



Προχωρημένα Θέματα Τεχνολογίας και Εφαρμογών Βάσεων Δεδομένων

Επεξεργασία ερωτήσεων

Πάνος Βασιλειάδης

pvassil@cs.uoi.gr

Απρίλιος 2020

www.cs.uoi.gr/~pvassil/courses/db_III/

Θεματολόγιο

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

Θεματολόγιο

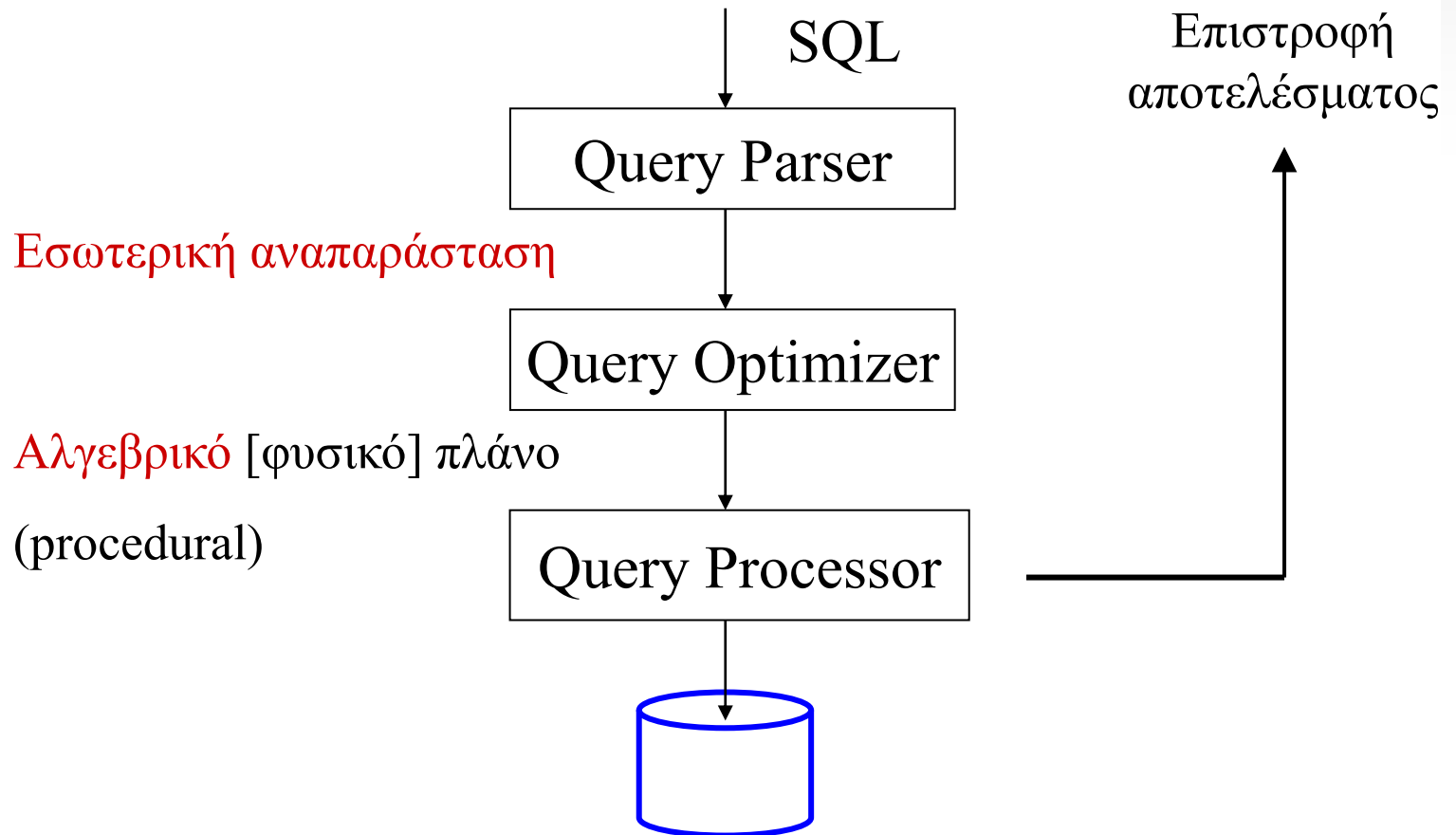
- Γενικές αρχές της αποτίμησης ερωτήσεων
- Επεξεργασία ερωτήσεων
 - Ερωτήσεις επιλογής
 - Ερωτήσεις σύνδεσης
 - Άλλες ερωτήσεις

*Material based on a
TR by Y. Ioannidis*

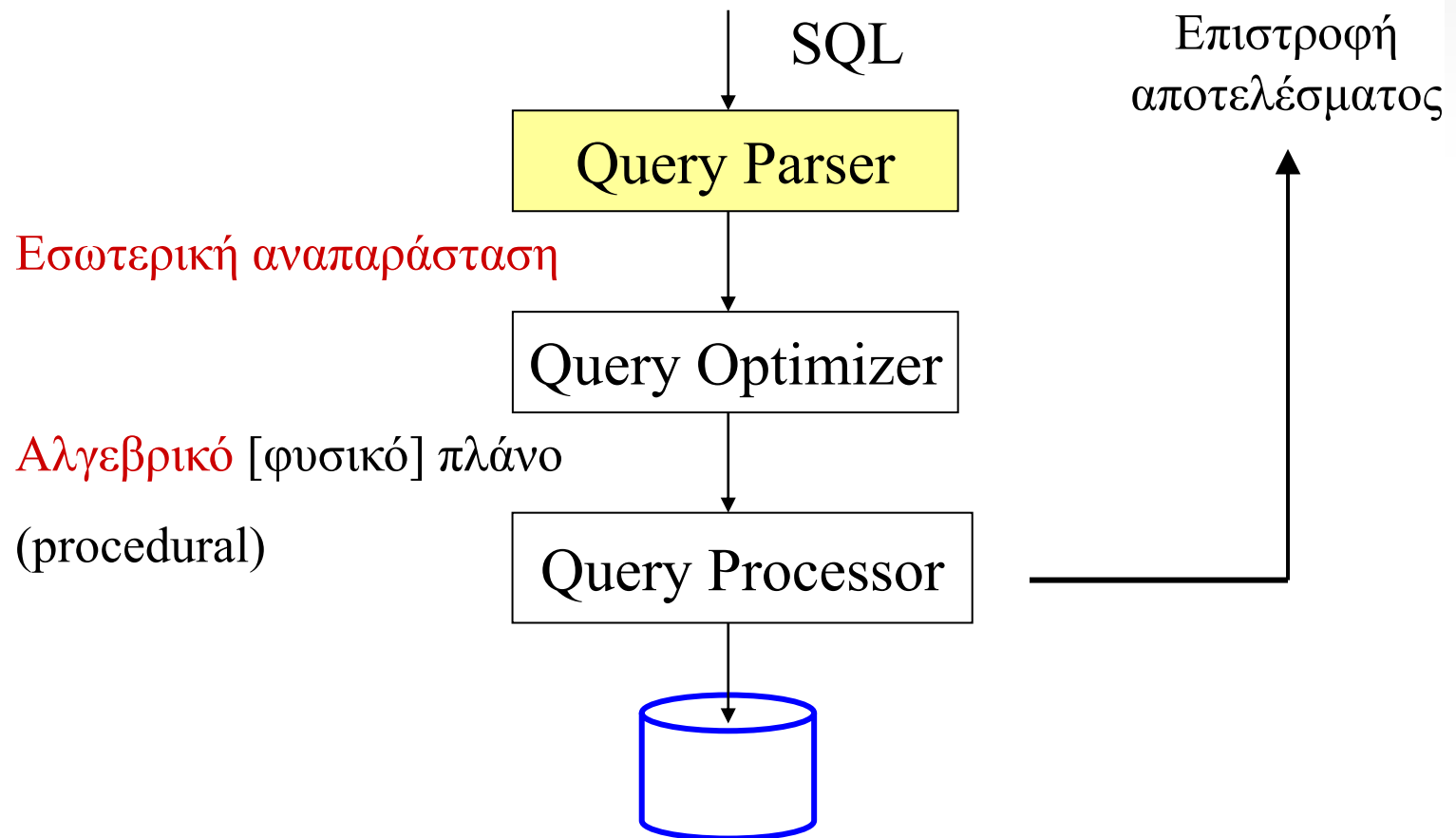
Επεξεργασία ερωτήσεων

- 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

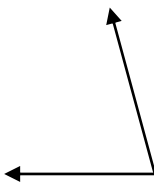


Παράδειγμα Ερώτησης

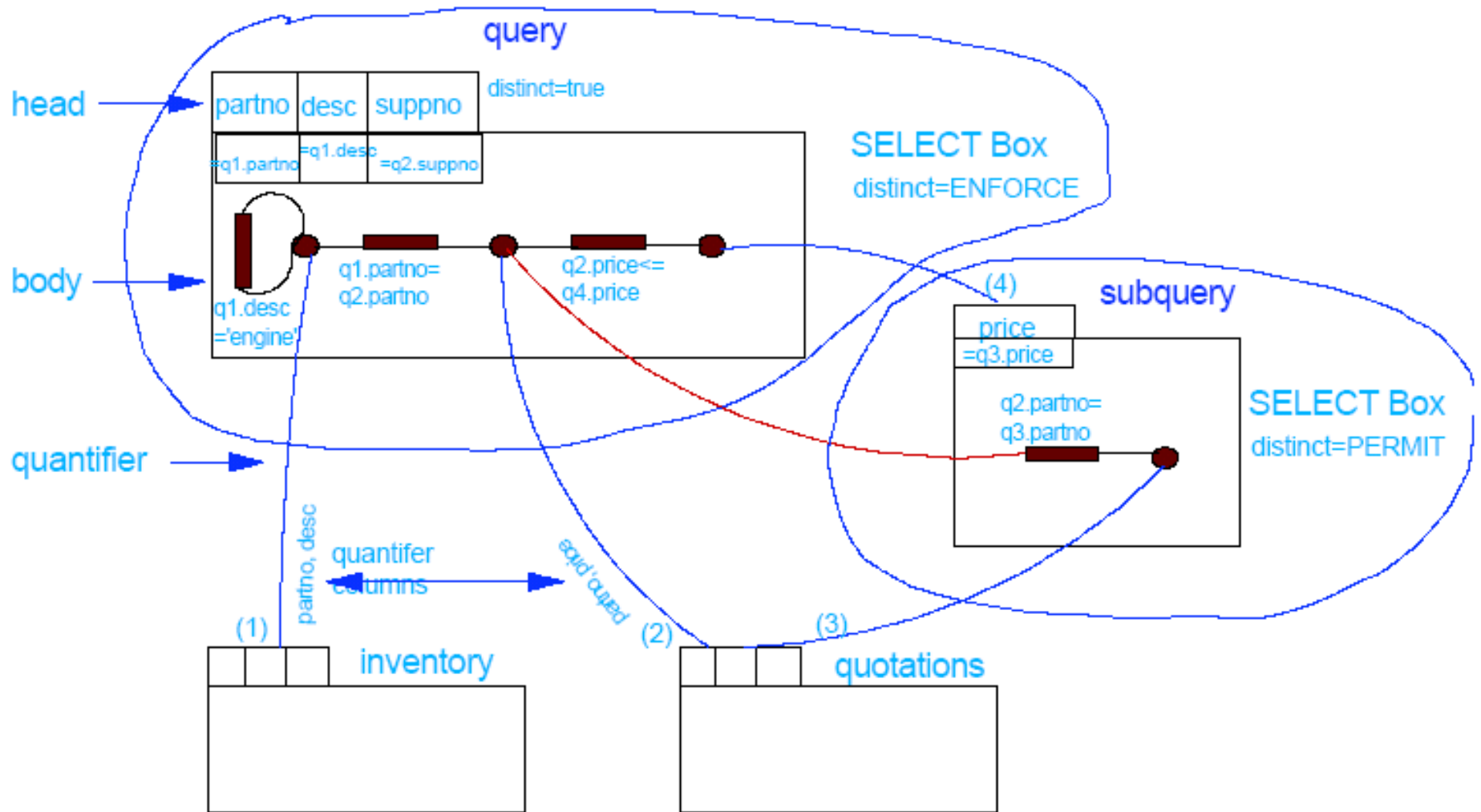
```
SELECT DISTINCT q1.partno, q1.descr, q2.suppno  
FROM inventory q1, quotations q2  
WHERE q1.partno = q2.partno  
      AND q1.descr = 'engine'  
      AND q2.price <= ALL
```

```
( SELECT q3.price  
  FROM quotations q3  
 WHERE q2.partno = q3.partno  
 );
```

