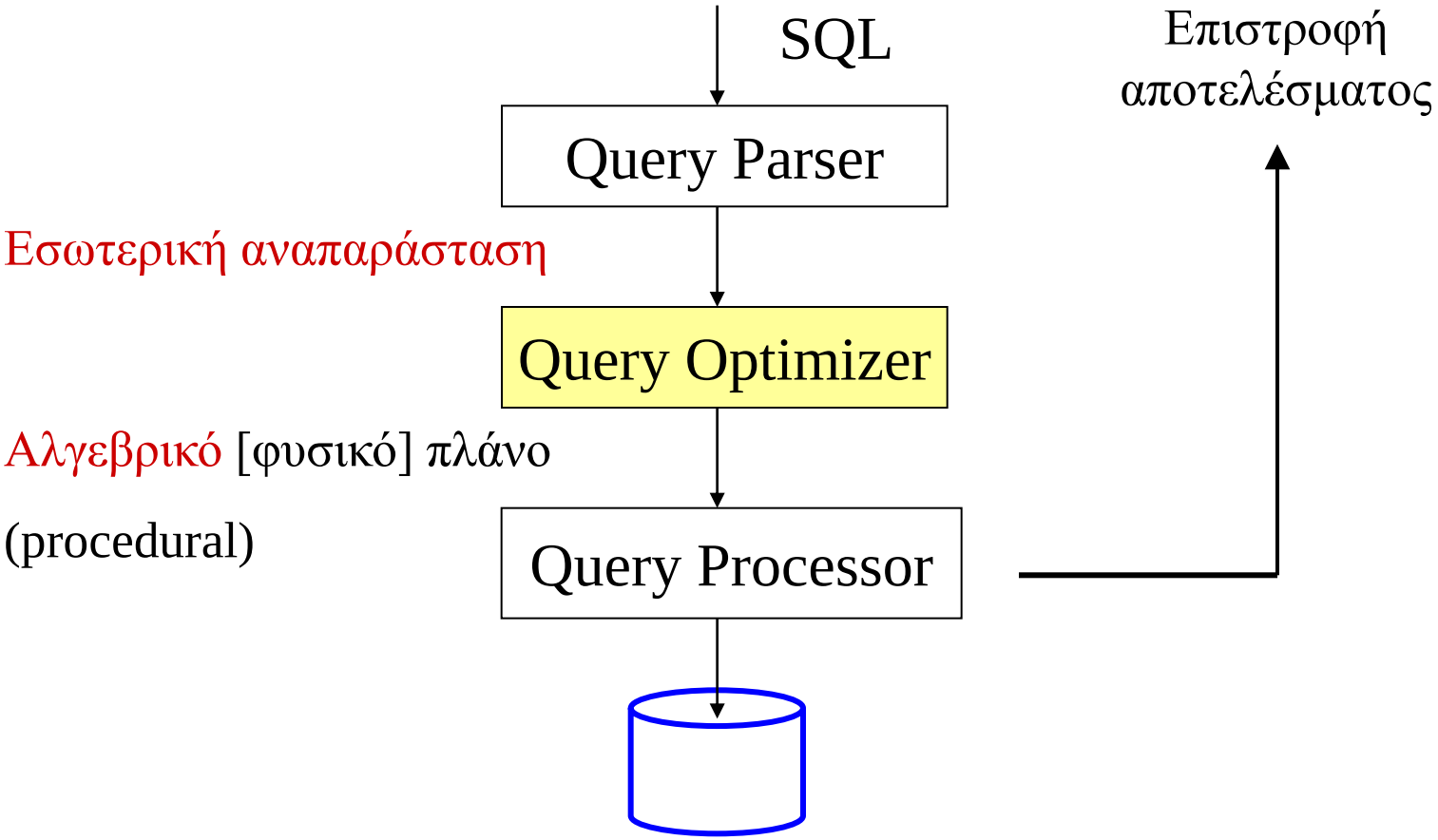
All students attending a lecture by a
Professor Larson

select distinct s.SName
from Student s, Attend a, Lecture l, Professor p
where s.SNo = a.ASNo and a.ALNo = l.LNo
and l.LPNo = p.PNo and p.PName = `Larson'

Query Graph Model



Επεξεργασία ερωτήσεων



Σε τι μας χρειάζεται ο βελτιστοποιητής ?

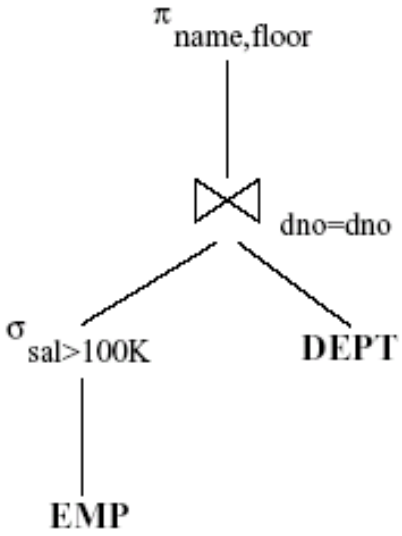
- Υπάρχουν πολλοί δυνατοί τρόποι για να εκτελέσουμε μια ερώτηση
- Μια λάθος επιλογή μπορεί να έχει σημαντικό αποτέλεσμα στην επίδοση μιας ερώτησης
- Σημαντικές παράμετροι:
  - Πώς θα προσπελάσω τα δεδομένα (table scan, full index scan, …)
  - Πώς θα εκτελέσω τις συνδέσεις
    - Λογικά: R ▷◁ S ▷◁ T ή S ▷◁ T ▷◁ R ή ... ?
    - Φυσικά: nested loops, merge join, ... ?

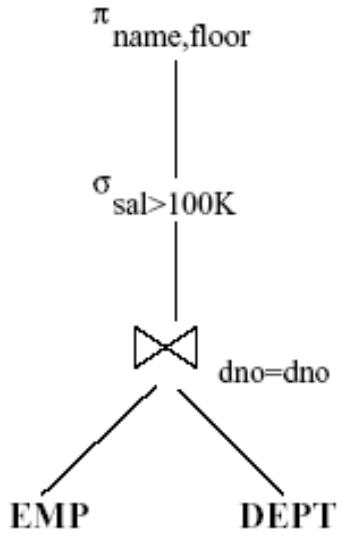
Παράδειγμα - Σχήμα και περιορισμοί

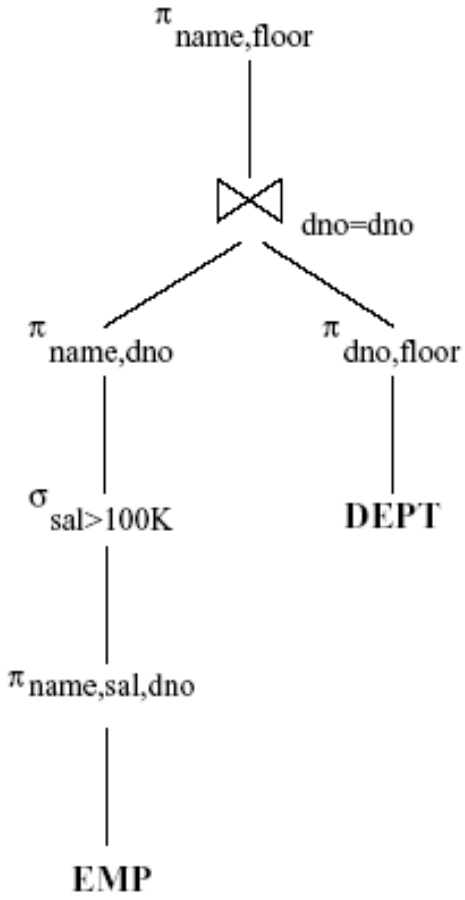
```
emp(name, age, sal, dno)
dept(dno, dname, floor, budget, mgr, ano)
```

- Κάθε υπάλληλος (emp) εργάζεται σε ένα τμήμα (dept)

Παράδειγμα ισοδύναμων, εναλλακτικών πλάνων

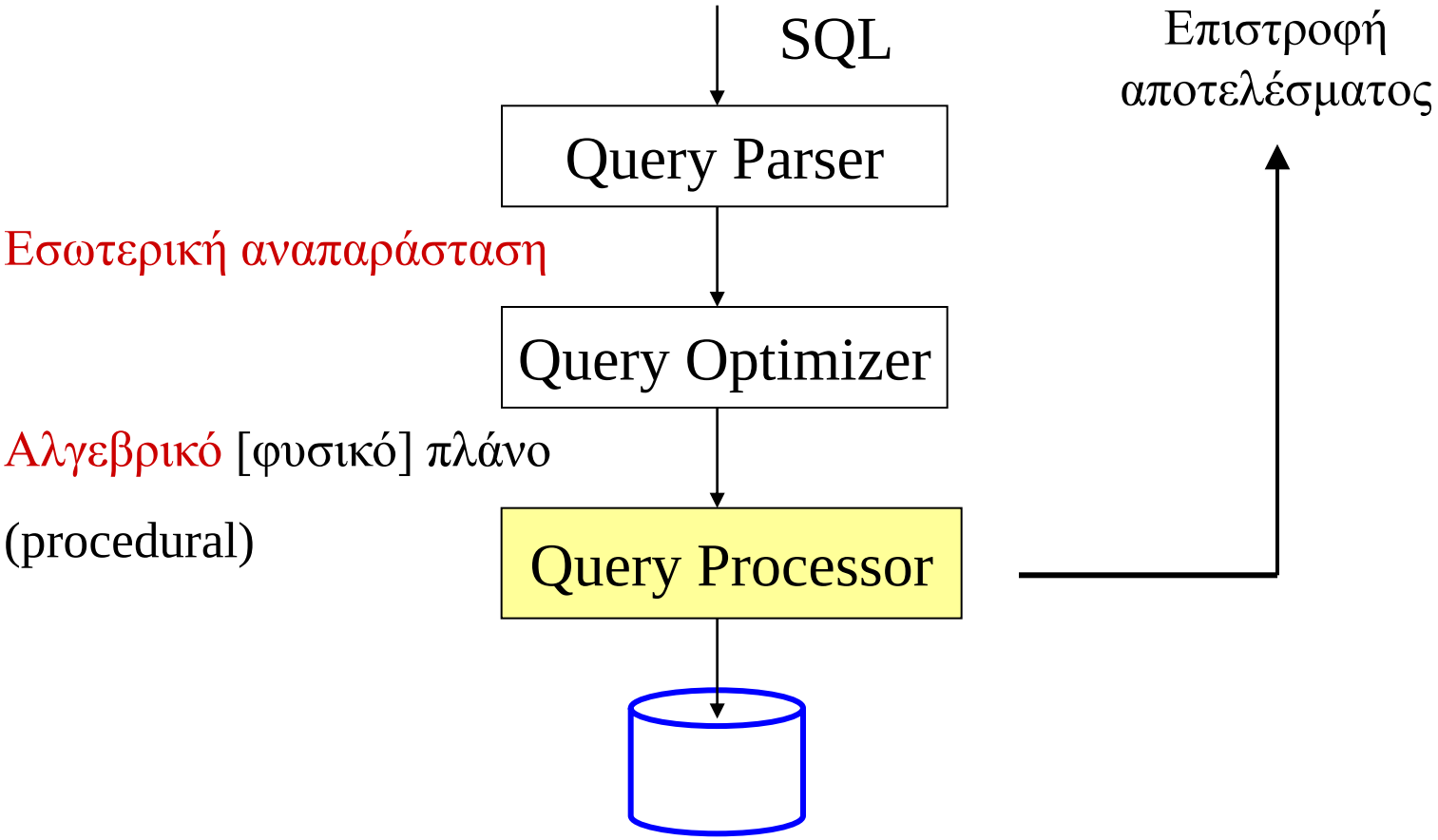







```
select name, floor
from emp, dept
where emp.dno = dept.dno
and sal > 100K
```

Επεξεργασία ερωτήσεων



Επεξεργασία ερωτήσεων

- Δοθέντος του πλάνου, κάθε λογικός τελεστής (π.χ., σ, π, ▷◁) έχει περισσότερους του ενός τρόπους να εκτελεστεί φυσικά (δηλ., στην πράξη).
- Στη συνέχεια θα εξετάσουμε τέτοιους διαφορετικούς φυσικούς τρόπους εκτέλεσης για τους πιο σημαντικούς τελεστές της σχεσιακής άλγεβρας.

