ΕΠΕΞΕΡΓΑΣΙΑ ΕΠΕΡΩΤΗΣΕΩΝ

Γιάννης Κωτίδης

Επεξεργασία Επερωτήσεων

SQL Επερώτηση $\rightarrow$ Πλάνο Εκτέλεσης

Παράδειγμα

Έστω 2 σχέσεις R(A,B,C) και S(C,D,E) και η επερώτηση:

```
Select B,D
From R,S
Where R.A = “c” AND S.E = 2 AND R.C=S.C
```

R	A	B	C
	a	1	10
	b	1	20
	c	2	10
	d	2	35
	e	3	45

S	C	D	E
	10	x	2
	20	y	2
	30	z	2
	40	x	1
	50	y	3

Select B,D
From R,S
Where R.A = “c”
AND S.E = 2 AND R.C=S.C

Απάντηση:

B	D
2	x

Πως θα εκτελέσει το ΣΔΒΔ την επερώτηση?

Select B,D

From R,S

Where R.A = “c” AND S.E = 2 AND R.C=S.C

Ιδέα!

- Υπολόγισε το καρτεσιανό γινόμενο RxS
- Επέλεξε τις εγγραφές που ικανοποιούν τη λογική έκφραση
- Επέστρεψε τις τιμές των γνωρισμάτων B,D

Υπολογισμός (απλοποιημένος) Καρτεσιανού Γινομένου RxS

- Μπορώ να εκτελέσω τους παρακάτω φωλιασμένους βρόγχους (loops)

```
for each tuple r ∈ R do        ⟵ εξωτερικός βρόγχος
    for each tuple s ∈ S do    ⟵ εσωτερικός βρόγχος
        output r,s pair
```

Αλγόριθμος υπολογισμού επερώτησης

Q: Select B,D
From R,S
Where R.A = “c” AND S.E = 2 AND R.C=S.C

```
Answer(Q):
  for each tuple r ∈ R do
    for each tuple s ∈ S do
      if (r.A = “c” AND s.E = 2 AND r.C=s.C)
        output (r.B,s.D)
```

- Μπορεί να δουλέψει (με μικρές τροποποιήσεις) για άλλα ερωτήματα της μορφής SELECT-FROM-WHERE?
- Είναι αποδοτικός?

Select B,D
From R,S
Where R.A = “c”
AND S.E = 2 AND
R.C=S.C

Να μία! →

R X S	R.A	R.B	R.C	S.C	S.D	S.E
	a	1	10	10	x	2
	a	1	10	20	y	2
	.					
	.					
	c	2	10	10	x	2
	.					
	.					

<2,x>

Μπορώ να περιγράψω το συγκεκριμένο πλάνο εκτέλεσης μέσω της σχεσιακής άλγεβρας!

$$\Pi_{B,D} [\sigma_{R.A=\text{“c”} \text{ AND } S.E=2 \text{ AND } R.C = S.C} (R \times S)]$$

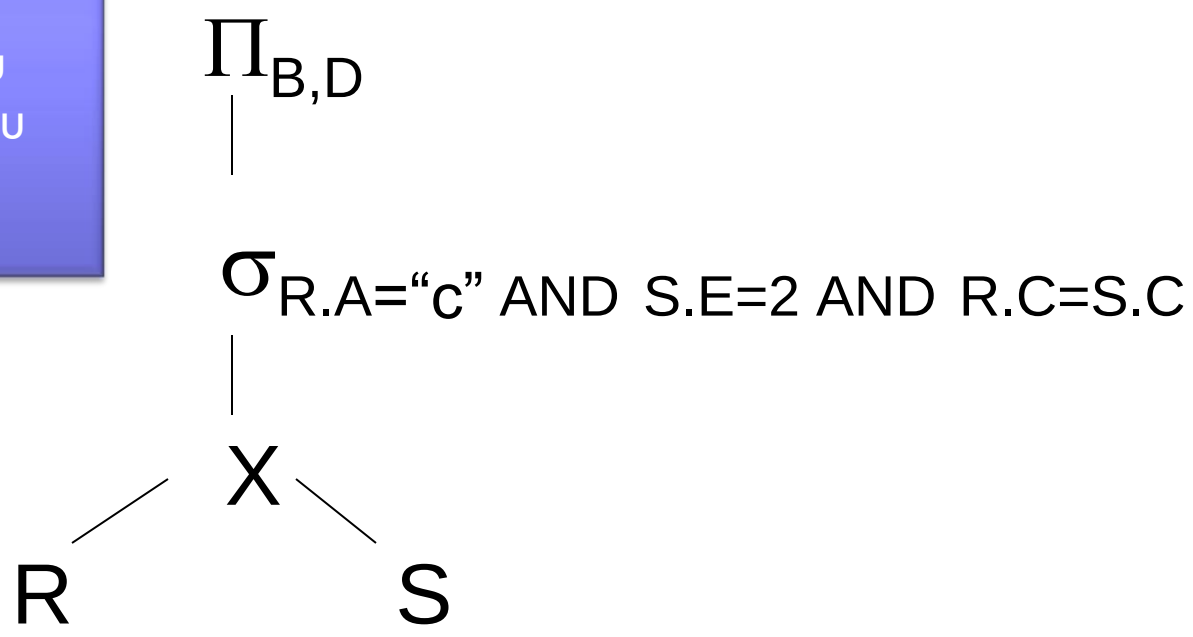
- Ονομάζεται «Λογικό Πλάνο» εκτέλεσης της επερώτησης