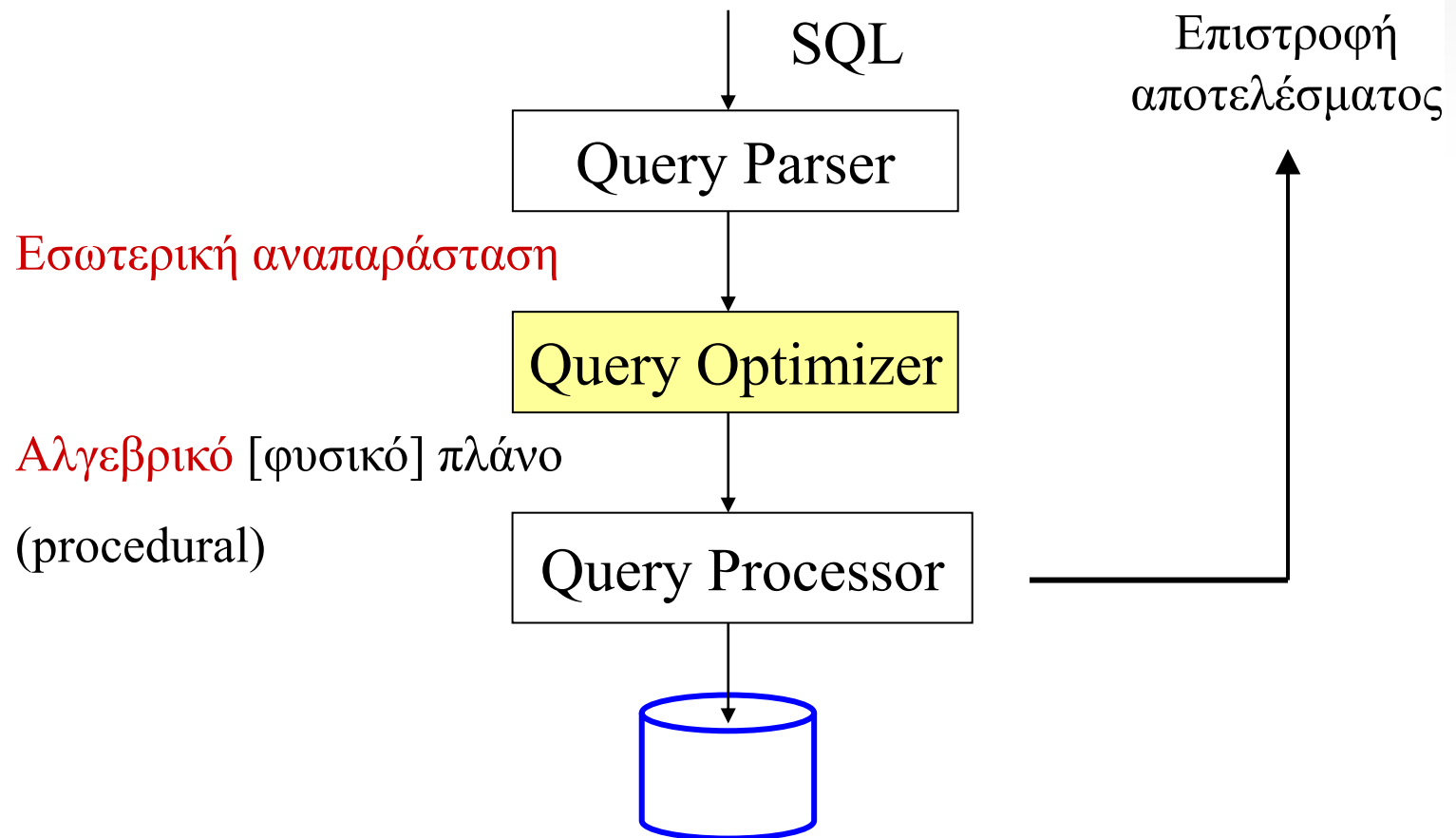
Query
blocks



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: $R \bowtie S \bowtie T$ ή $S \bowtie T \bowtie R$ ή ... ?
 - Φυσικά: nested loops, merge join, ... ?

Παράδειγμα - Σχήμα και περιορισμοί

`emp(name, age, sal, dno)`

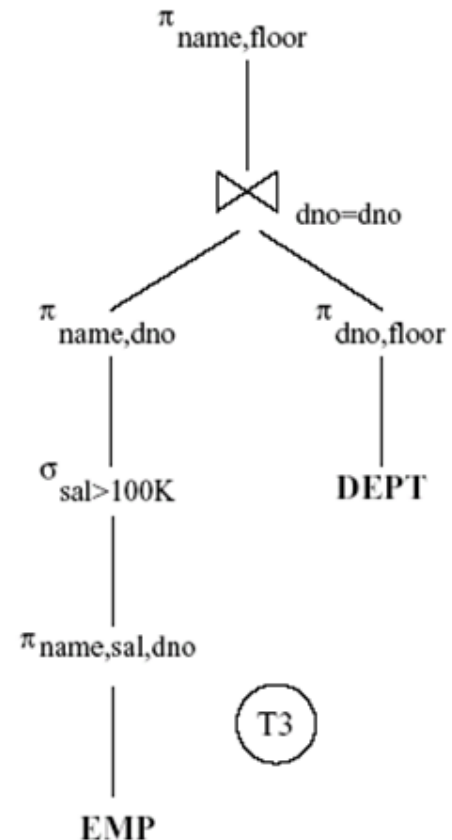
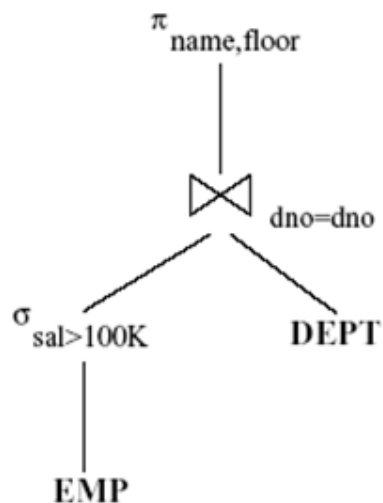
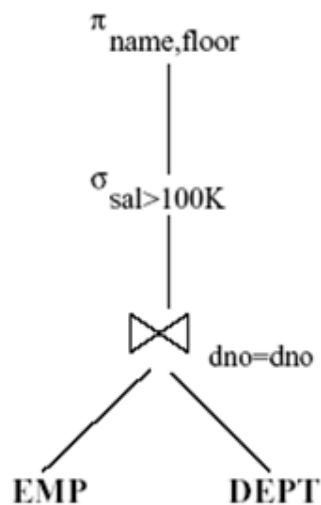
`dept(dno, dname, floor, budget, mgr, ano)`

`acnt(ano, type, balance, bno)`

`bank(bno, bname, address)`

- Κάθε υπάλληλος (`emp`) εργάζεται σε ένα τμήμα (`dept`)
- Κάθε τμήμα έχει ένα λογαριασμό (`acnt`) ο οποίος βρίσκεται σε μια τράπεζα (`bank`)

Παράδειγμα ισοδύναμων, εναλλακτικών πλάνων

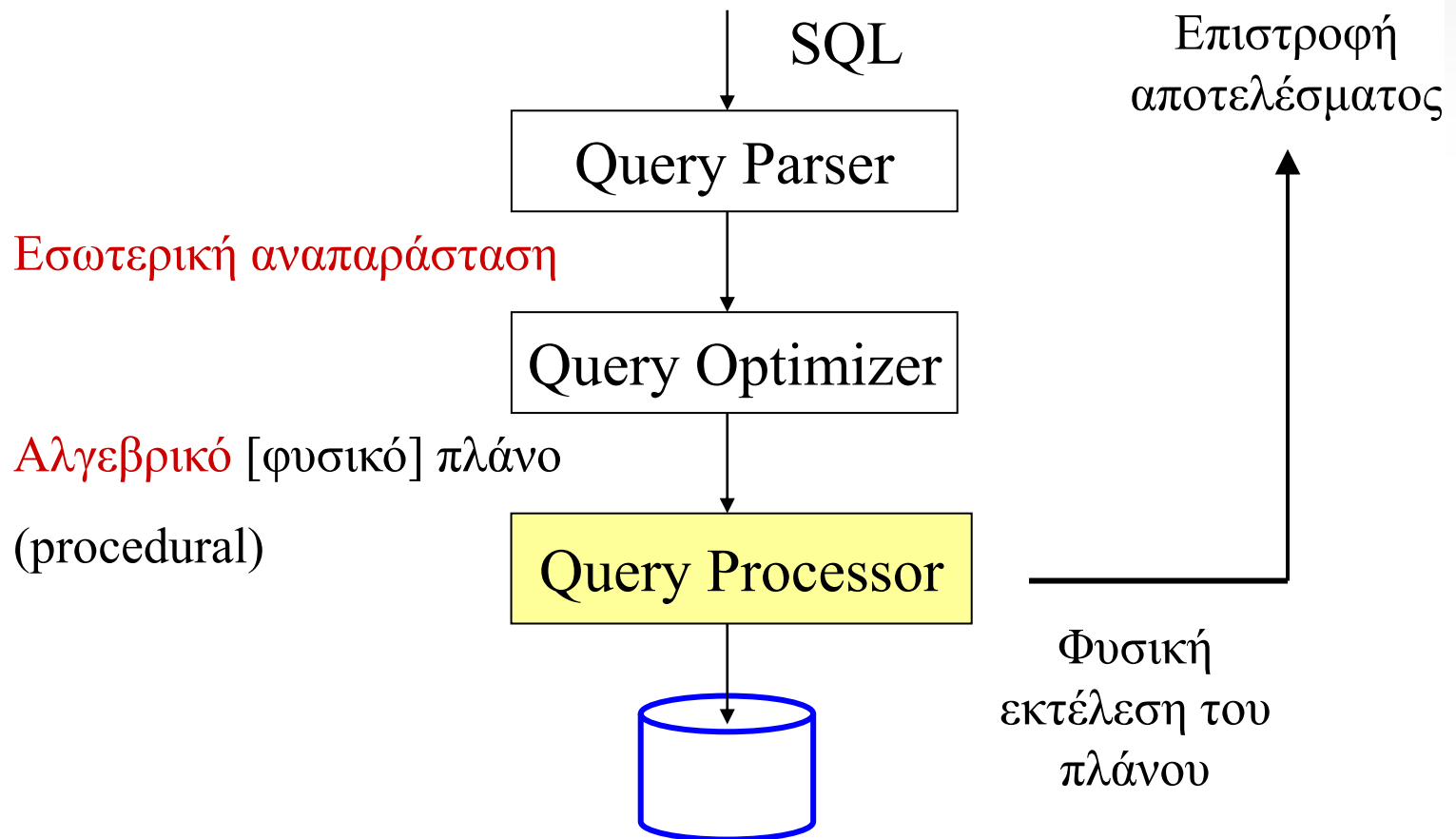


```
select name, floor
from emp, dept
where emp.dno = dept.dno
and sal > 100K
```

Επεξεργασία ερωτήσεων

- Δοθέντος του πλάνου, κάθε **λογικός** τελεστής (π.χ., σ , π , $\triangleright \triangleleft$) 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.