Θεματολόγιο

- Γενικές αρχές της αποτίμησης ερωτήσεων
- Επεξεργασία ερωτήσεων
  - Ερωτήσεις επιλογής
  - Ερωτήσεις σύνδεσης
  - Άλλες ερωτήσεις

Notation

- B_R …: αριθμός σελίδων (blocks) μιας σχέσης R
- n …: αριθμός πλειάδων μιας σχέσης
- p_R…: αριθμός πλειάδων ανά block μιας σχέσης R

Επιλεκτικότητα μιας συνθήκης

- **Επιλεκτικότητα** (selectivity) είναι το ποσοστό των εγγραφών της σχέσης που πληρούν την συνθήκη, ήτοι:

$$\text{Sel}(\varphi) = \frac{\#\text{ εγγραφών που πληρούν τη συνθήκη}}{\#\text{ εγγραφών της σχέσης}}$$

- Προσοχή: στη βιβλιογραφία ο όρος *higher selectivity* έχει χρησιμοποιηθεί για δύο αντίθετα πράγματα:
  - Υψηλότερο ποσοστό αποδεκτών εγγραφών (ερμηνεία με βάση τη φόρμουλα)
  - Υψηλότερο ποσοστό απορριπτόμενων εγγραφών (ερμηνεία με βάση τη λογική – όσο πιο επιλεκτικός είσαι, τόσο λιγότερες εγγραφές γίνονται αποδεκτές)

Επιλογή

- Υπάρχουν διάφοροι τρόποι να εκτελέσουμε στην πράξη μια λειτουργία επιλογής (σ), ο οποίοι εξαρτώνται από τους εξής παράγοντες:
  - Είδος ερώτησης
    - Ερώτηση ισότητας $\sigma_{A=k}(R)$ ή εύρους $\sigma_{A<=k}(R)$
    - Απλή συνθήκη, σύζευξη ή διάζευξη απλών συνθηκών
  - Αποθήκευση των δεδομένων
    - Η σχέση στο δίσκο είναι αποθηκευμένη με τρόπο ordered ή unordered
  - Ύπαρξη ευρετηρίου ή όχι
    - B+ tree, hash index
  - Σημασιολογικά χαρακτηριστικά
    - Αν το εμπλεκόμενο πεδίο είναι πρωτεύον κλειδί ή όχι

Απλές ερωτήσεις ισότητας χωρίς ταξινομημένα δεδομένα στο δίσκο

$\sigma_{A=k}(R)$

- Η απλούστερη περίπτωση ερωτήσεων ισότητας (αγγλιστί: equality ή point queries).
- Απαιτείται η σάρωση όλων των blocks της σχέσης
- Κόστος: B_R

- *Αλλάζει κάτι αν το πεδίο **A** είναι πρωτεύον κλειδί?*

Απλές ερωτήσεις ισότητας με ταξινομημένα δεδομένα στο δίσκο

$\sigma_{A=k}(R)$

- Δυαδική αναζήτηση!
- Αν το πεδίο **A** είναι πρωτεύον κλειδί, η απάντηση περιλαμβάνει ακριβώς μία εγγραφή
  - Κόστος: $\log(B_R)$
- Αν όχι, μετά την πρώτη, πρέπει να ανακτήσουμε και τις υπόλοιπες εγγραφές, οι οποίες είναι σειριακά αποθηκευμένες μετά την πρώτη
  - Κόστος: $\log(B_R) + \lceil sel(\varphi) * B_R \rceil - 1$

Απλές ερωτήσεις ισότητας μέσω πρωτεύοντος ευρετηρίου B+

$\sigma_{A=k}(R)$

- Ψάχνουμε όλο το δέντρο **I** μέχρι τα φύλλα. Μετά φέρνουμε όσες σελίδες δεικτοδοτούνται από το κατάλληλο φύλλο.
- Αν το πεδίο **A** είναι πρωτεύον κλειδί
  - Κόστος: height(I) + 1
- Αν όχι,
  - Κόστος: height(I) + cost of retrieving R-tuples
- Το κόστος της ανάκτησης των πλειάδων της R που πληρούν την συνθήκη εξαρτάται από το αν είναι ταξινομημένες ή όχι από $\lceil (sel(\varphi) * B_R) \rceil$ ως $\lceil (sel(\varphi) * n) \rceil$

Απλές ερωτήσεις ισότητας μέσω πρωτεύοντος hash ευρετηρίου $\sigma_{A=k}(R)$

- Αλλάζει το κόστος στο ευρετήριο. Συνήθως θέλουμε 1 – 2 I/O για να φτάσουμε στον κάδο με τους δείκτες στις σελίδες της σχέσης R.

Ερωτήσεις εύρους (range queries)

$\sigma_{A<=k}(R)$

- Ενδιαφέρον παρουσιάζει η ορθή πρόβλεψη/εκτίμηση του μεγέθους του αποτελέσματος.
  - Default: |result| = n / 2
- Αν ξέρουμε [min, max] του εύρους τιμών του **A**:

$$|result| = \begin{cases} 0, & k < min \\ n, & k >= max \\ (k-min)/(max-min), & \text{αλλιώς} \end{cases}$$

- Πλέον, διαθέτουμε ιστογράμματα που μας επιτρέπουν αξιοπρεπέστερη πρόβλεψη…

Θεματολόγιο

- Γενικές αρχές της αποτίμησης ερωτήσεων
- Επεξεργασία ερωτήσεων
  - Ερωτήσεις επιλογής
  - Ερωτήσεις σύνδεσης
  - Άλλες ερωτήσεις

- Nested Loops Join
- Sort-Merge Join
- Hash Join

Σχήμα αναφοράς

Sailors (*sid*: integer, *sname*: string, *rating*: integer, *age*: real)
Reserves (*sid*: integer, *bid*: integer, *day*: dates, *rname*: string)

- Reserves:
  - Κάθε εγγραφή έχει μέγεθος 40 bytes, 100 εγγραφές ανά σελίδα, 1000 σελίδες.
- Sailors:
  - Κάθε εγγραφή έχει μέγεθος 50 bytes, 80 εγγραφές ανά σελίδα, 500 σελίδες.

Συνδέσεις Ισότητας πάνω σε Ένα πεδίο

```
SELECT  *
FROM    Reserves R1, Sailors S1
WHERE   R1.sid=S1.sid
```

- Η πράξη της σύνδεσης R ▷◁ S είναι ιδιαίτερα κοινή και από τις πλέον χρονοβόρες => πρέπει να εκτελεσθεί με προσοχή!
- Το R×S είναι πολύ μεγάλο => ΔΕΝ μπορούμε να εκτελέσουμε τη σύνδεση κάνοντας πρώτα R × S και μετά μια επιλογή (σ).

Συνδέσεις Ισότητας πάνω σε Ένα πεδίο

- Έστω: B_R σελίδες στην R, p_R εγγραφές ανά σελίδα, B_S σελίδες στην S, p_S εγγραφές ανά σελίδα.
  - Στα παραδείγματα που ακολουθούν, R είναι η Reserves και S είναι η Sailors.
- *Μετρική Κόστους*: # I/Os (ήτοι, πόσα I/O κάνουμε για να μεταφέρουμε σελίδες από τον δίσκο στην κύρια μνήμη).
  - Θα αγνοήσουμε το κόστος του output.

Σύνδεση Εμφωλευμένων Βρόγχων (Simple Nested Loops Join)

```
foreach page x in R do
        foreach page y in S do
                foreach tuple r in x and s in y
                        if r_i == s_j  then add <r, s> to result
```

- Σύνδεση εμφωλευμένων βρόγχων με σελίδες (Page-oriented Nested Loops join): Για κάθε σελίδα της R, φέρε κάθε σελίδα της S, και επέστρεψε τις εγγραφές <r, s>, όπου η εγγραφή r είναι στην σελίδα της R, η εγγραφή s στην S και οι τιμές τους στα πεδία της σύνδεσης ταιριάζουν.
  - Κόστος: $B_R + B_R * B_S = 1000 + 1000*500$
- Αν η μικρότερη σχέση είναι εξωτερική τότε
  - Κόστος: $B_S + B_R * B_S = 500 + 1000*500$

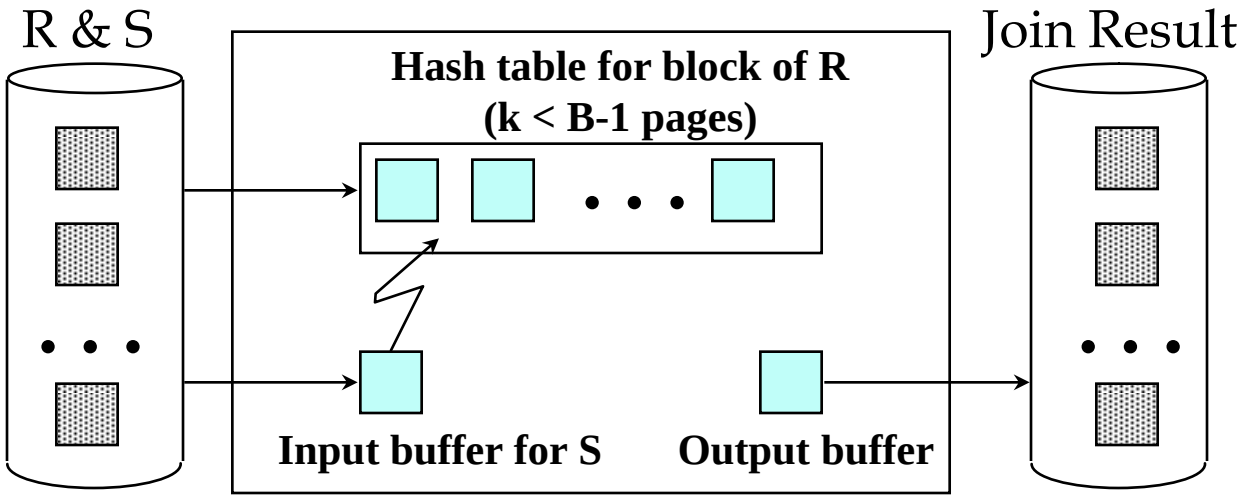
Σύνδεση Εμφωλευμένων Βρόγχων μέσω Ευρετηρίου (Index Nested Loops Join)

```
foreach tuple r in R do
        foreach tuple s in S where r_i == s_j do
                add <r, s> to result
```