Ας θυμηθούμε τους βασικούς σχεσιακούς τελεστές

- Προβολή (project): π
- Επιλογή (selection): σ
- Καρτεσιανό γινόμενο (cartesian product): X
- Φυσική σύζευξη (Natural Join): $\bowtie$
- Σύζευξη θήτα (Theta Join): $\bowtie_{\theta}$

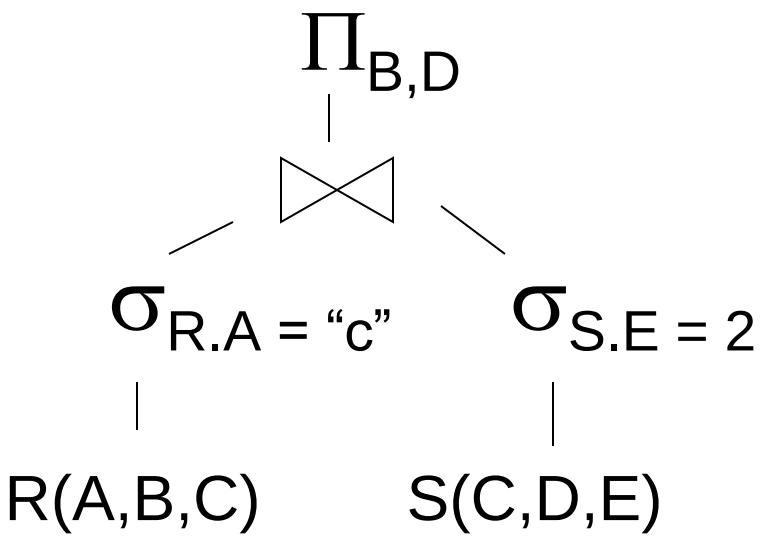
Ισοδύναμη αλγεβρική περιγραφή με τη μορφή δέντρου

- ✓ Οι αλγεβρικοί τελεστές είναι ενδιάμεσοι κόμβοι του δέντρου
- ✓ Οι σχέσεις είναι τα φύλλα του δέντρου



$$\Pi_{B,D}\,[\,\sigma_{R.A=\text{“c”}\ \text{AND}\ S.E=2\ \text{AND}\ R.C=S.C}\,(R \times S)]$$