Επεξεργασία ερωτήσεων



Θεματολόγιο

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

Notation

- $N, M, \dots$: αριθμός **σελίδων** μιας σχέσης
- $n, m, \dots$: αριθμός **πλειάδων** μιας σχέσης
- $p_R, p_S, \dots$: αριθμός **πλειάδων ανά σελίδα**

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 $\sigma_{A=k}(\mathbf{R})$ ή εύρους $\sigma_{A \leq k}(\mathbf{R})$
 - ✦ 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 σελίδων της σχέσης
- **Κόστος**: N

➤ *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:** $\log(N)$
- Αν όχι, μετά την πρώτη, πρέπει να ανακτήσουμε και τις υπόλοιπες εγγραφές, οι οποίες είναι σειριακά αποθηκευμένες μετά την πρώτη
 - ✦ **Κόστος:** $\log(N) + \lceil \text{sel}(\varphi) * N \rceil - 1$

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.
- Το ύψος του δέντρου είναι συνήθως 3 – 4.
- Αν το πεδίο **A** είναι πρωτεύον κλειδί
 - **Κόστος**: $\text{height}(I) + 1$
- Αν όχι,
 - **Κόστος**: $\text{height}(I) + \text{cost of retrieving R-tuples}$
- Το κόστος της ανάκτησης των πλειάδων της R που πληρούν την συνθήκη εξαρτάται από το αν είναι ταξινομημένες ή όχι από $\lceil \text{sel}(\varphi) * N \rceil$ ως $\lceil \text{sel}(\varphi) * n \rceil$

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)

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

- Ενδιαφέρον παρουσιάζει η ορθή πρόβλεψη/εκτίμηση του μεγέθους του αποτελέσματος.

- ✦ Default: $|\text{result}| = n / 2$

- Αν ξέρουμε $[\min, \max]$ του εύρους τιμών του **A**:

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

- Πλέον, διαθέτουμε **ιστογράμματα** που μας επιτρέπουν αξιοπρεπέστερη πρόβλεψη...

Ερωτήσεις εύρους

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

- Αν δεν υπάρχει ευρετήριο, τι αλλάζει σε σχέση με τις point queries?
- Αν υπάρχει B+ δέντρο?
 - ✦ Τι αλλάζει αν η συνθήκη είναι $A \geq k$?
- Αν υπάρχει hash index?

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 $R \bowtie S$ είναι ιδιαίτερα κοινή και από τις πλέον χρονοβόρες $\Rightarrow$ πρέπει να εκτελεσθεί με προσοχή!
- Το $R \times S$ είναι πολύ μεγάλο $\Rightarrow$ ΔΕΝ μπορούμε να εκτελέσουμε τη σύνδεση κάνοντας πρώτα $R \times S$ και μετά μια επιλογή (σ).

Συνδέσεις Ισότητας πάνω σε Ένα πεδίο

- ✦ Έστω: M σελίδες στην R , p_R εγγραφές ανά σελίδα, N 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 (Naïve Nested Loops Join)

```
foreach tuple r in R do  
    foreach tuple s in S do  
        if  $r_i == s_j$  then add  $\langle r, s \rangle$  to result
```

- Η πλέον απλοϊκή σύνδεση εμφωλευμένων βρόγχων (με εγγραφές): Για κάθε εγγραφή στην εξωτερική σχέση R, εξετάζουμε (ελληνικά: scan) ολόκληρη την εσωτερική σχέση S.

✦ Κόστος: $M + p_R * M * N = 1000 + 100 * 1000 * 500$ I/Os.

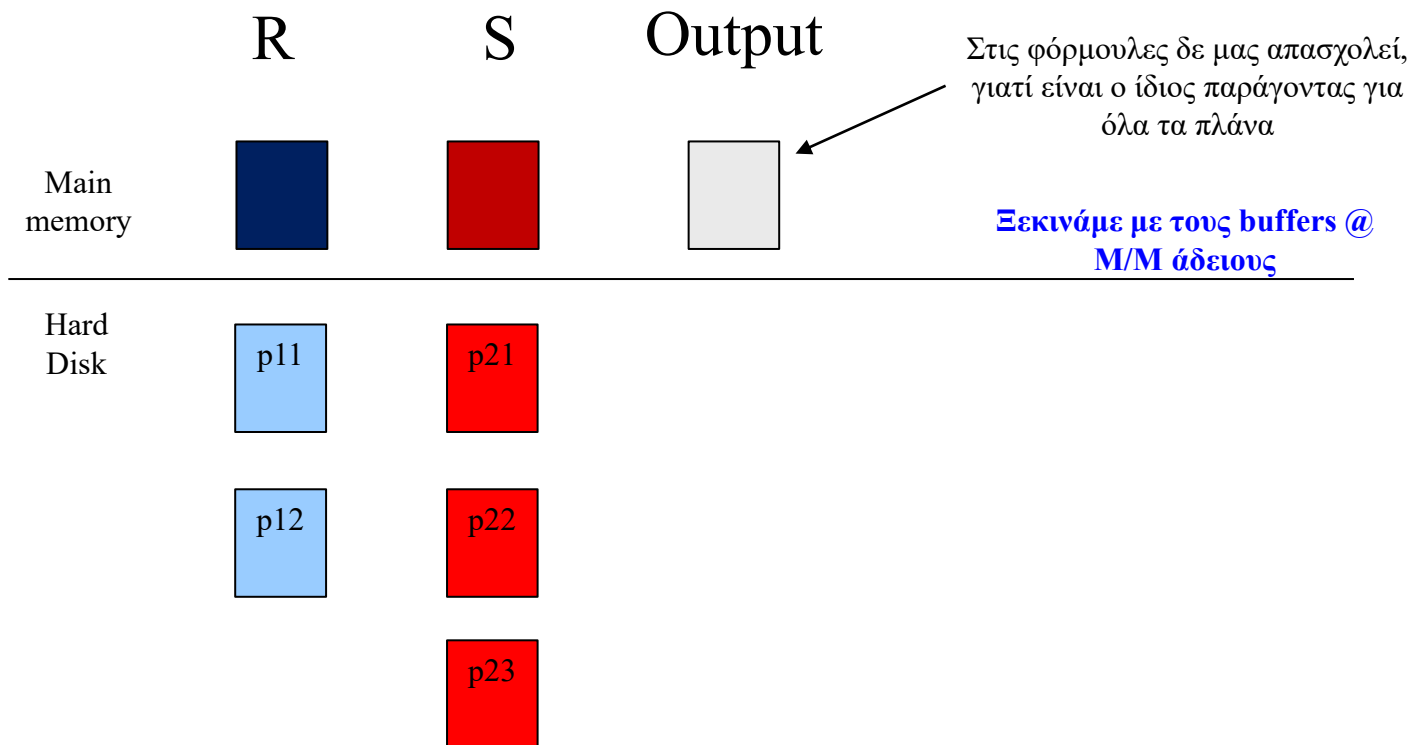
Σύνδεση Εμφωλευμένων Βρόγχων (Simple Nested Loops Join)

```
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  $\langle r, s \rangle$  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 $\langle r, s \rangle$, όπου η εγγραφή r είναι στην σελίδα της R, η εγγραφή s στην S και οι τιμές τους στα πεδία της σύνδεσης ταιριάζουν.
 - **Κόστος:** $M + M * N = 1000 + 1000 * 500$
- Αν η μικρότερη σχέση είναι εξωτερική τότε
 - **Κόστος:** $N + M * N = 500 + 1000 * 500$

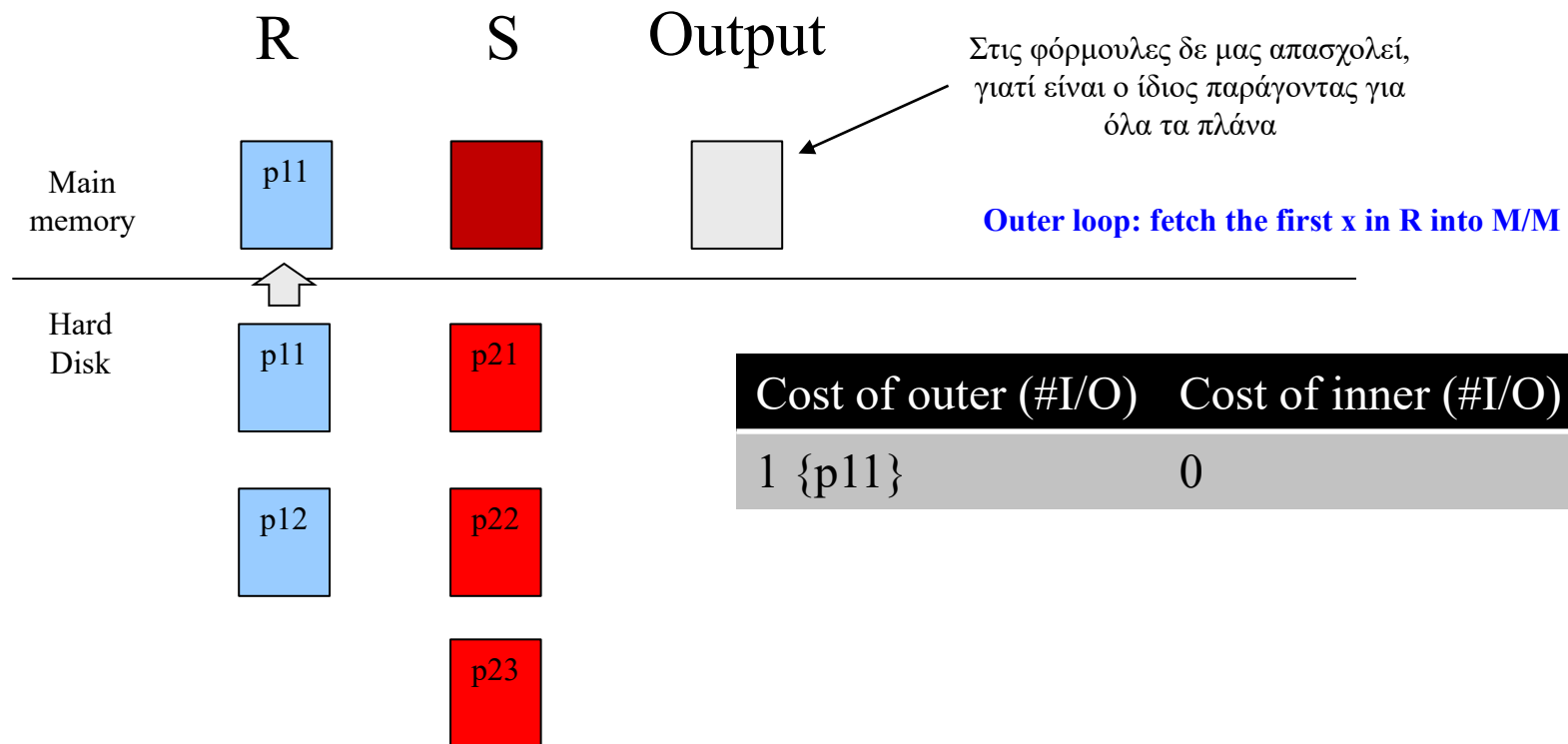
Simple Nested Loops