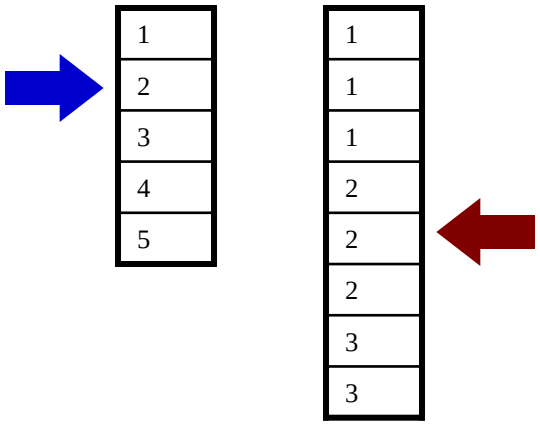
- Αν υπάρχει ευρετήριο στην στήλη στην οποία γίνεται η σύνδεση για κάποια από τις δύο σχέσεις (έστω η S), μπορούμε να κάνουμε την σχέση αυτή εσωτερική και να εκμεταλλευθούμε το ευρετήριο.
  - Κόστος: $B_R + ((B_R * p_R) *$ cost of finding matching S tuples)

Block Nested Loops Join

- Κράτα μία σελίδα σαν input buffer για την εσωτερική S, μία σελίδα σαν buffer για το output, και όλους τους άλλους buffers δώστους στην εξωτερική R (όπου λέμε ότι κρατάμε ένα block της R).



Σύνδεση με Sort-Merge Join




```
SELECT  S.sname, R.rname
FROM  Sailors S, Reserves R
WHERE  R.day='1/1/05' AND R.sid=S.sid
```

Ταξινόμησε τα R, S με βάση το πεδίο sid

Για κάθε r ∈ R [& day='1/1/05']

Για κάθε s ∈ S με ίδιο sid με το r

Επέστρεψε S.sname, R. rname

Ταξινόμηση

- Για να ταξινομήσουμε μια σχέση R σε ένα πεδίο A, μπορούμε να χτίσουμε ένα B+ ευρετήριο πάνω στο πεδίο A και να προσπελάζουμε τη σχέση από το ευρετήριο
- Αν μια σχέση χωρά στην κύρια μνήμη, μπορούμε να την ταξινομήσουμε με τεχνικές όπως η quicksort.
- Στη γενική περίπτωση, όμως, που μια σχέση δεν χωρά στην κύρια μνήμη, η τεχνική που χρησιμοποιούμε ονομάζεται εξωτερική ταξινόμηση (external sorting)

Sort-Merge Join ($R \underset{i=j}{\triangleright\triangleleft} S$)

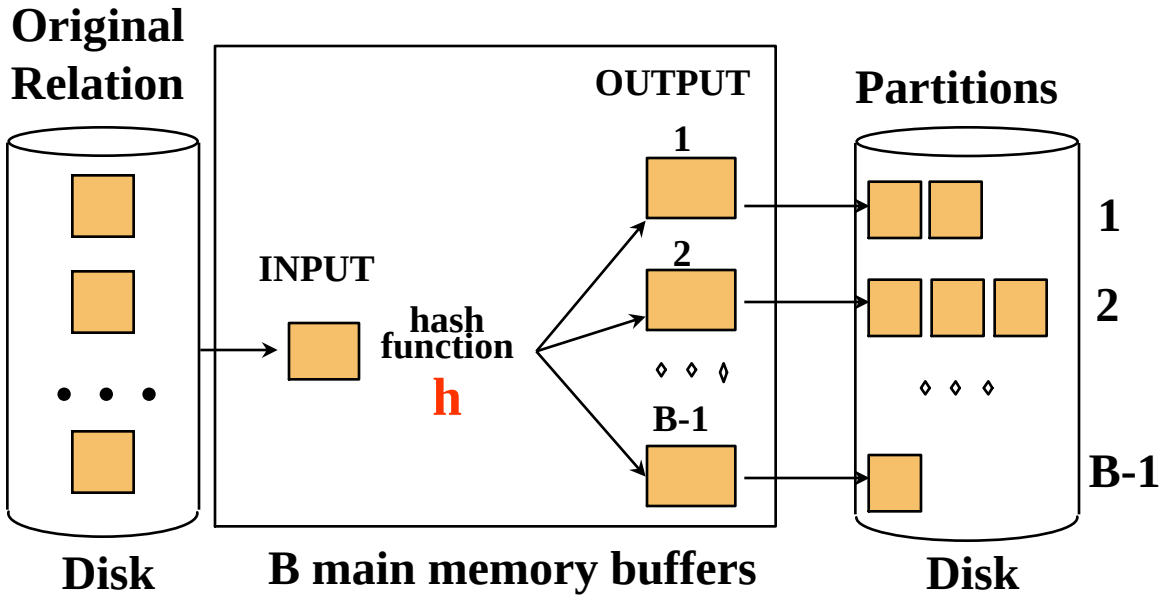
- Ταξινόμησε (Sort) και την R και την S στη στήλη της σύνδεσης, και μετά προσπέλασέ τις, ώστε να τις συνδέσεις (merge) πάνω στην στήλη της σύνδεσης (on join col.) ως εξής:
  - Διάβαζε την R μέχρι η τρέχουσα εγγραφή της R >= τρέχουσας εγγραφής της S.
  - Μετά, διάβασε την S μέχρι η τρέχουσα εγγραφή της S >= τρέχουσας εγγραφής της R tuple.
  - Σταμάτα όταν η τρέχουσα εγγραφής της R = τρέχουσα εγγραφή της S.
  - Τώρα έχουμε μαζέψει εγγραφές της R με την ίδια τιμή στο πεδίο Ri (current R group) και εγγραφές της S με την ίδια τιμή στο πεδίο Sj (current S group). Προφανώς αυτές οι εγγραφές κάνουν match; output <r, s> για όλους τους συνδυασμούς αυτών των εγγραφών.
  - Συνέχισε το παραπάνω μέχρι να τελειώσει μία από τις R και S.

Sort-Merge Join ($R \underset{i=j}{\triangleright\triangleleft} S$)

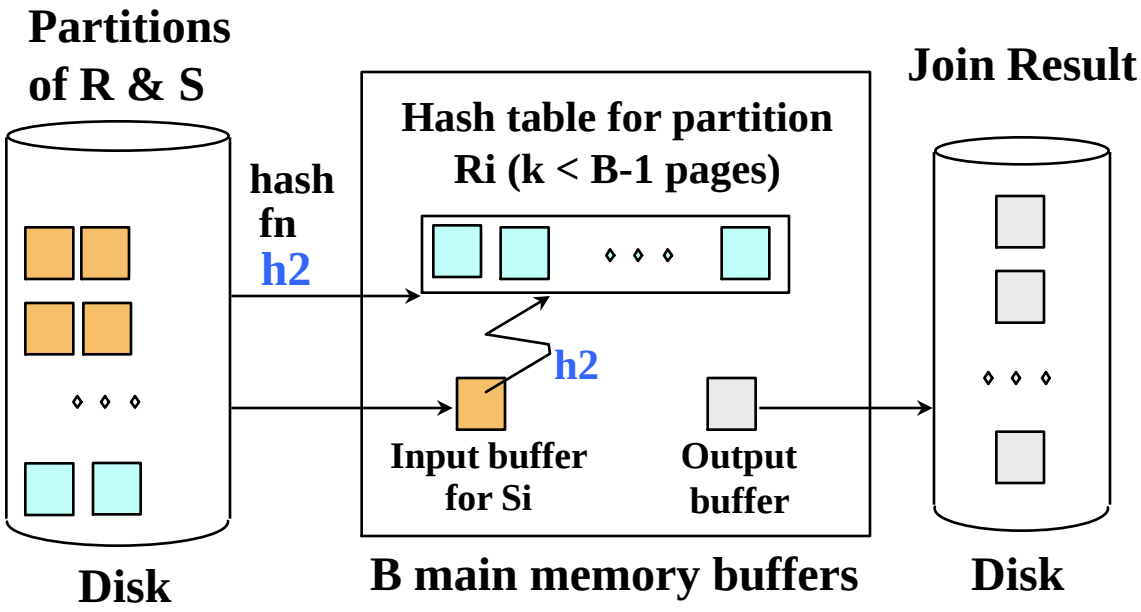
- Διαβάζουμε την R μία φορά.
- Κάθε S-group το διαβάζουμε μία φορά για κάθε εγγραφή του αντίστοιχου R tuple.
- Είναι πιθανό το S group μετά την πρώτη ανάγνωσή του να βρίσκεται ήδη στη μνήμη.
- Κόστος: sort R + sort S + merge R&S
- Κόστος: cost(sort R) + cost (sort S) + ($B_R + B_S$)
  - Το κόστος του merge τυπικά το θεωρούμε $B_R + B_S$ (δηλαδή διαβάζουμε κάθε σελίδα μία φορά), αν και στην χειρότερη περίπτωση μπορεί να φτάσει ως $B_R * B_S$ (όχι πιθανό, όμως!)

Hash-Join

- Σπάσε και τις δύο σχέσεις μέσω της hash function **h**: οι εγγραφές της R στο partition i θα αντιστοιχούν μόνο με της εγγραφές του partition i της S.