Ένα δεύτερο λογικό πλάνο



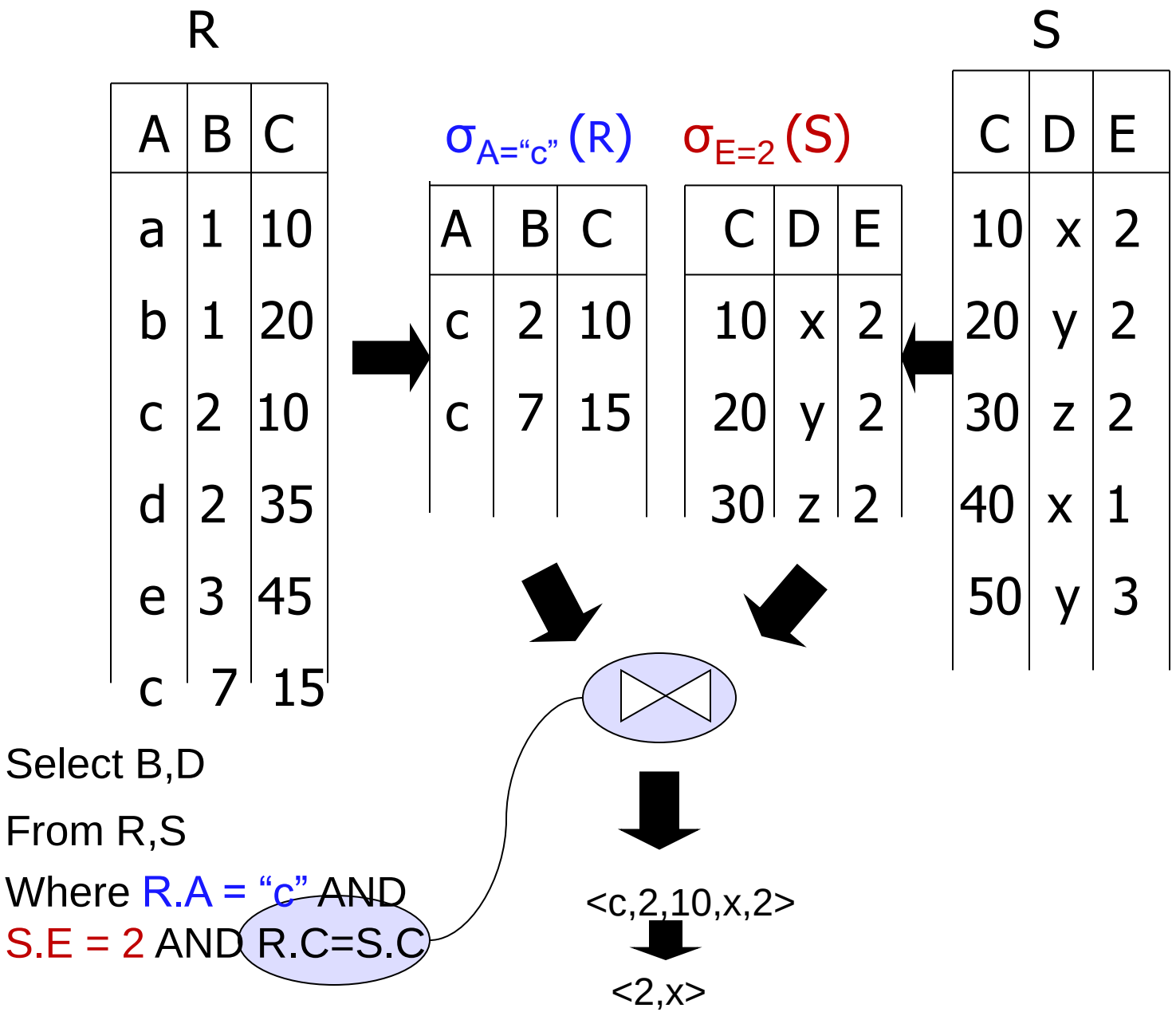
Select B,D

From R,S

Where R.A = “c”

AND S.E = 2

AND R.C=S.C


R
A B C
a 1 10
b 1 20
c 2 10
d 2 35
e 3 45
c 7 15
$\sigma_{A="c"}(R)$
A B C
c 2 10
c 7 15
$\sigma_{E=2}(S)$
C D E
10 x 2
20 y 2
30 z 2
S
C D E
10 x 2
20 y 2
30 z 2
40 x 1
50 y 3
Select B,D
From R,S
Where R.A = "c" AND
S.E = 2 AND R.C=S.C
<c,2,10,x,2>
<2,x>

3ο πλάνο!