```
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  $\langle r, s \rangle$  to result
```



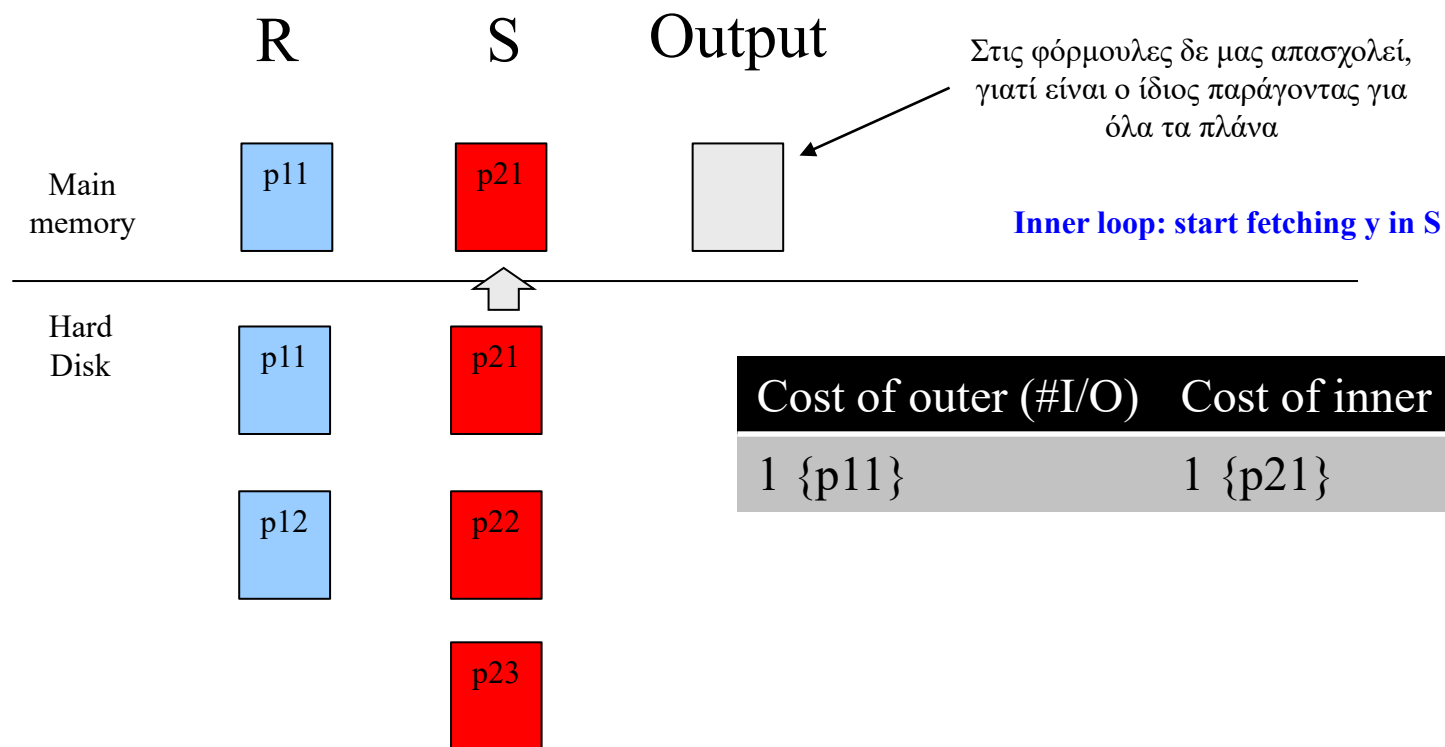
Simple Nested Loops

➔ **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 $\langle r, s \rangle$ to result



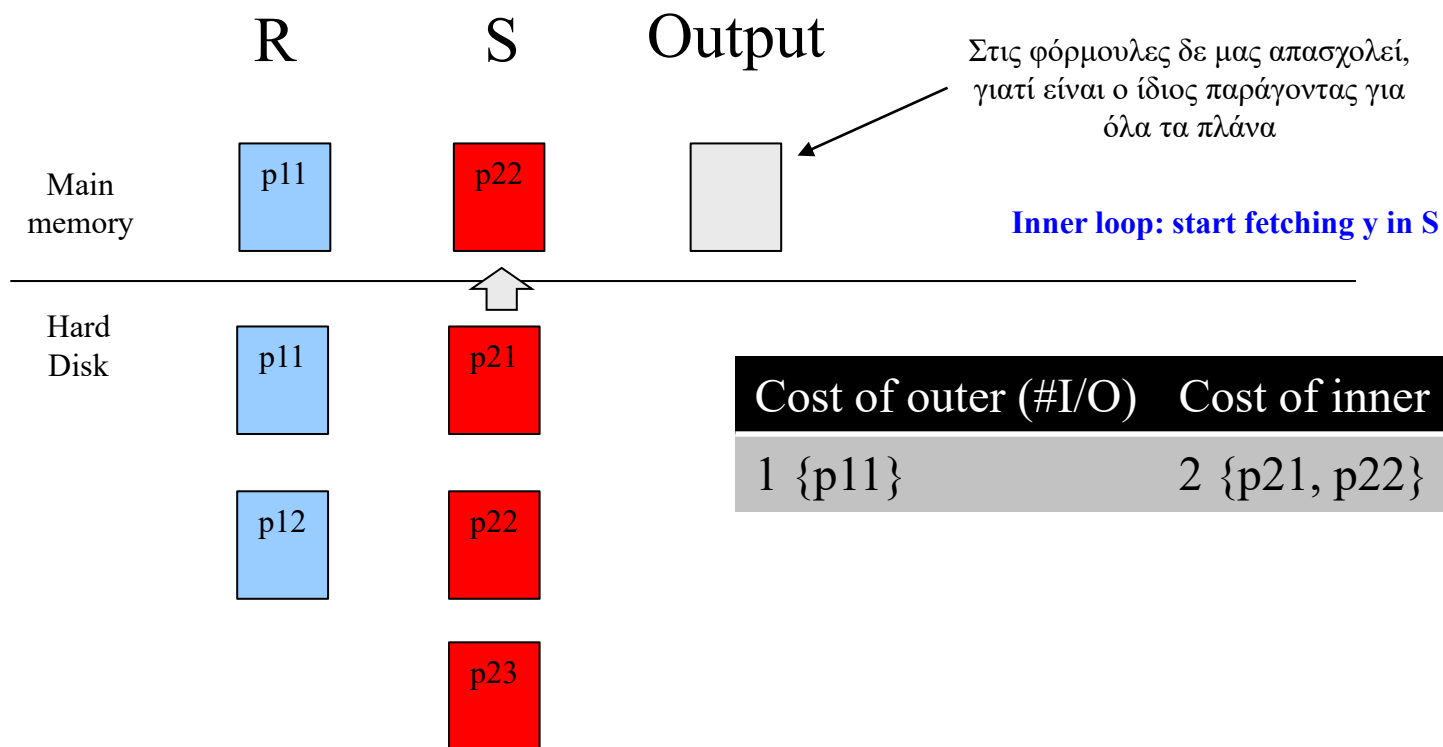
Simple Nested Loops

```
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  $\langle r, s \rangle$  to result
```



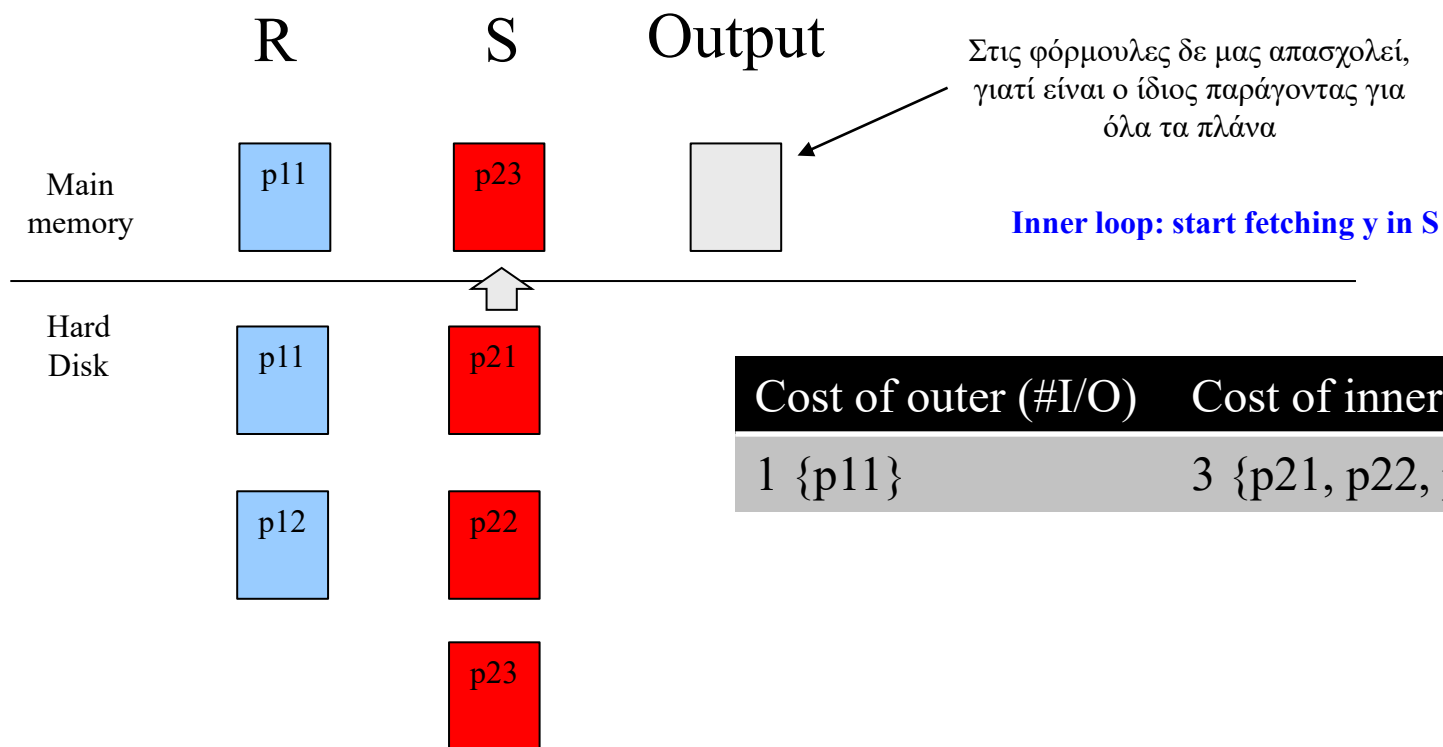
Simple Nested Loops