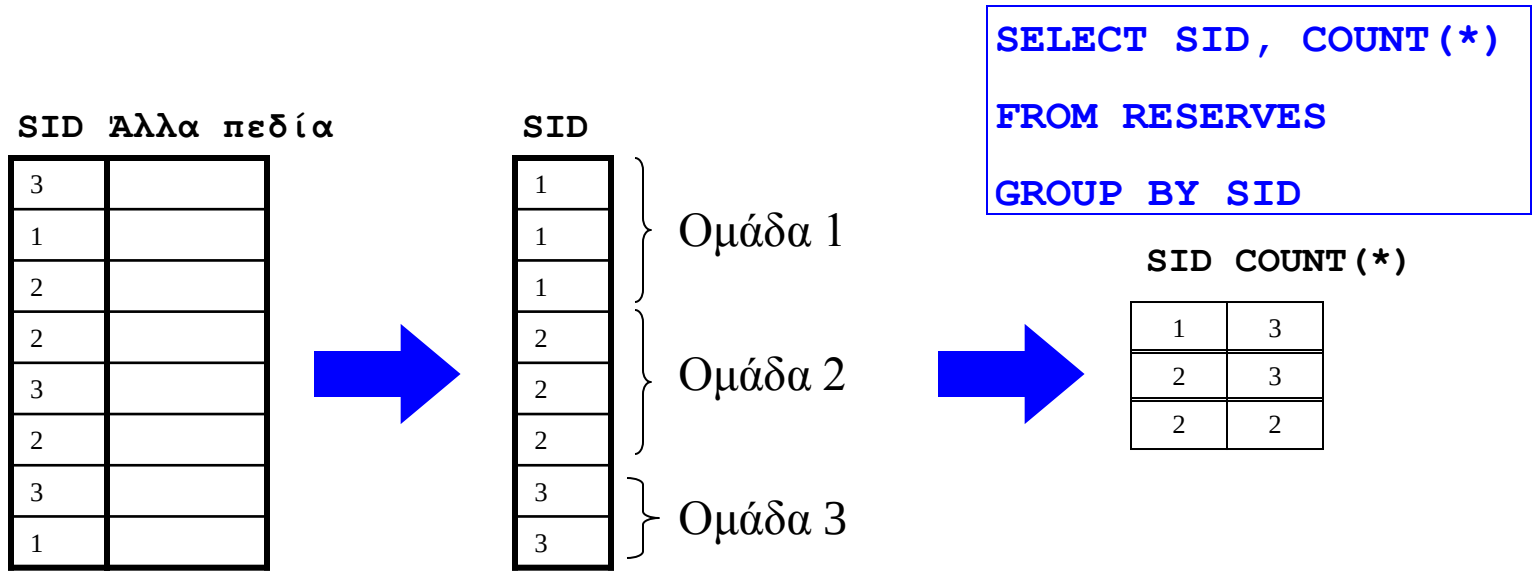
- Διάβασε ένα partition της R, κάνε του hash με μια συνάρτηση **h2 (<> h!)**. Ψάξε το αντίστοιχο partition της S, και βρες τις εγγραφές που κάνουν match.



Θεματολόγιο

- Γενικές αρχές της αποτίμησης ερωτήσεων
- Επεξεργασία ερωτήσεων
  - Ερωτήσεις επιλογής
  - Ερωτήσεις σύνδεσης
  - Άλλες ερωτήσεις

Συνάθροιση

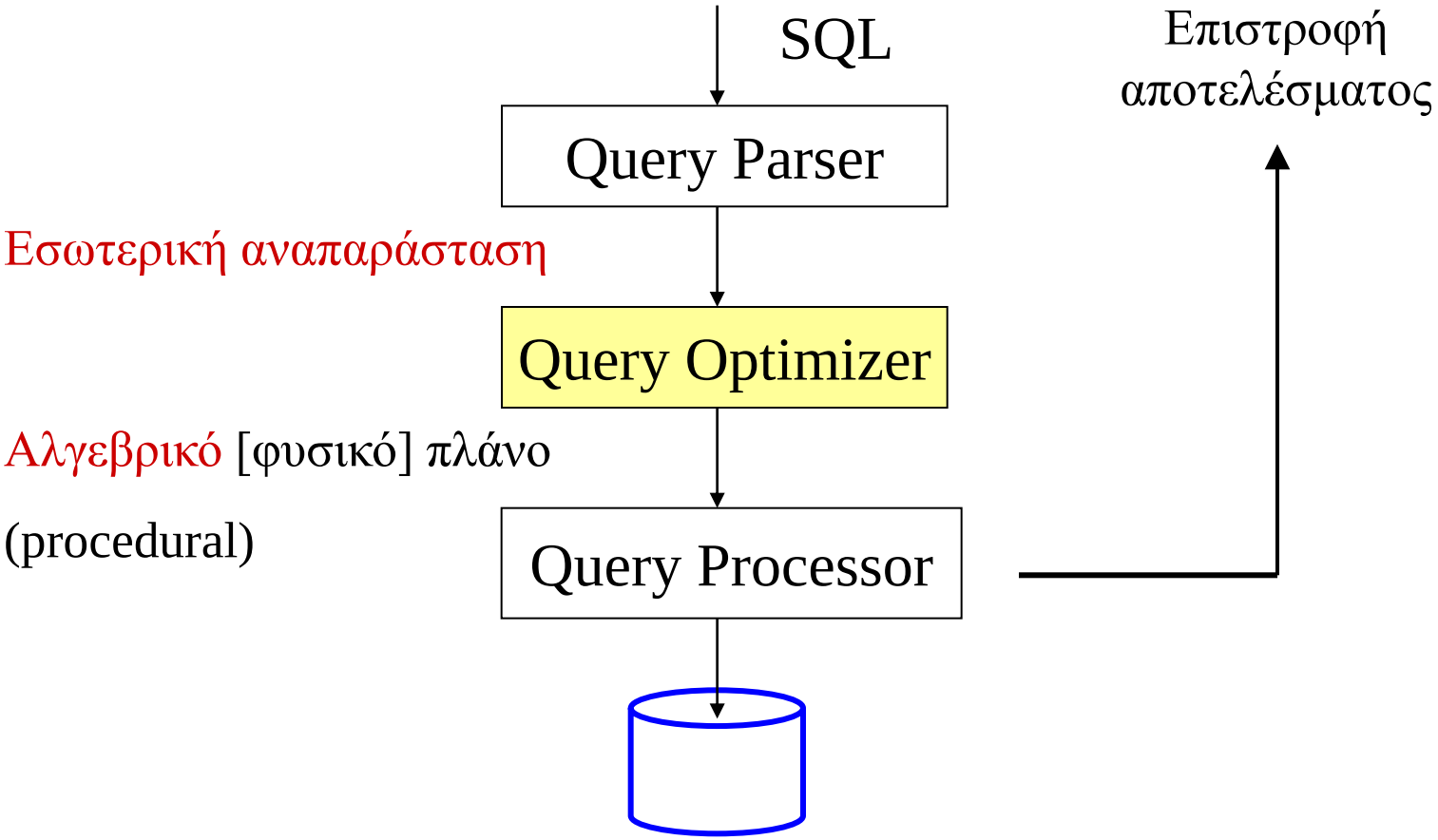


- Όλη η δυσκολία στην συνάθροιση είναι η κατασκευή και η δημιουργία των ομάδων
- Συνήθως χρησιμοποιούμε ταξινόμηση ή hashing για το σκοπό αυτό

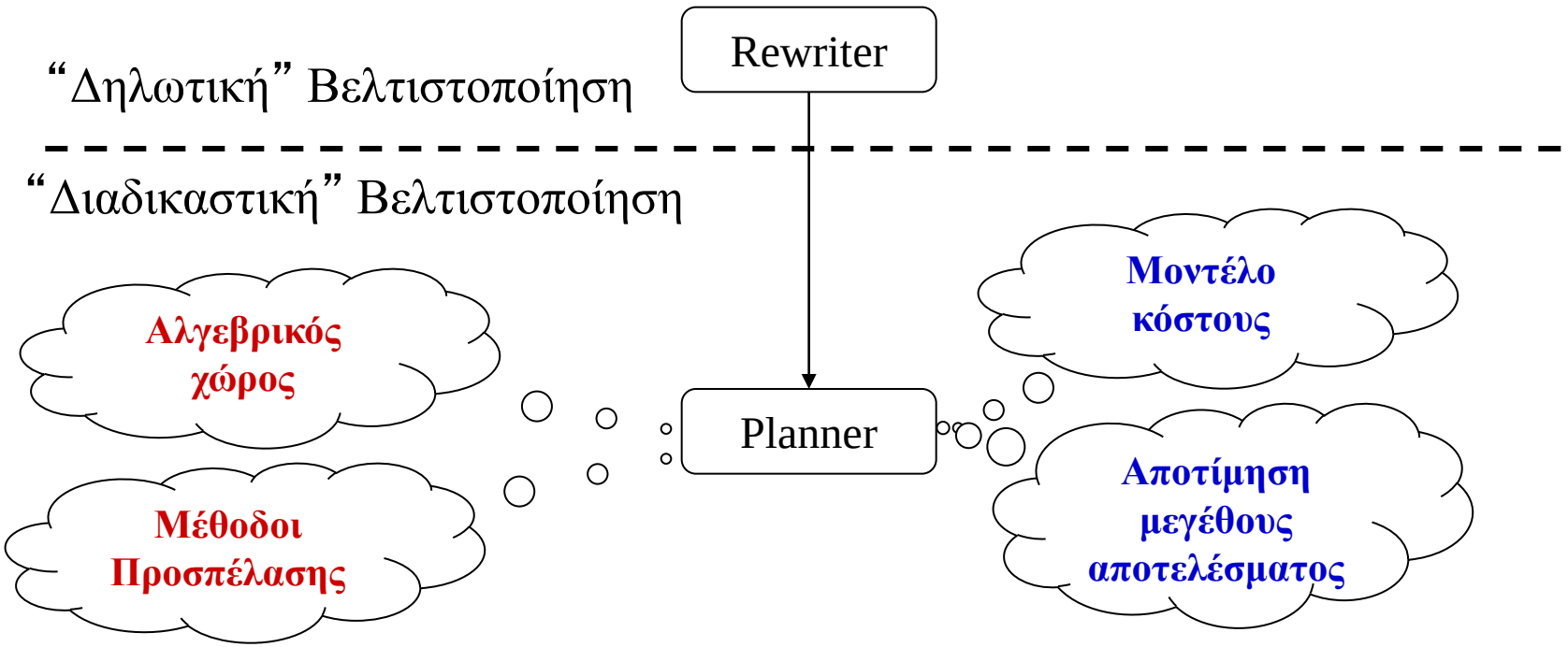
Ανακεφαλαίωση

- Είδαμε ότι μια ερώτηση ανάγεται στον υπολογισμό βασικών τελεστών όπως η επιλογή, η σύνδεση, η συνάθροιση κλπ.
- Ανάλογα με τη φύση των δεδομένων, μπορούμε κάθε φορά να επιλέξουμε και τον ταχύτερο τρόπο εκτέλεσης ενός τελεστή
- Μια αρετή του σχεσιακού μοντέλου και των σχεσιακών DBMS's είναι ότι οι τελεστές μπορούν να συντίθενται.
- Αφού οι τελεστές συντεθούν σε ένα σχεσιακό πλάνο, το ερώτημα παραμένει: ποιο το βέλτιστο πλάνο?
- Η απάντηση δίνεται από τον **βελτιστοποιητή ερωτήσεων**…

Επεξεργασία ερωτήσεων



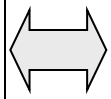
Αφαιρετική δομή του βελτιστοποιητή



Επίπεδα βελτιστοποίησης

- Υπάρχει ένα επίπεδο «δηλωτικής» βελτιστοποίησης, ή επανεγγραφής, όπου παράγουμε λογικά ισοδύναμους τρόπους να εκφράσουμε μια ερώτηση μέσω του **rewriter**

```
select *
from emp
where emp.dno in
        (select dept.dno
         from dept) and
        sal > 100K
```