```
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  $\langle r, s \rangle$  to result
```



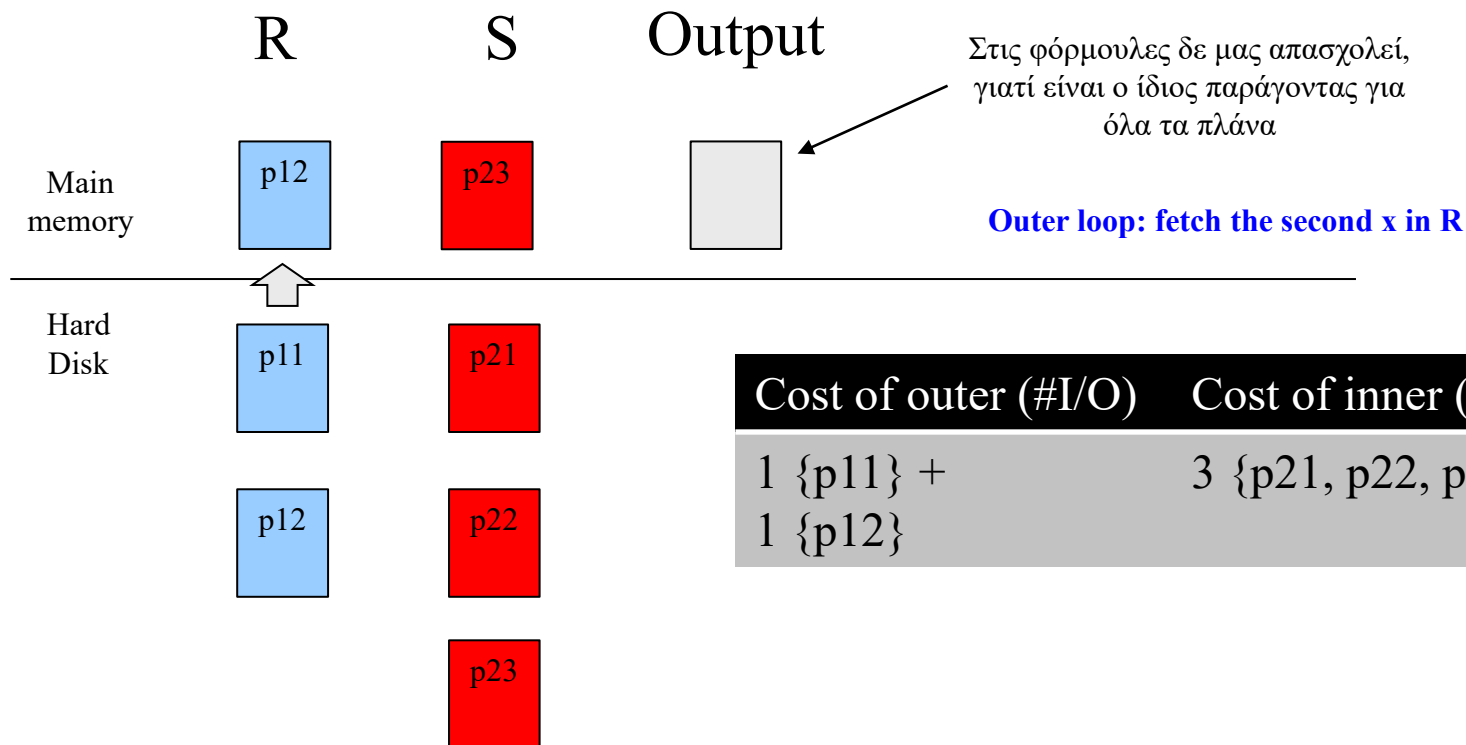
Simple Nested Loops

```
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  $\langle r, s \rangle$  to result
```



Simple Nested Loops

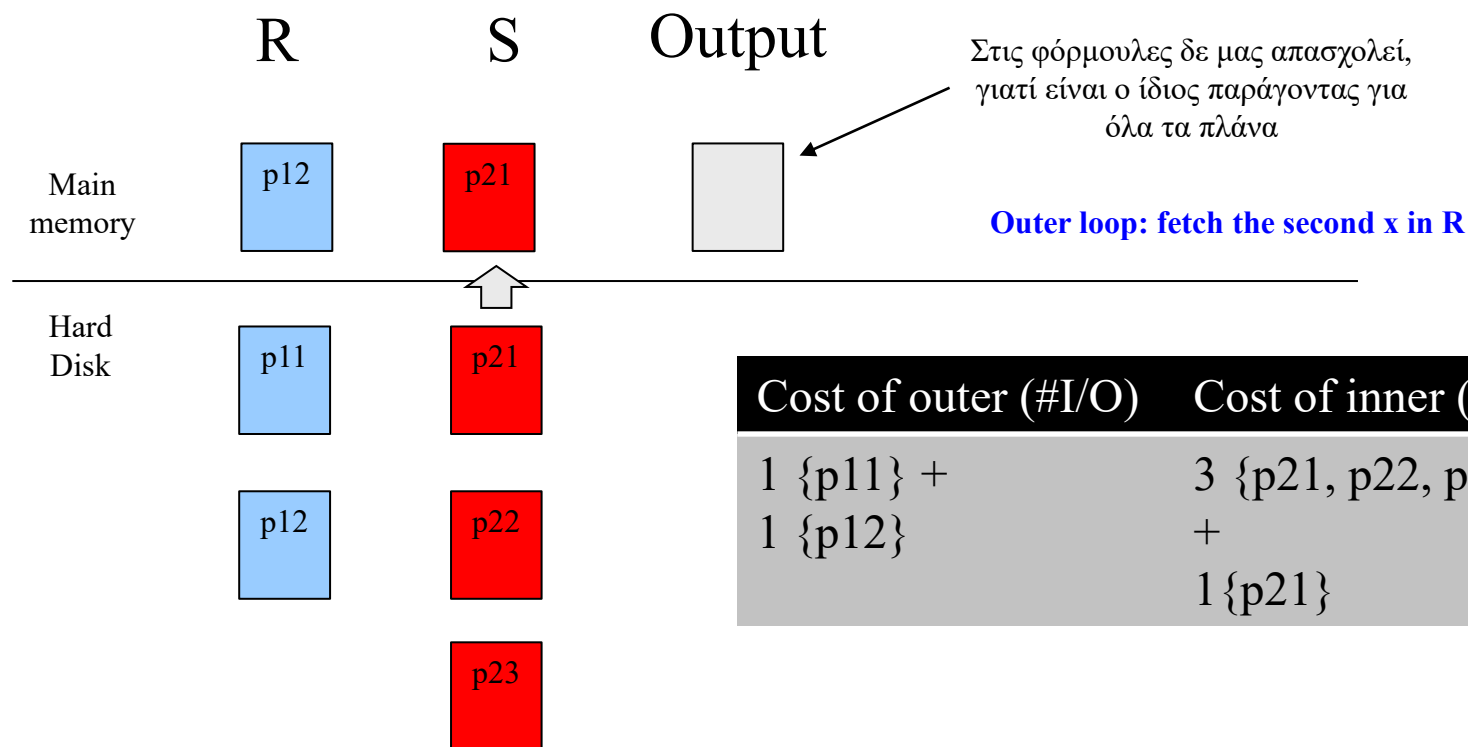
➔ **foreach** page x in R do
 foreach page y in S do
 foreach tuple r in x and s in y
 if $r_i == s_j$ **then** add $\langle r, s \rangle$ to result



Cost of outer (#I/O)	Cost of inner (#I/O)
1 {p11} + 1 {p12}	3 {p21, p22, p23} //3 for p11

Simple Nested Loops

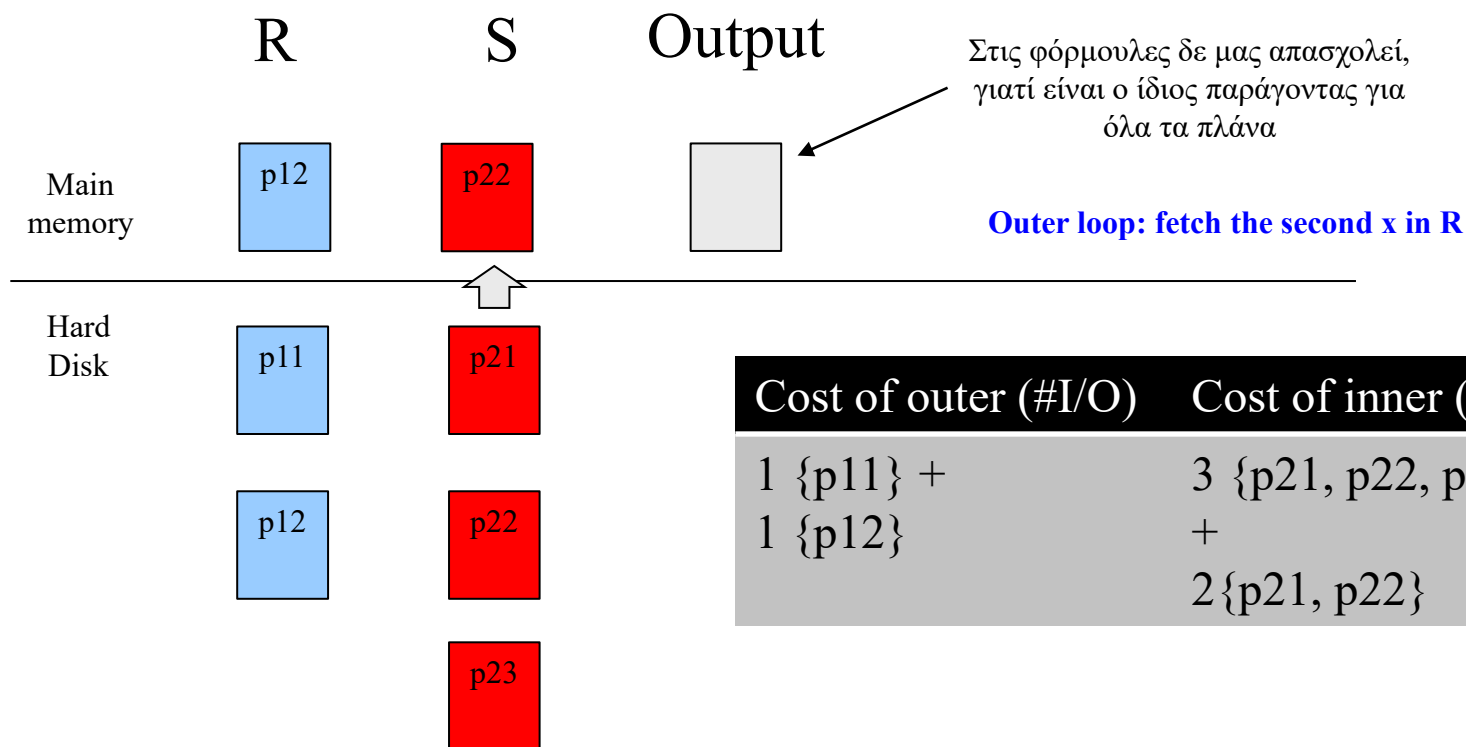
```
foreach page x in R do
  ➔ foreach page y in S do
    foreach tuple r in x and s in y
      if  $r_i == s_j$  then add  $\langle r, s \rangle$  to result
```



Cost of outer (#I/O)	Cost of inner (#I/O)
1 {p11} + 1 {p12}	3 {p21, p22, p23} //3 for p11 + 1 {p21}

Simple Nested Loops

```
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  $\langle r, s \rangle$  to result
```

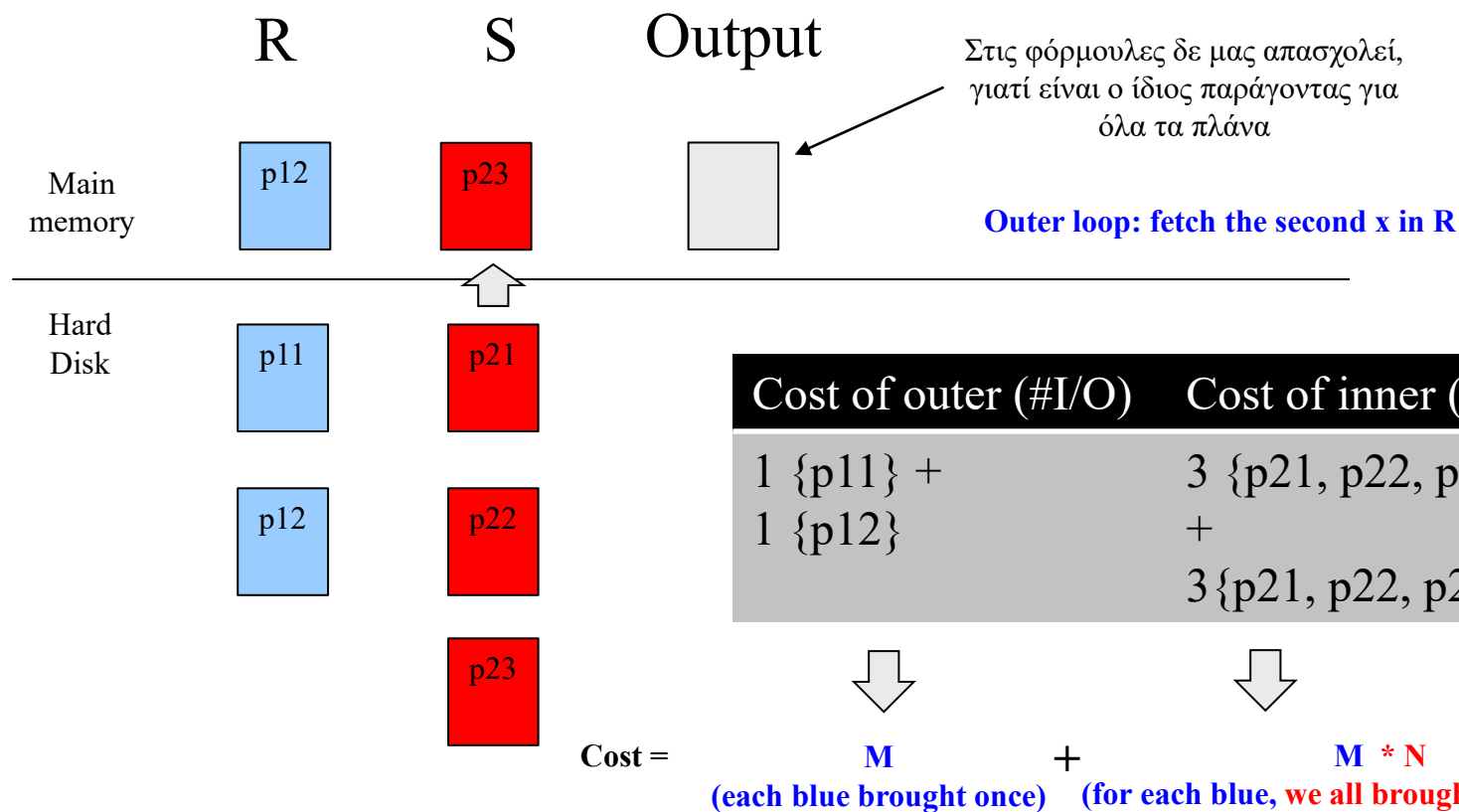


Cost of outer (#I/O)	Cost of inner (#I/O)
1 {p11} + 1 {p12}	3 {p21, p22, p23} //3 for p11 + 2 {p21, p22}

Simple Nested Loops

```

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  $\langle r, s \rangle$  to result
    
```



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  $\langle r, s \rangle$  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:** $M + (M * p_R) * \text{cost of finding matching S tuples}$

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
```

- Κόστος: $M + (M * p_R) * \text{cost of finding matching S tuples}$
- Τυπικές τιμές: για κάθε εγγραφή της R, το κόστος της προσπάθειας του ευρετηρίου της S index είναι
 - 1.2 for hash index,
 - 2-4 for B+ tree.
- Το κόστος της εύρεσης των εγγραφών της S εξαρτάται από το clustering και είναι γύρω στο 1 I/O.
 - Clustered index: 1 I/O (typical),
 - Unclustered: up to 1 I/O per matching S tuple.

ΠΡΟΣΟΧΗ!!!!

Κόστος: $M + (M * pR) * \text{cost of finding matching } S \text{ tuples}$

➤ Hash-index στο sid της Sailors (που είναι εσωτερική):

✦ Διάβασε κάθε σελίδα της Reserves:

✦ Κόστος = 1000 I/Os, #εγγραφών = $100 * 1000$ εγγραφές.

✦ Για κάθε εγγραφή της Reserves:

✦ 1.2 I/Os για να βρεις την αντίστοιχη εγγραφή στον index της Sailors,

✦ 1 I/O για να βρεις την (ακριβώς μία!) εγγραφή της Sailors που κάνει match.

✦ Σύνολο: $(1 + 1.2) * 100,000 = 220,000$ I/Os.

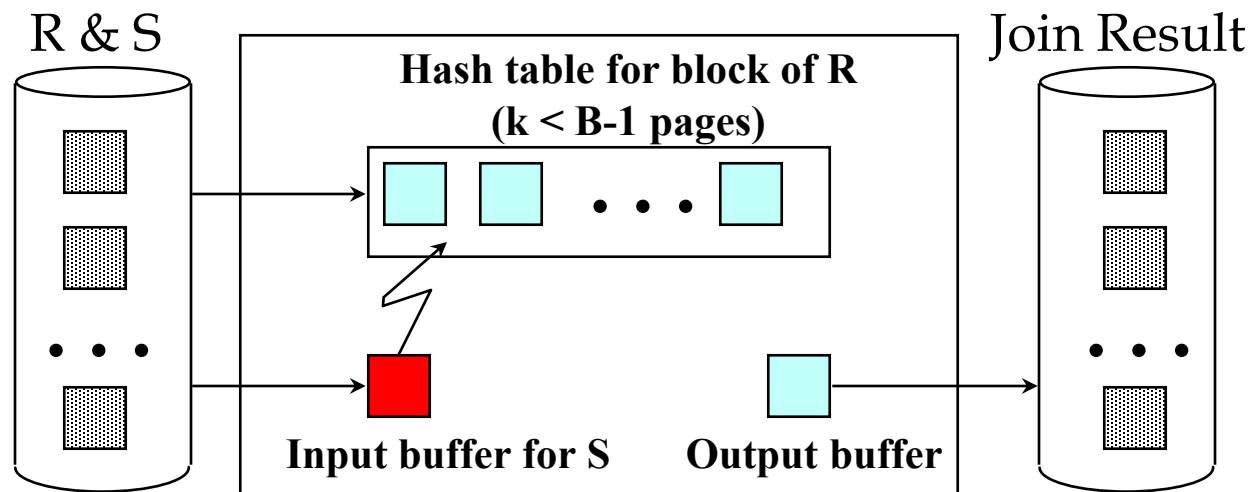
✦ Συνολικά: $1,000 + 220,000 = 221,000$ I/O's.

Κόστος: $M + (M * pR) * \text{cost of finding matching } S \text{ tuples}$

- ✦ Hash-index στο sid της Reserves (που είναι εσωτερική):
 - ✦ Διάβασε κάθε σελίδα της Sailors :
 - ✦ Κόστος = 500 I/Os, #εγγραφών = $80 * 500 = 40,000$ εγγραφές.
 - ✦ Για κάθε εγγραφή της Sailors :
 - ✦ 1.2 I/Os για να βρεις την αντίστοιχη εγγραφή στον index της Reserves.
 - ✦ Πρέπει να βρούμε τώρα τις εγγραφές της Reserves που κάνουν match. Έστω ομοιόμορφη κατανομή με 2.5 κρατήσεις ανά ναύτη ($100,000 / 40,000$). Το κόστος για να ανακτήσουμε τις κρατήσεις κάθε ναύτη είναι:
 - 1 I/O αν ο index είναι clustered
 - 2.5 I/O αν δεν είναι
 - ✦ Σύνολο: $(1.2 + X) * 40,000$, X είναι είτε 1 είτε 2.5
 - ✦ Συνολικά: όπως πριν...

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).