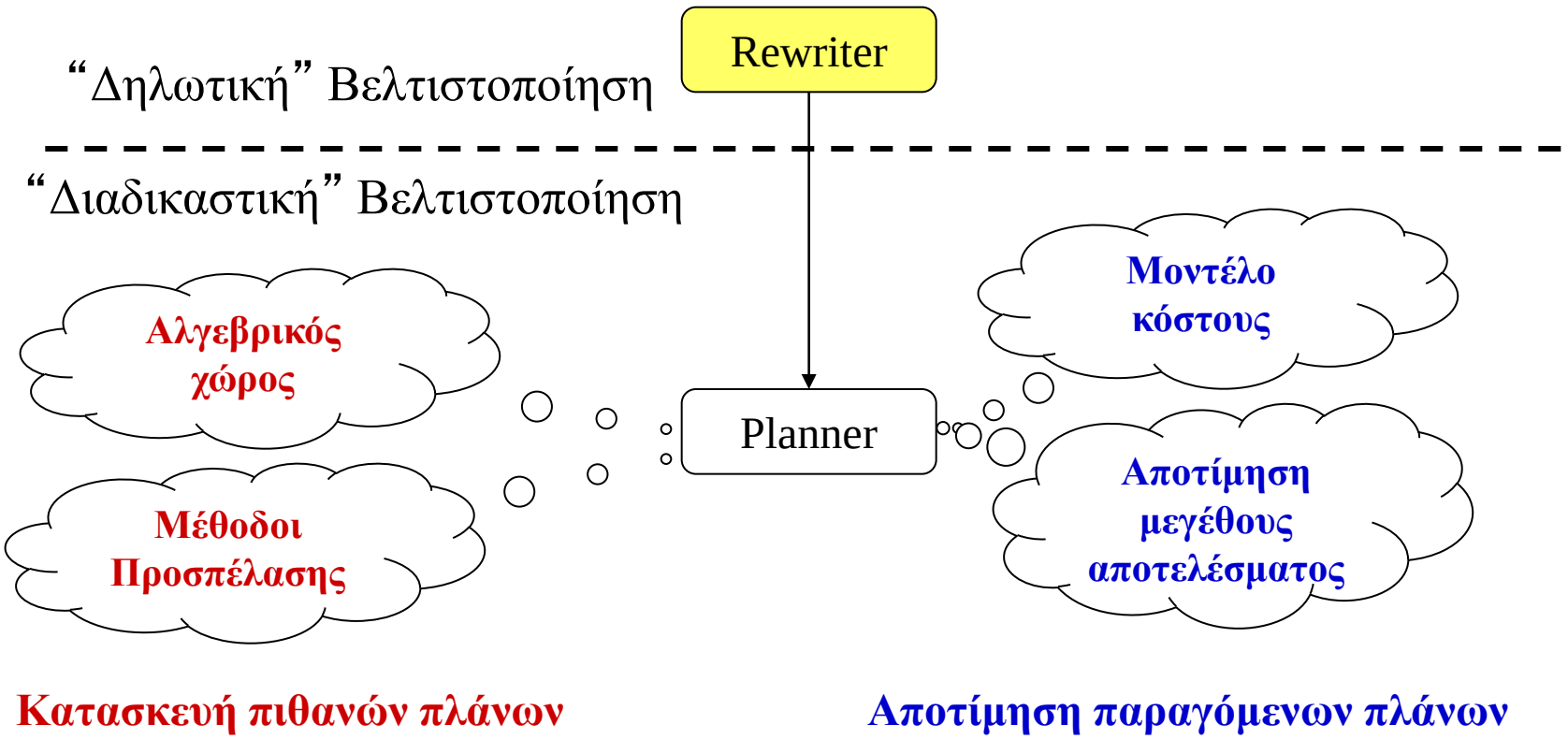


```
select name,age,sal,ndo
from emp, dept
where emp.dno = dept.dno
        and sal > 100K
```

Επίπεδα βελτιστοποίησης

- Υπάρχει ένα επίπεδο «διαδικαστικής» βελτιστοποίησης, όπου παράγουμε (όλα τα) διαφορετικά πλάνα εκτέλεσης μέχρι να διαλέξουμε το πιο αποδοτικό. Η δουλειά αυτή ανατίθεται στον planner.
- Ο planner οφείλει:
  - Να αποφασίσει ποια πλάνα εκτέλεσης θα δημιουργηθούν
  - Ποιο εξ αυτών είναι το καλύτερο

Θεματολόγιο: Επανεγγραφή



Επίπεδα βελτιστοποίησης

- Καθήκοντα του **«rewriter»**
- Αυτό συμπεριλαμβάνει, συνήθως:
  - Μετατροπή εκφράσεων σε «βολική» μορφή
  - Απλοποίηση εμφωλευμένων ερωτήσεων
  - Σημασιολογικά έξυπνες μετατροπές

Ισοδυναμίες της σχεσιακής άλγεβρας

- Επιλογή: $\sigma_{c1 \wedge \ldots \wedge cn}(R) \equiv \sigma_{c1}(\ldots \sigma_{cn}(R))$ *(Cascade)*

  $\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$ *(Commute)*

- Προβολή: $\pi_{a1}(R) \equiv \pi_{a1}(\ldots(\pi_{an}(R)))$ *(Cascade)*

- Σύνδεση: $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$ *(Associative)*

  $(R \bowtie S) \equiv (S \bowtie R)$ *(Commute)*

Σύνθετες ισοδυναμίες

- $\sigma_\theta(\pi_A(R)) \equiv \pi_A(\sigma_\theta(R))$, αρκεί τα πεδία που εμπλέκονται στη συνθήκη θ να είναι υποσύνολο των πεδίων του A
- $\sigma_\theta(R \times S) \equiv (R \triangleright\triangleleft_\theta S)$, αρκεί τα πεδία που εμπλέκονται στη συνθήκη θ να είναι από τις σχέσεις R και S
- $\sigma_\theta(R \triangleright\triangleleft S) \equiv \sigma_\theta(R) \triangleright\triangleleft S$, αρκεί τα πεδία που εμπλέκονται στη συνθήκη θ να είναι αποκλειστικά της R
- $\pi_A(R \triangleright\triangleleft S) \equiv \pi_A,(R) \triangleright\triangleleft S$, με την A' να περιέχει (α) τα πεδία που είχε η A (αρκεί να είναι αποκλειστικά της R) και (β) τα πεδία που εμπλέκονται στη συνθήκη σύνδεσης

Επανεγγραφή Ερωτήσεων (παράδειγμα από IBM DB2)

Distribute NOT

... WHERE NOT(COL1 = 10 OR COL2 > 3)

γίνεται

... WHERE COL1 <> 10 AND COL2 <= 3

Μετασχηματισμοί τιμών:

...WHERE COL = YEAR(`1994-09-08')

γίνεται

... WHERE COL = 1994

Μεταβατική κλειστότητα

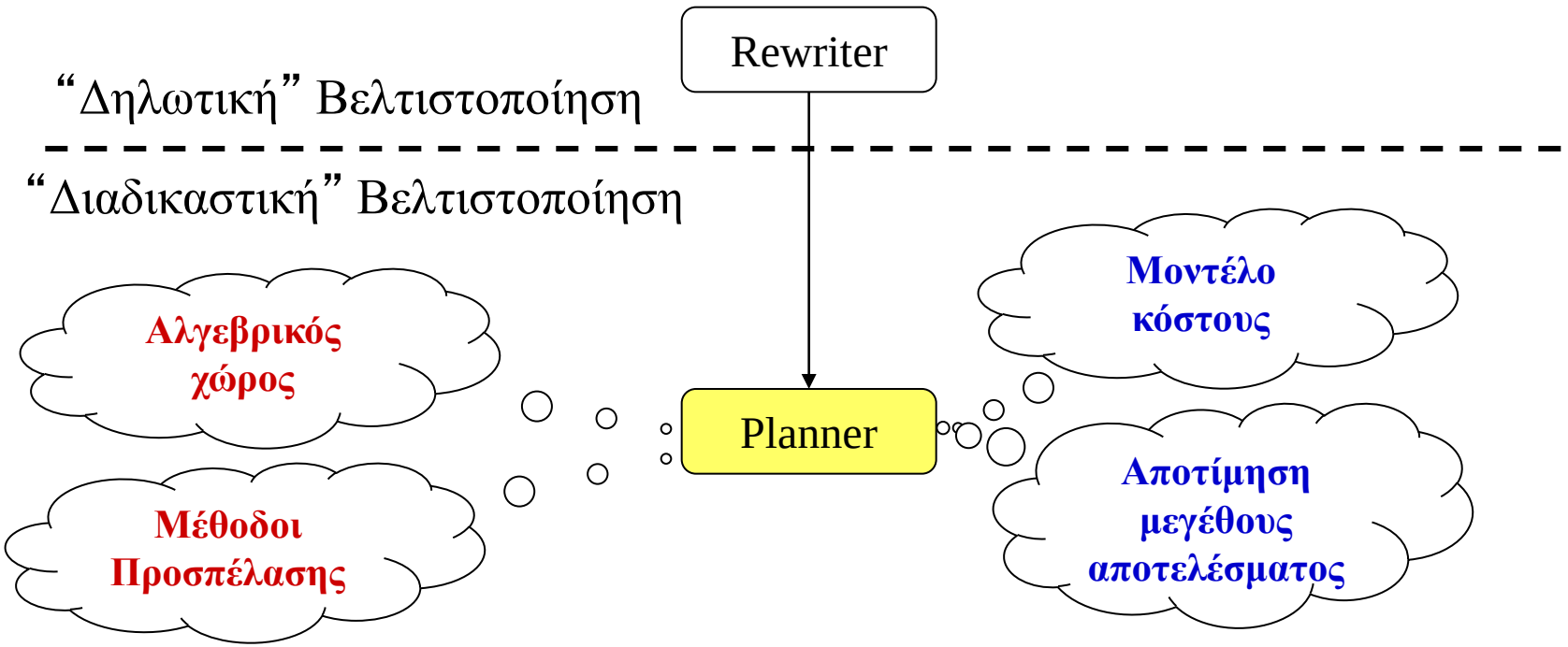
δοθέντος:

T1.C1 = T2.C2, T2.C2 = T3.C3, T1.C1 > 5

προστίθενται...

T1.C1 = T3.C3 AND T2.C2 > 5 AND T3.C3 > 5

Αφαιρετική δομή του βελτιστοποιητή



Επίπεδα βελτιστοποίησης

- Ο planner οφείλει:
  - Να αποφασίσει ποια πλάνα εκτέλεσης θα δημιουργηθούν
  - Ποιο εξ αυτών είναι το καλύτερο

Ο planner οφείλει ...