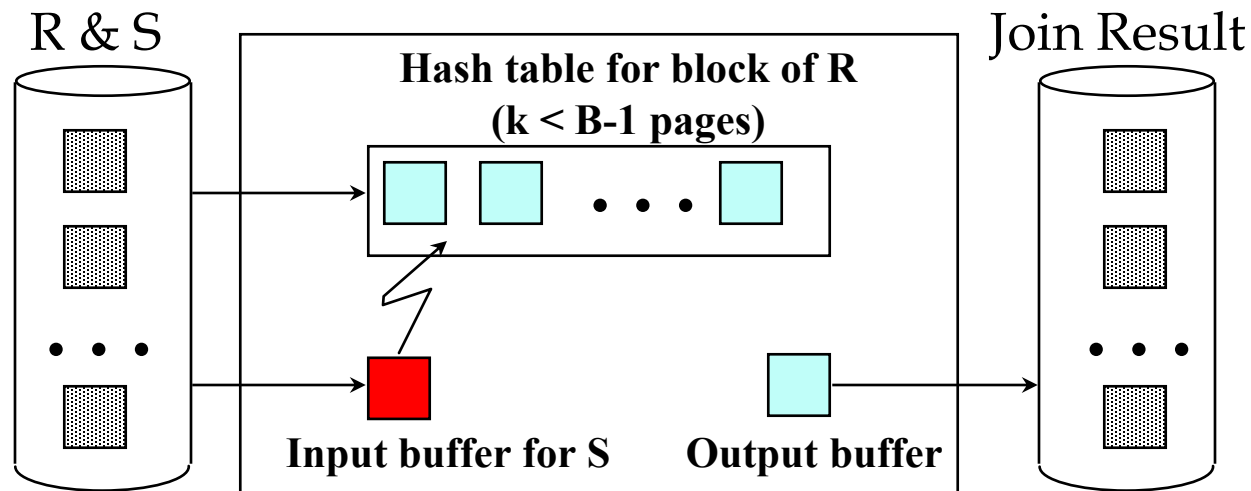
Block Nested Loops Join

L0: Διάβασε ένα block της R (R-block)

Για κάθε σελίδα της S (S-page)

Αν μια εγγραφή r στο R-block και μια εγγραφή s στην S-page κάνουν match, output $\langle r, s \rangle$

Αν η R δεν εξαντλήθηκε, Goto L0 //διάβασε το επόμενο R-block, scan S, κ.ο.κ.



Παραδείγματα Block Nested Loops

- Cost: Scan of outer + #outer blocks * scan of inner
 - #outer blocks = $\lceil \# \text{ of pages of outer} / \text{blocksize} \rceil$
- $M + M*N$ //simple NL
- $M + [M/B]*N$ //block based
- Με την Reserves (R) ως εξωτερική, και 100 σελίδες στη μνήμη ως block buffers της R:
 - Cost of scanning R = 1000 I/Os, με αποτέλεσμα 10 blocks.
 - Για κάθε block της R, scan Sailors (S): $10*500$ I/Os.
 - Σύνολο: $1000 + 5,000 = 6,000$ I/O

Παραδείγματα Block Nested Loops

✦ Cost: Scan of outer + #outer blocks * scan of inner

$$\text{✦ \#outer blocks} = \lceil \# \text{ of pages of outer} / \text{blocksize} \rceil$$

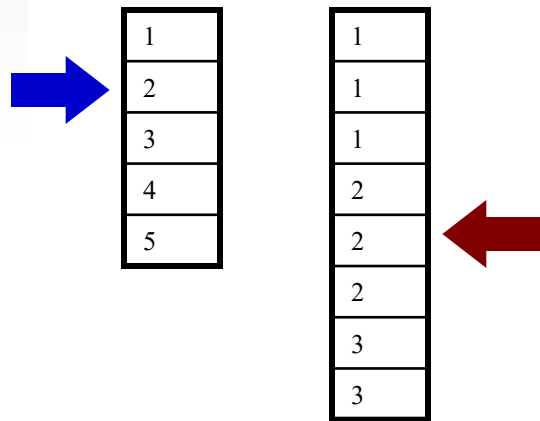
✦ Με 100-page block της Sailors ως εξωτερικής:

✦ Cost of scanning S = 500 I/Os, με αποτέλεσμα 5 blocks.

✦ Για κάθε block της S, scan Reserves: 5*1000 I/Os.

✦ Σύνολο: 500 + 5,000 = 5,500

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
```

Ταξινομήσε τα R, S με βάση το πεδίο sid

Για κάθε $r \in R$ [& day='1/1/05']

Για κάθε $s \in S$ 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)

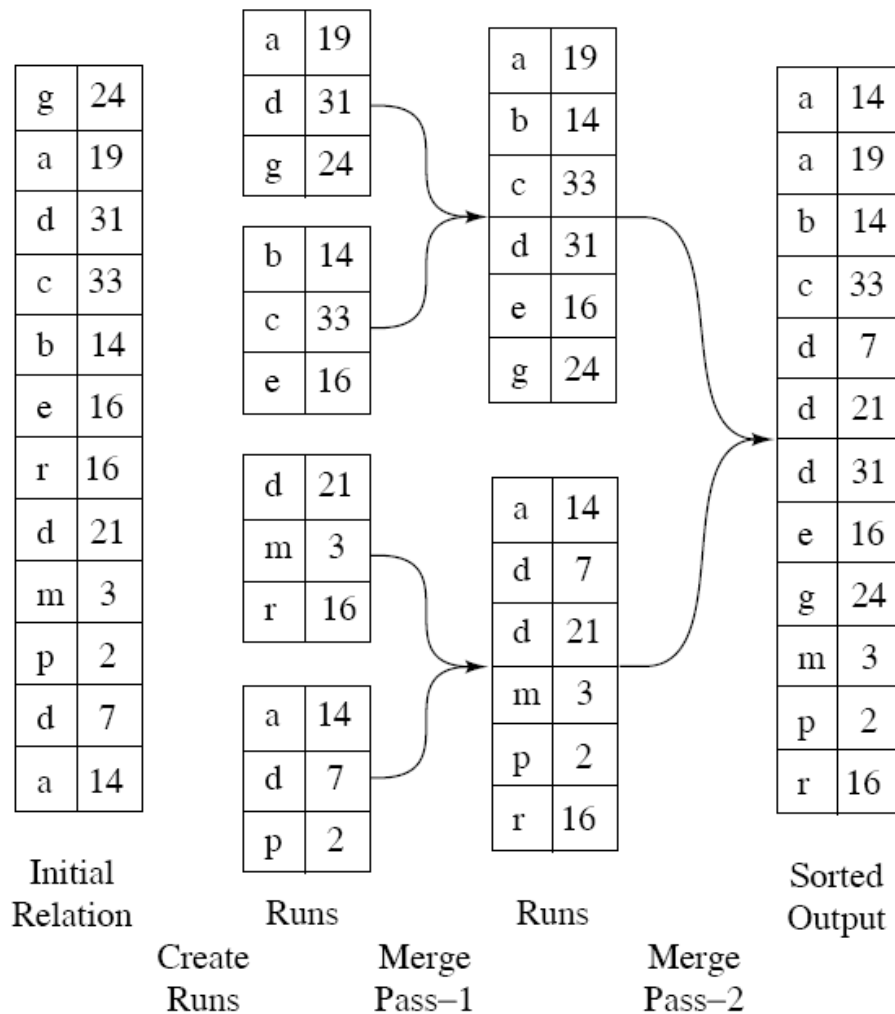
Εξωτερική ταξινόμηση

Έστω ότι έχουμε M διαθέσιμους buffers στην κύρια μνήμη

- Κατασκευάζουμε ταξινομημένες υποακολουθίες της σχέσης (sorted runs). Ο πιο απλός τρόπος είναι:
 - $i = 1$;
 - while (υπάρχουν μη επισκεφθείσες σελίδες της R) {
 - Γέμισε τους M buffers με M pages από το σκληρό δίσκο
 - Ταξινόμησε τα περιεχόμενα των M αυτών σελίδων
 - Γράψε το αποτέλεσμα της ταξινόμησης σε ένα αρχείο R_i στο σκληρό δίσκο.
 - }
- Συγχώνευσε τις ταξινομημένες υποακολουθίες.
 - if ($i < M$) {
 - Αφιέρωσε από ένα buffer για κάθε run
 - Όσοι να επισκεφθείς και την τελευταία σελίδα από όλα τα runs {
 - Διάβασε την εγγραφή με την επόμενη τιμή σε σχέση με την ταξινόμηση (π.χ., την πιο μικρή) από αυτές που είναι στην κύρια μνήμη και γράψτην στο τελικό ολικά ταξινομημένο αρχείο.
 - Αν είναι η τελευταία εγγραφή του buffer, φέρε την επόμενη σελίδα από αντίστοιχο run (αν υπάρχει)
 - }
 - }

Εξωτερική ταξινόμηση

από
Siberschatz,
Korth &
Sudarsan



Εξωτερική ταξινόμηση

- ✦ Αν έχω πιο πολλά runs από M , επαναληπτικά συγχώνευσε $M-1$ runs σε ένα ενδιάμεσο run ώσπου να μείνουν λιγότερα από M runs
- ✦ Ερωτήσεις κρίσεως:
 - ✦ Μπορούμε να κάνουμε κάτι καλύτερο από τον απλό αλγόριθμο?
 - ✦ Παίζει ρόλο ποια runs θα συγχωνεύσω?

Εξωτερική ταξινόμηση

- Ανάλυση κόστους, για μια σχέση με N σελίδες, έχοντας M buffers:
- Αρχική κατασκευή runs: $2N$
- Συνολικός αριθμός συγχωνεύσεων: $\lceil \log_{M-1}[N/M] \rceil$
 - ✦ Για κάθε σελίδα στη συγχώνευση θα έχουμε ένα διάβασμα και ένα γράψιμο άρα για κάθε συγχώνευση θα έχουμε κόστος $2N$
- Τελικά: $2N + 2N * \lceil \log_{M-1}[N/M] \rceil$

Sort-Merge Join ($R \bowtie_{i=j} S$)

- **Ταξινόμηση (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 $\geq$ τρέχουσας εγγραφής της S.
 - Μετά, διάβασε την S μέχρι η τρέχουσα εγγραφή της S $\geq$ 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 R_i (current R group) και εγγραφές της S με την ίδια τιμή στο πεδίο S_j (current S group). Προφανώς αυτές οι εγγραφές κάνουν match; output $\langle r, s \rangle$ για όλους τους συνδυασμούς αυτών των εγγραφών.
 - Συνέχισε το παραπάνω μέχρι να τελειώσει μία από τις R και S.

Sort-Merge Join $(R \bowtie_{i=j} S)$

- 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:** $\text{sort } R + \text{sort } S + \text{merge } R \& S$
- **Κόστος:** $\text{cost}(\text{sort } R) + \text{cost}(\text{sort } S) + (M+N)$
 - ✦ Το κόστος του merge τυπικά το θεωρούμε $M+N$ (δηλαδή διαβάζουμε κάθε σελίδα μία φορά), αν και στην χειρότερη περίπτωση μπορεί να φτάσει ως $M*N$ (όχι πιθανό, όμως!)

Παράδειγμα του Sort-Merge Join

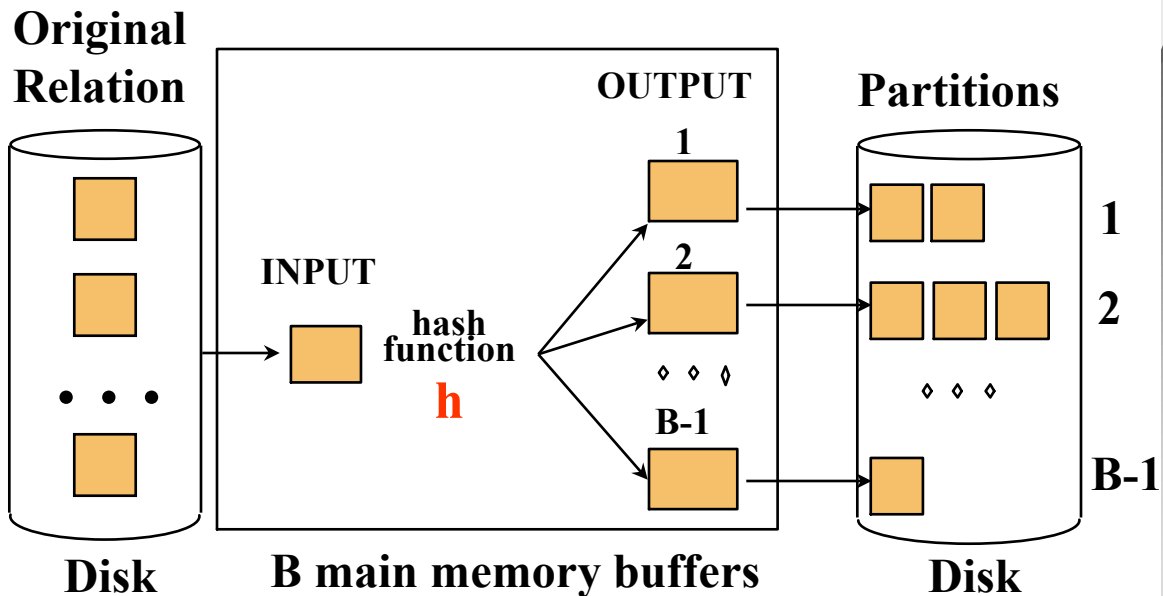
<u>sid</u>	sname	rating	age	<u>sid</u>	<u>bid</u>	<u>day</u>	rname
22	dustin	7	45.0	28	103	12/4/96	guppy
28	yuppy	9	35.0	28	103	11/3/96	yuppy
31	lubber	8	55.5	31	101	10/10/96	dustin
44	guppy	5	35.0	31	102	10/12/96	lubber
58	rusty	10	35.0	31	101	10/11/96	lubber
				58	103	11/12/96	dustin

- Με 35, 100 ή 300 buffers στη μνήμη, και οι Reserves και οι Sailors μπορούν να ταξινομηθούν σε 2 περάσματα, οπότε το συνολικό κόστος: 7500 I/O.

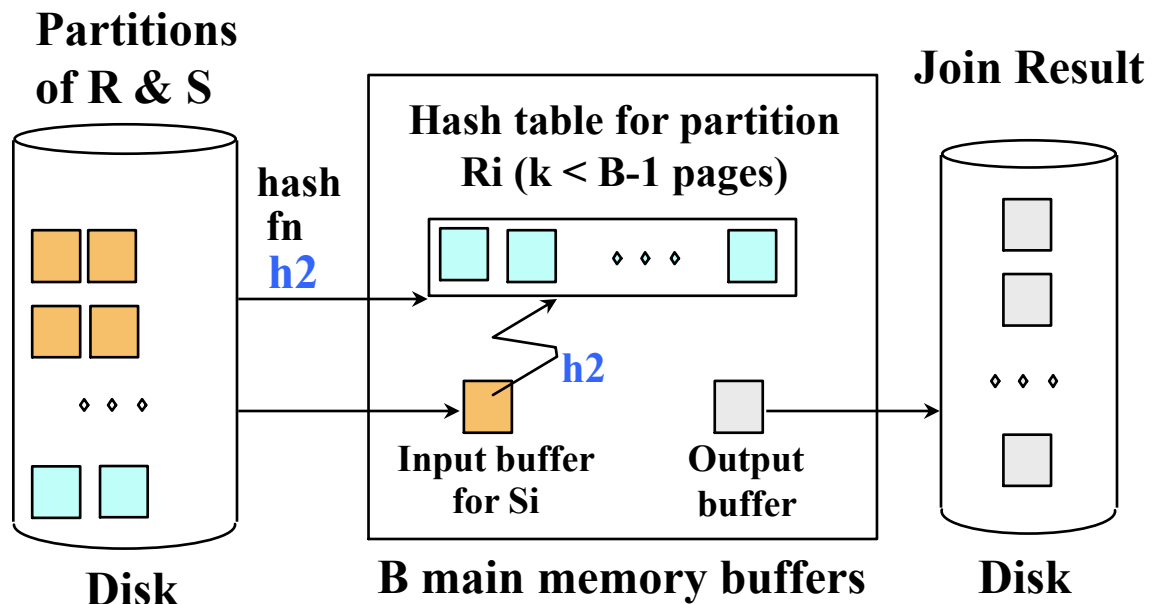
(BNL cost: 2500 to 15000 I/Os)

Hash-Join

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



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



Παρατηρήσεις στο Hash-Join

- Έστω B ο αριθμός των buffers στη μνήμη
- Θέλουμε $\#partitions\ k < B-1$ (γιατί?)
- Θέλουμε $B-2 >$ μέγεθος του μέγιστου partition που θα κρατηθεί στην μνήμη (γιατί?)
- Θεωρώντας ομοιόμορφη κατανομή στο μέγεθος των partitions, και μεγιστοποιώντας το k , έχουμε:
 - $k = B-1$, και $M/(B-1) < B-2$, $\Rightarrow$ πρέπει $B > \sqrt{M}$

Παρατηρήσεις στο Hash-Join

- Αν φτιάξουμε ένα in-memory hash table για να επιταχύνουμε το matching των εγγραφών, χρειαζόμαστε λίγο περισσότερη μνήμη.
- Αν η hash function δεν διαμοιράζει τις τιμές ομοιόμορφα, ένα ή παραπάνω partitions της R ενδέχεται να μη χωρά στη μνήμη. Μπορούμε να εφαρμόσουμε την μέθοδο του hash-join αναδρομικά (recursively) για το join του συγκεκριμένου R -partition με το αντίστοιχο S -partition.

Κόστος του Hash-Join

- Στη φάση του partitioning, **read+write** και τις δύο σχέσεις: **$2(M+N)$** .
- Στη φάση του matching, **read** και τις δύο σχέσεις: **$M+N$** I/Os.
- Στο παράδειγμά μας, αυτό βγάζει 4500 I/Os.

Sort-Merge Join vs. Hash Join:

- Υπάρχει κάποιο όριο στο ελάχιστο μέγεθος της διαθέσιμης μνήμης (*βρέστε το για κάθε μία τεχνική*) όπου το κόστος είναι $3(M+N)$ I/Os και για τις δύο τεχνικές.
- Το **hash join** είναι καλύτερο όταν τα μεγέθη των σχέσεων έχουν μεγάλη απόκλιση. Επίσης, το hash join είναι ιδιαίτερα παραλληλοποιήσιμο.
- Το **sort-merge** είναι λιγότερο ευαίσθητο στην ανομοιομορφία των δεδομένων. Επίσης, το αποτέλεσμα του βγαίνει ταξινομημένο (εξαιρετική ιδιότητα αν έχουμε σύνδεση παραπάνω από δύο σχέσεων – *γιατί?*).

Άλλες περιπτώσεις σύνδεσης

- ✦ Ισότητα πολλών πεδίων (π.χ., **R.sid=S.sid AND R.rname=S.sname**):
 - ✦ Για το Index NL, φτιάξε index στο **<sid, sname>** (αν η S είναι εσωτερική), ή χρησιμοποίησε υπάρχοντες indexes στα **sid** και **sname**.
 - ✦ Για τα Sort-Merge και Hash Join, sort/partition στο συνδυασμό των στηλών της σύνδεσης.

Άλλες περιπτώσεις σύνδεσης

✦ Συνθήκες Ανισότητας:

- ✦ Για το Index NL, χρειάζεται (clustered!) B+ tree index.
 - ✦ Όταν ψάχνουμε στην εσωτερική σχέση, κάνουμε αναζητήσεις εύρους με τον αριθμό των matches να είναι λογικά πολύ μεγαλύτερος απ' ότι στην ισότητα.
- ✦ Hash Join, Sort Merge Join δεν μπορούν να εφαρμοσθούν.
- ✦ Ο Block NL είναι πιθανότατα η καλύτερη μέθοδος σε αυτή την περίπτωση.

✦ Παρατήρηση: η μέθοδος εμφωλευμένων βρόγχων είναι η γενικότερη μέθοδος που μπορεί να εφαρμοσθεί σε όλες τις περιπτώσεις σύνδεσης

Θεματολόγιο

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

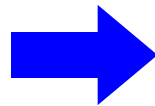
Συνάθροιση

- Η συνάθροιση είναι μια λειτουργία της οποίας η σημασιολογία υπονοεί:
 - ✦ Την κατανομή των εγγραφών της σχέσης επί της οποίας εφαρμόζεται η συνάθροιση σε ομάδες, με βάση τα πεδία ομαδοποίησης (του GROUP BY clause, δηλ.).
 - ✦ Την εφαρμογή μιας συναθροιστικής συνάρτησης (συγκεκριμένα, αυτής που βρίσκεται στο SELECT clause) επί των ομάδων αυτών
- Αν δεν υπάρχει GROUP BY clause, τότε εκτελείται μόνο το δεύτερο τμήμα

Συνάθροιση

SID Άλλα πεδία

3	
1	
2	
2	
3	
2	
3	
1	

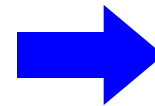


SID

1
1
2
2
2
3
3
3

} Ομάδα 1
} Ομάδα 2
} Ομάδα 3

```
SELECT SID, COUNT(*)  
FROM RESERVES  
GROUP BY SID
```



SID COUNT(*)

1	2
2	3
3	3

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

Συνάθροιση

- Αν δεν υπάρχει GROUP BY clause, τότε όλη η σχέση είναι ένα τμήμα, οπότε απλώς διαβάζουμε όλη τη σχέση
 - Αν υπάρχει ευρετήριο, μπορεί να καταφέρουμε να απαντήσουμε την ερώτηση μόνο από το ευρετήριο.
- Αν υπάρχει GROUP BY clause, τότε μπορούμε να κάνουμε ένα εκ των κάτωθι:
 - Ταξινόμηση στα πεδία του GROUP BY clause (οπότε η ομαδοποίηση των εγγραφών προκύπτει φυσιολογικά) και υπολογισμός του συναθροισμένου ποσού για κάθε ομάδα
 - Χρήση hash functions στα πεδία του GROUP BY clause, ώστε να κατασκευαστούν οι ομάδες
 - Χρήση ευρετηρίου στα πεδία της συνάθροισης (είτε B+, είτε hash) για να επιταχύνουμε την δημιουργία των ομάδων

Λειτουργίες συνόλων

- Η **τομή** και το **καρτεσιανό γινόμενο** είναι ειδικές περιπτώσεις σύνδεσης
- Η **απλή ένωση (UNION)** εκτελείται ως δύο ερωτήσεις
- Η **UNION DISTINCT** και η **αφαίρεση (EXCEPT)** είναι βαριάντες της σύνδεσης και μπορεί να εκτελεστεί είτε με sort-merge join είτε με hash-join. Η μόνη διαφορά είναι στο τι στέλνουμε στο output (**τροφή για σκέψη: πώς γίνεται –αναλυτικά– η διαδικασία?**)

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...**