- Να **κατασκευάσει ένα σύνολο πλάνων**, με βάση
  - Ένα αλγεβρικό χώρο για τη σειρά εκτέλεσης των λειτουργιών (π.χ., να αποφασίσει με ποια σειρά θα κάνει το R ⋈ S ⋈ T)
  - Ένα σύνολο από μεθόδους προσπέλασης στα δεδομένα (π.χ., full-index scan, full table scan, …)
- Να **αποτιμά κάθε πλάνο** που παράγει, μέχρι στο τέλος να βρει το πιο αποδοτικό, με βάση
  - Ένα μοντέλο κόστους που προβλέπει πόσο χρόνο/disk I/O/… κοστίζει το κάθε πλάνο
  - Ένα μοντέλο πρόβλεψης του μεγέθους, κυρίως των ενδιάμεσων αποτελεσμάτων
  - Προσοχή: η αποτίμηση είναι πάντα προσέγγιση/πρόβλεψη και όχι ακριβής υπολογισμός...

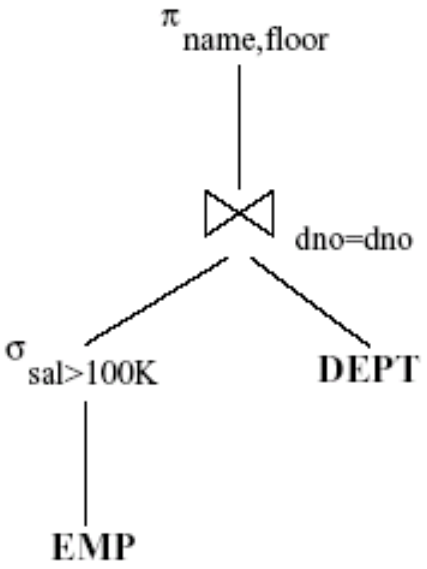
Υποθέσεις

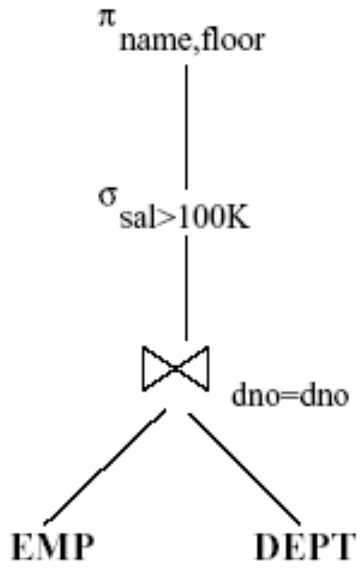
- Θα κάνουμε τις εξής υποθέσεις εργασίας (που αφορούν πρακτικά το σύνολο των DBMS) σε ότι αφορά τις εναλλακτικές λύσεις που θα εξετάσουμε:
- Οι μέθοδοι προσπέλασης που έχουμε είναι (α) πλήρες διάβασμα ενός πίνακα (β) προσπέλαση μέσω ενός ευρετηρίου (συνήθως B+ δέντρο)
- Οι μέθοδοι σύνδεσης που έχουμε είναι (α) nested loops και (β) merge-join, στα οποία χρησιμοποιούμε και των δύο ειδών τις μεθόδους προσπέλασης

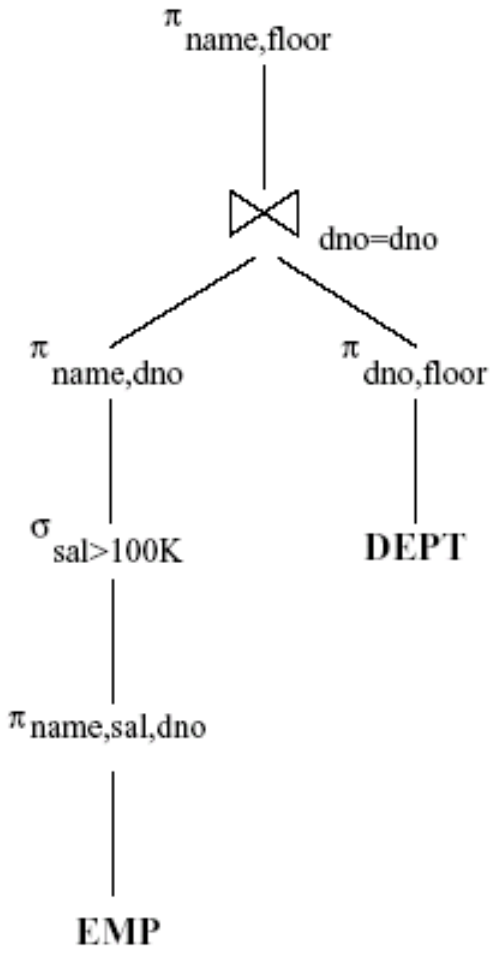
Αλγεβρικός χώρος: τι είναι ένα πλάνο

- Το πλάνο εκτέλεσης μιας SQL ερώτησης είναι ένα **δέντρο**, με:
  - Τις σχέσεις που συμμετέχουν στην ερώτηση, για φύλλα
  - Αλγεβρικούς τελεστές για ενδιάμεσους κόμβους και συγκεκριμένα τους π, σ και ⋈
- Το πλάνο έχει **σειρά εκτέλεσης από κάτω και αριστερά προς τα πάνω**.
- Κοιτώντας ένα ενδιάμεσο κόμβο, ξέρουμε ότι τα παιδιά του έχουν εκτελεστεί και αυτός στέλνει το αποτέλεσμα προς τα πάνω

Πλάνα εκτέλεσης – παράδειγμα








```
select name, floor
from emp, dept
where emp.dno = dept.dno
and sal > 100K
```

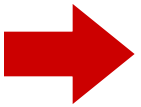
Πλάνα εκτέλεσης

- Για μια απλή SELECT-FROM-WHERE ερώτηση SQL, ο αριθμός των ισοδύναμων εναλλακτικών πλάνων είναι τεράστιος.
- Υπάρχουν κάποιοι λογικοί κανόνες, που επιτρέπουν στον βελτιστοποιητή να μειώσει τον αλγεβρικό χώρο πλάνων

Λογικοί κανόνες βελτιστοποίησης

- Σπρώξε όλες τις επιλογές όσο πιο χαμηλά στο δέντρο γίνεται
- Ενσωμάτωσε τις προβολές μέσα στους άλλους τελεστές

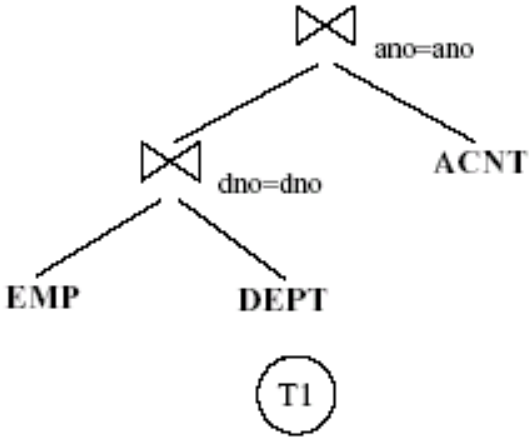
- *... και πάλι όμως, ο αλγεβρικός χώρος παραμένει τεράστιος ...*

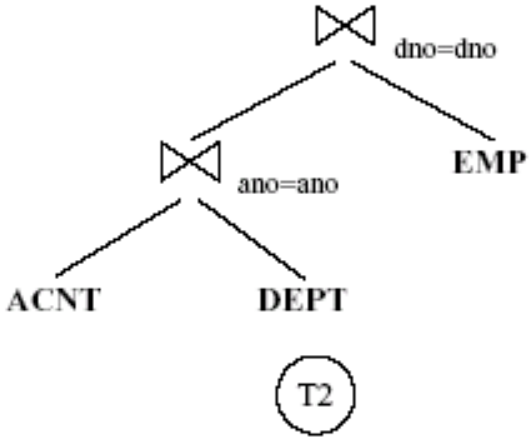


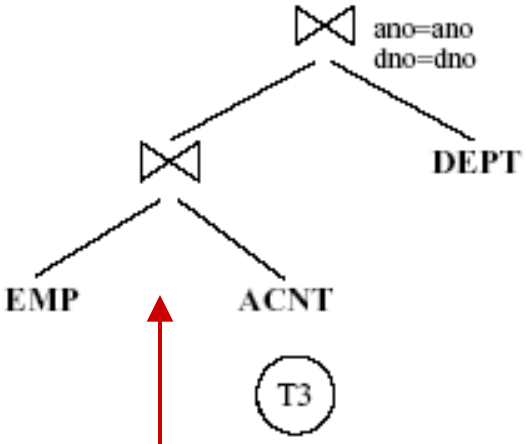
Λογικοί κανόνες βελτιστοποίησης

- Η βασική αιτία είναι οι ιδιότητες της σύνδεσης:
  - $R \bowtie S \Leftrightarrow S \bowtie R$
  - $(R \bowtie S) \bowtie T \Leftrightarrow R \bowtie (S \bowtie T)$
- Το αποτέλεσμα είναι ότι για **N** έχω $(2(n-1))!/(n-1)!$ διαφορετικές διατάξεις join orders for above expression. Με $n = 7$, δίνει 665280, για $n = 10$, πάνω από 176 billion!
- Επιπλέον κανόνας:
  - Ποτέ μην κάνεις καρτεσιανά γινόμενα, εκτός κι αν πρέπει...

Ποτέ να μην επιλέγονται καρτεσιανά γινόμενα





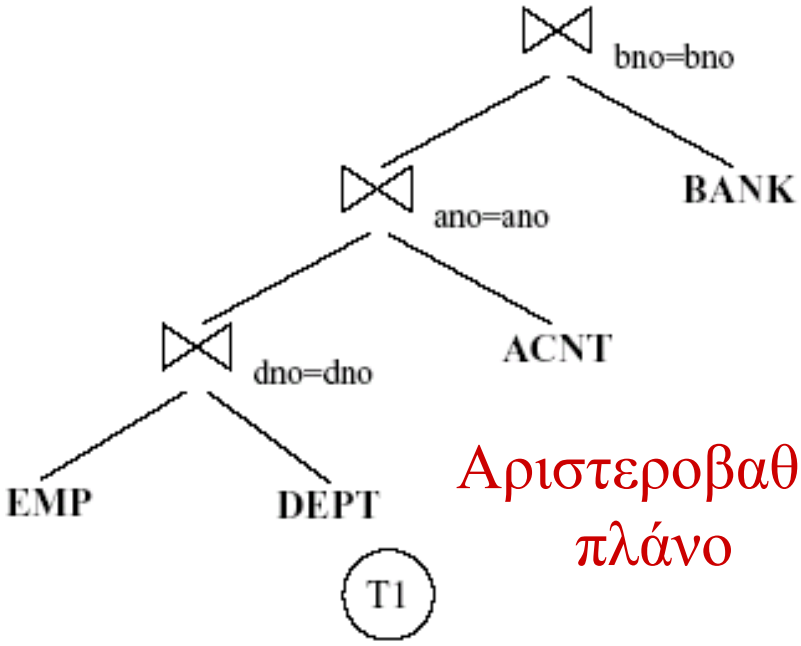



```
select name, floor, balance
from emp, dept, acnt
where emp.dno = dept.dno
and dept.ano = acnt.ano
```

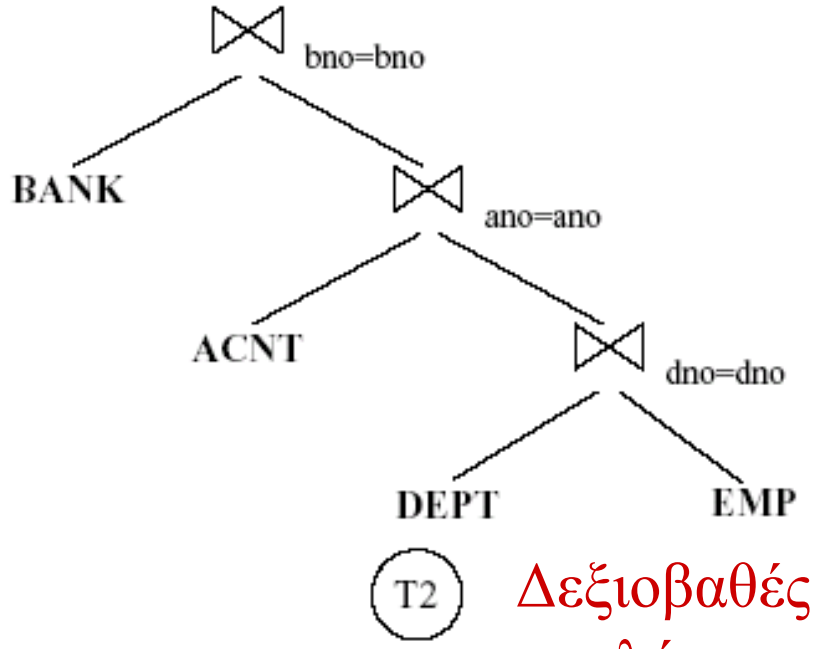
ΠΡΟΣΟΧΗ!

Αριστεροβαθή Δέντρα

- Ακόμα και τώρα όμως, ο αλγεβρικός χώρος είναι μεγάλος
- Όλα τα σύγχρονα DBMS έχουν εισάγει τον ακόλουθο πρακτικό κανόνα:
- Η εσωτερική σχέση ενός τελεστή είναι ΠΑΝΤΑ μια σχέση της ΒΔ και ποτέ ενδιάμεσο αποτέλεσμα!
- Τα δέντρα που προκύπτουν έτσι, λέγονται **αριστεροβαθή** (left-deep)

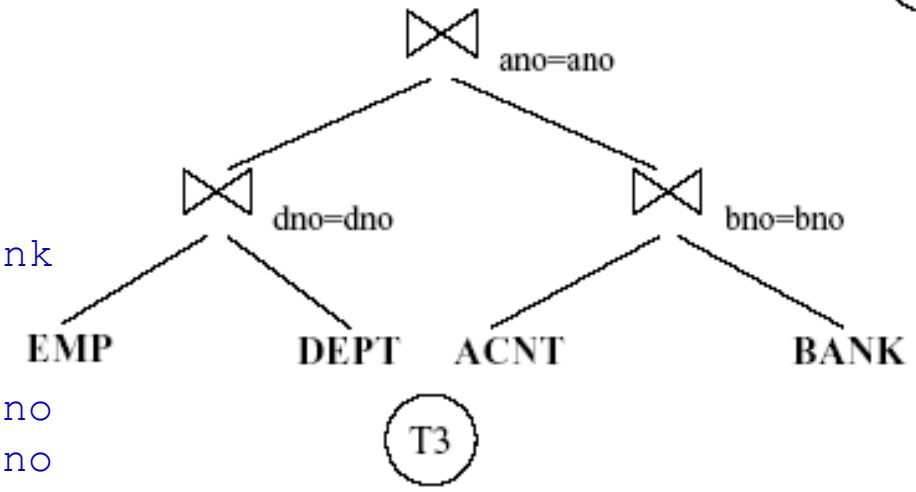


Αριστεροβαθές πλάνο



Δεξιοβαθές πλάνο

```
Select  name, floor,
        balance, bank
from emp,dept,acnt,bank
where
emp.dno = dept.dno
and dept.ano = acnt.ano
and acnt.bno = bank.bno
```



Θαμνώδες (bushy) πλάνο

Αριστεροβαθή Δέντρα

- Κέρδη από αριστεροβαθή δέντρα:
  - Μπορούμε εύκολα να χρησιμοποιούμε ευρετήρια για τις σχέσεις!
  - Τα αποτελέσματα από μια σύνδεση μπορούν να γίνουν pipeline σε μια επόμενη σύνδεση!

Με ποια σειρά?

- Ακόμα δεν αποφασίσαμε τη σειρά των συνδέσεων!

$$(R \bowtie S) \bowtie T \Leftrightarrow R \bowtie (S \bowtie T)$$

- Ο planner, σε όλα τα εμπορικά DBMS χρησιμοποιεί ένα αλγόριθμο δυναμικού προγραμματισμού για να ανακαλύψει τη σειρά

- Προτού δώσουμε το γενικό τρόπο δημιουργίας των πλάνων, **θα κατηγοριοποιήσουμε τις ερωτήσεις** ως:
  - Ερωτήσεις που αφορούν **μία σχέση** στο FROM clause
  - Ερωτήσεις που αφορούν **πολλές σχέσεις** στο FROM clause

Ερωτήσεις με μία σχέση στο FROM clause

- Οι ερωτήσεις αφορούν ένα συνδυασμό προβολών, επιλογών και συναθροίσεων. Η επιλογή του πλάνου γίνεται ως ακολούθως:
  - Εξετάζεται κάθε διαθέσιμη μέθοδος προσπέλασης (file scan / index) και επιλέγεται αυτή με το ελάχιστο κόστος
  - Οι τελεστές εκτελούνται, όσο το δυνατόν γίνεται, μαζί (π.χ., οι προβολές και οι επιλογές ενσωματώνονται στην προσπέλαση μέσω ευρετηρίου).

Εκτίμηση κόστους για πλάνα μίας σχέσης

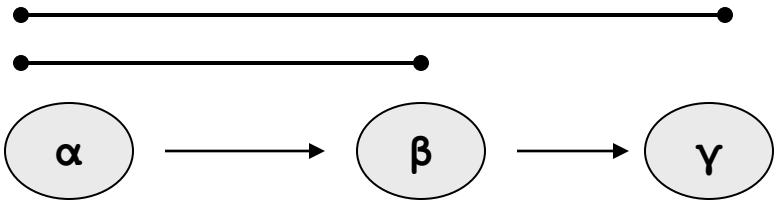
- Index I στο πρωτεύον κλειδί που χρησιμοποιείται για μια επιλογή:
  - Κόστος: Height(I)+1 για B+ tree, περίπου 1.2 για hash index.
- Clustered index I που χρησιμοποιείται για μια ή περισσότερες επιλογές $\sigma_1,\ldots, \sigma_n$:
  - (#Pages(I) + #Pages(R)) * $\prod_i$ (sel(σ_i)), i=1,..,n
- Non-clustered index I που χρησιμοποιείται για μια ή περισσότερες επιλογές $\sigma_1,\ldots, \sigma_n$:
  - (#Pages(I) + #Tuples(R)) * $\prod_i$ (sel(σ_i)), i=1,..,n
- Sequential scan μιας σχέσης:
  - #Pages(R).

Αριστεροβαθή πλάνα για ερωτήσεις πολλών σχέσεων

- Τα αριστεροβαθή πλάνα διαφέρουν
  - στη σειρά των σχέσεων,
  - στη μέθοδο προσπέλασης για κάθε σχέση (index/file scan), και
  - στον τρόπο εκτέλεσης κάθε σύνδεσης (π.χ., nested loop j, hash j, sort-merge j, etc)
- Οι πράξεις ORDER BY, GROUP BY, κλπ., εξετάζονται ως μια τελική πράξη που επικάθεται ενός πλάνου, ενδεχομένως ταξινομώντας το αποτέλεσμα των συνδέσεων αν αυτό δεν είναι ήδη βολικά ταξινομημένο.
- Και πάλι, όμως, ο αριθμός των υπό εξέταση πλάνων είναι εκθετικός σε σχέση με τον αριθμό των εμπλεκόμενων σχέσεων

Δυναμικός προγραμματισμός

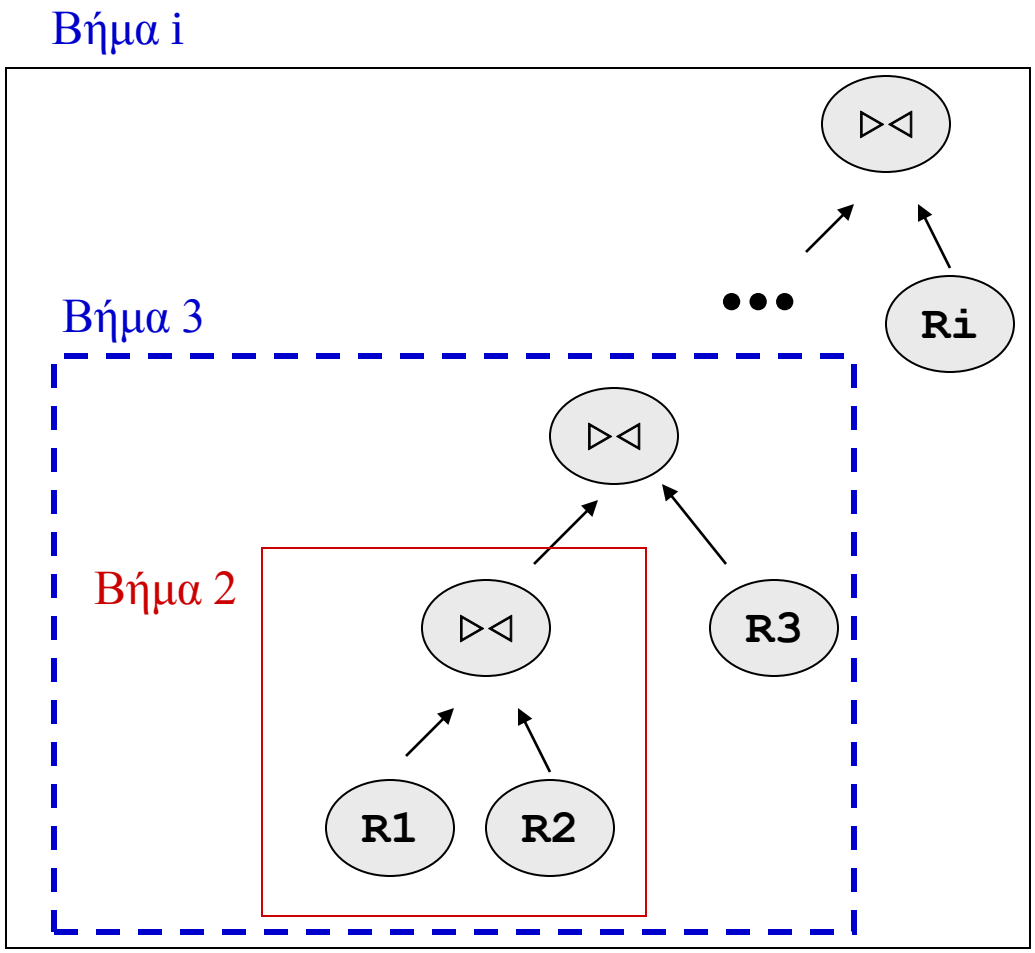
- Εφαρμόζεται σε προβλήματα, στα οποία η λύση μπορεί να εκφρασθεί ως μια ακολουθία αποφάσεων
- Εκμεταλλεύεται το principle of optimality: *μια ακολουθία αποφάσεων (λύση) δεν μπορεί να είναι βέλτιστη, αν μια υπακολουθία της δεν είναι βέλτιστη*



Δυναμικός προγραμματισμός για επεξεργασία ερωτήσεων

- Πρόβλημα: ποια η σωστή σειρά για να εκτελέσω το R ⋈ S ⋈ T?
- Δυναμικός Προγραμματισμός:
  1. Θα βρω όλους τους τρόπους για να προσπελάσω κάθε σχέση χωριστά
  2. Θα πάρω κάθε τέτοιο τρόπο προσπέλασης και θα φτιάξω το καλύτερο υποδέντρο με δύο φύλλα που του αντιστοιχεί
  3. Θα πάρω κάθε τέτοιο υποδέντρο και θα φτιάξω το καλύτερο υποδέντρο με τρία φύλλα που του αντιστοιχεί

Δυναμικός προγραμματισμός για επεξεργασία ερωτήσεων



Εν παραλλήλω, φτιάχνω πολλά δέντρα.

Σιγά σιγά όμως, μειώνω τον αριθμό τους, κρατώντας μόνο το πιο φτηνό για κάθε ενδιαφέρουσα σειρά

Και τι πάει να πει «καλύτερο» πλάνο?

- **Μοντέλο κόστους**: ένα σύνολο αριθμητικών εκφράσεων που μου επιτρέπει να υπολογίζω πόσο θα κοστίσει ο κάθε φυσικός τελεστής
- Π.χ., για το nested loops του R ▷◁ S :
  - Size(R) + Size (S) αν ένα εκ των δύο χωρά στη μνήμη
  - [Size(R) / Size(buffers) - 1] * Size (S), αλλιώς

Και τι πάει να πει καλύτερο ?

- Κι αν έχω ((R ▷◁ S) ▷◁ T), ήτοι, πρώτα το R ▷◁ S και, μετά, το αποτέλεσμά του με το T, τότε τι κόστος θα έχω ?
- Απλό:
  - Cost(R ▷◁ S) = [Size(R) / Size(buffers) - 1] * Size (S)
  - Cost((R ▷◁ S) ▷◁ T) =

    [**Size(R▷◁ S)** / Size(buffers) - 1] * Size (T)

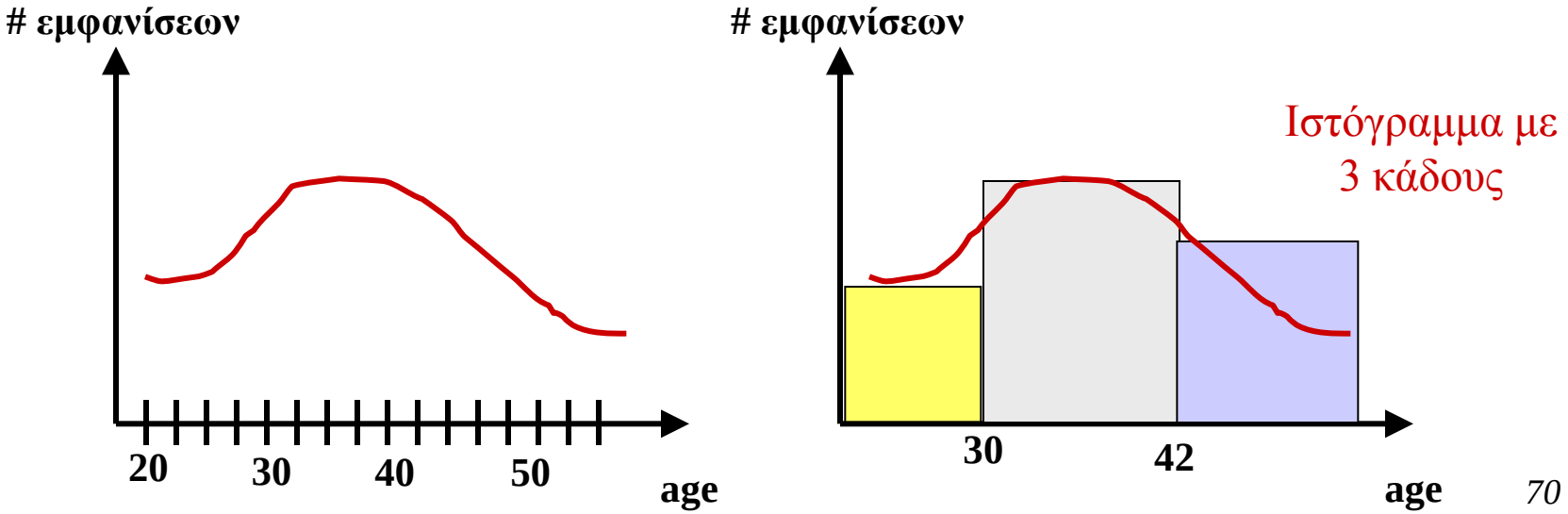
- Size(R▷◁ S)??? Και πού το ξέρουμε αυτό ??

Εκτίμηση μεγέθους

- Για να δουλέψουν οι φόρμουλες κόστους που έχουμε, πρέπει να μπορούμε να αποτιμήσουμε το μέγεθος των ενδιάμεσων αποτελεσμάτων
- Εν γένει, δεν είμαστε πολύ καλοί σ' αυτό, πρακτικά οι τρόποι εκτίμησης που έχουμε δουλεύουν σε δέντρα με ύψος πάνω από 5...
- Η πιο καλή τεχνική που έχουμε είναι τα **ιστογράμματα**

Ιστογράμματα

- Σ’ ένα ιστόγραμμα, διαιρούμε το εύρος των τιμών ενός πεδίου σε κάδους (αγγλιστί: buckets)
- Για κάθε τιμή που παίρνει το πεδίο, μετράω τον αριθμό που αυτή εμφανίζεται



Ιστογράμματα

- Και πώς αποφασίζω **πόσους** κάδους?
  - Ιστογράμματα ίσου πλάτους: κάθε κάδος έχει τον ίδιο αριθμό τιμών στον άξονα των x
  - Ιστογράμματα ίσου ύψος: κάθε κάδος έχει το ίδιο ύψος στον άξονα των y
  - Σειριακά ιστογράμματα: οι συχνότητες ενός κάδου είναι μεγαλύτερες από αυτές του προηγούμενου

Ιστογράμματα

- Αν κάνω μια επιλογή $\sigma_{age > 43}(emp)$ το DBMS μπορεί να εκτιμήσει περίπου πόσες εγγραφές θα μου επιστραφούν
- Αντίστοιχα, αν κάνω μια σύνδεση $R \triangleright\triangleleft S$ πάλι μπορεί να κάνει την αντίστοιχη εκτίμηση ανά ζεύγος κάδων.

Ιστογράμματα

- Είναι σαφές ότι όσο πιο πολλές πράξεις, τόσο πιο πολύ απομακρύνεται η εκτίμηση από την πραγματικότητα...
- (Λανθασμένες) Υποθέσεις εργασίας: οι τιμές των πεδίων είναι ισοπίθανα μοιρασμένες + τα πεδία είναι ανεξάρτητα μεταξύ τους...

Ιστογράμματα

Name	Salary	Department
Zeus	100K	General Manager
Poseidon	80K	Defense
Pluto	80K	Justice
Aris	50K	Defense
Ermis	60K	Commerce
Apollo	60K	Energy
Hefestus	50K	Energy
Hera	90K	General Manager
Athena	70K	Education
Aphro	60K	Domestic Affairs
Demeter	60K	Agriculture
Hestia	50K	Domestic Affairs
Artemis	60K	Energy

Department	Frequency
General Manager	2
Defense	2
Education	1
Domestic Affairs	2
Agriculture	1
Commerce	1
Justice	1
Energy	3

Ιστογράμματα

Department	Απόλυτη Συχνότητα	Συχνότητα Κάδου
Agriculture	1	1.50
Commerce	1	1.50
Defense	2	1.50
Domestic Affairs	2	1.50
Education	1	1.75
Energy	3	1.75
General Manager	2	1.75
Justice	1	1.75

#πραγματικών εμφανίσεων

εκτίμηση ιστογράμματος

- **Ιστόγραμμα ίσου πλάτους**: κάθε κάδος έχει τον ίδιο αριθμό τιμών στον άξονα των x
- Εδώ: **δύο κάδοι**, ο πρώτος από A – D και ο άλλος από E-Z
- Συχν. Κάδου: Σ(x)/count(x), x∈κάδο

Ιστογράμματα

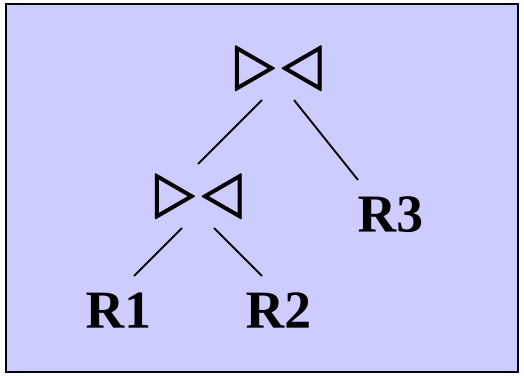
Department	Απόλυτη Συχνότητα	Συχνότητα Κάδου
Agriculture	1	1.33
Commerce	1	1.33
Defense	2	1.33
Education	1	1.33
General Manager	2	1.33
Justice	1	1.33
Domestic Affairs	2	2.50
Energy	3	2.50

#πραγματικών εμφανίσεων

εκτίμηση ιστογράμματος

- **Ιστόγραμμα σειριακό**: οι συχνότητες του δεύτερου κάδου είναι μεγαλύτερες από αυτές του πρώτου

Επανάληψη



Rewriter

"Δηλωτική" Βελτιστοποίηση

"Διαδικαστική" Βελτιστοποίηση

Αλγεβρικός χώρος

Μέθοδοι Προσπέλασης

Planner

Μοντέλο κόστους

Αποτίμηση μεγέθους αποτελέσματος

Κατασκευή πιθανών πλάνων

Αποτίμηση παραγόμενων πλάνων