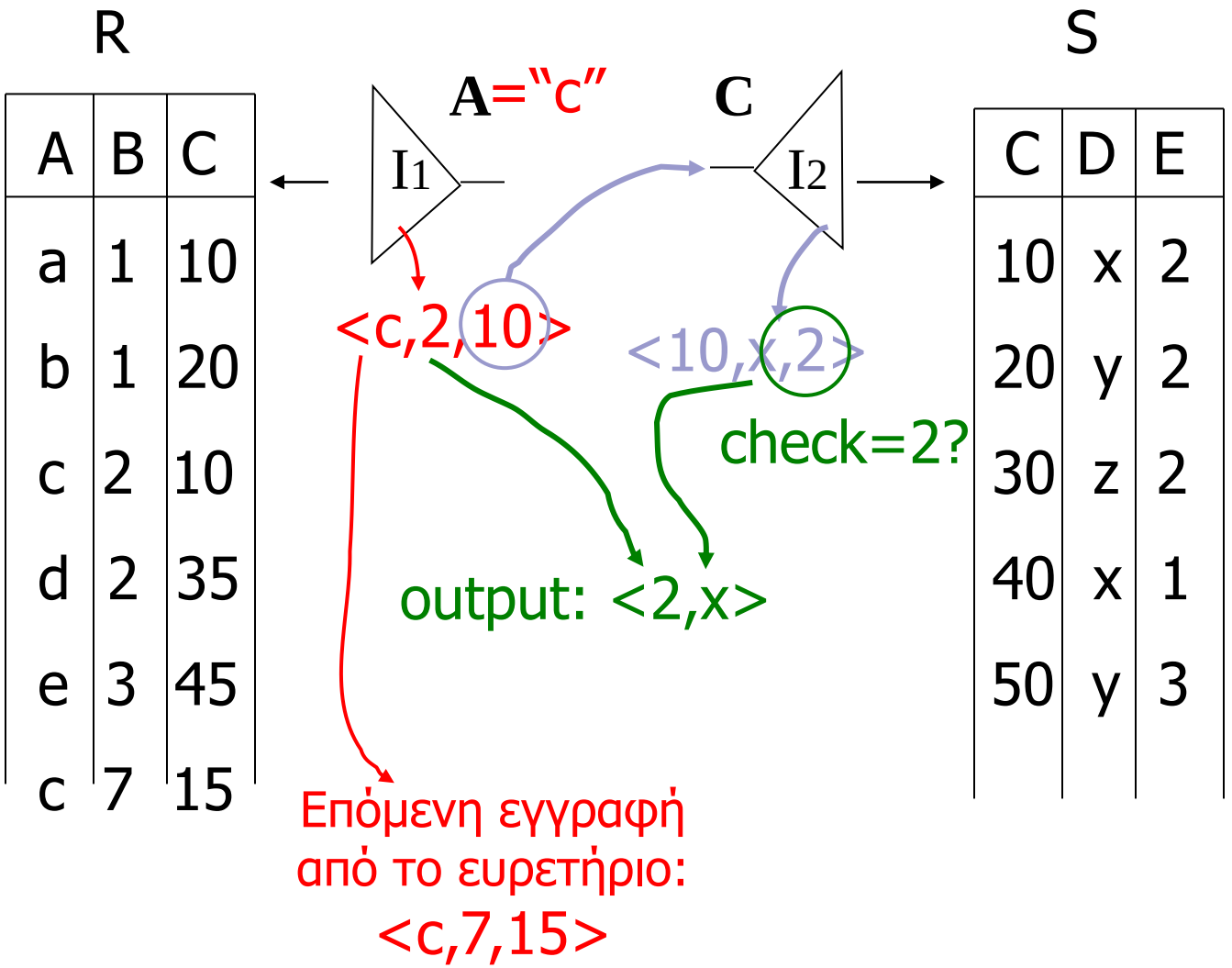
Έστω ότι έχω ευρετήρια στα γνωρίσματα R.A and S.C

(1) Χρησιμοποίησε το ευρετήριο στο R.A για να επιλέξεις εγγραφές της R με τιμή R.A = “c”

(2) Για κάθε τέτοια εγγραφή, κοίτα την τιμή R.C και χρησιμοποίησε το ευρετήριο στο S.C για να βρεις εγγραφές από τη σχέση S που κάνουν join

(3) Από τις εγγραφές που μας γυρίζει το ευρετήριο της σχέσης S, κράτα μόνο αυτές με τιμή S.E=2

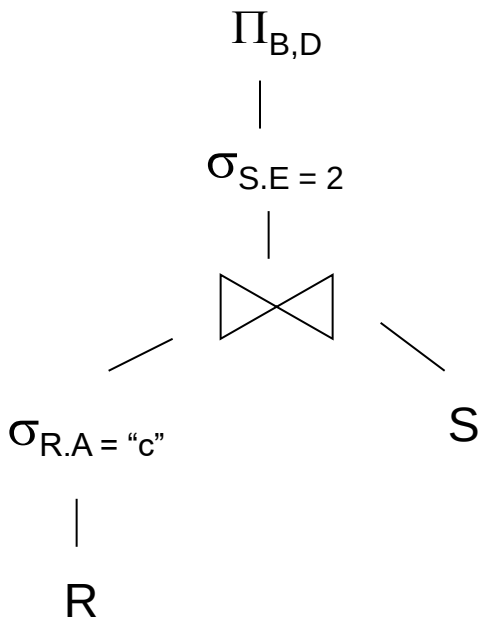
(4) Επέστρεψε στο χρήστη τις τιμές R.B,S.D από τα ζεύγη εγγραφών που ικανοποιούν την παραπάνω συνθήκη


R
A
B
C
a
1
10
b
1
20
c
2
10
d
2
35
e
3
45
c
7
15
A="c"
I1
C
I2
<c,2,10>
<10,x,2>
check=2?
output: <2,x>
Επόμενη εγγραφή
από το ευρετήριο:
<c,7,15>
S
C
D
E
10
x
2
20
y
2
30
z
2
40
x
1
50
y
3

Ας σκεφτούμε

- Αυτό που περιγράψαμε ως πλάνο #3 είναι λογικό πλάνο; (ναι/όχι)

Ποιο είναι το λογικό πλάνο που ενδέχεται να οδηγήσει στον αλγόριθμο που περιγράψαμε;



Ερώτηση

Είναι οι περιγραφές Α και Β ισοδύναμες;

A

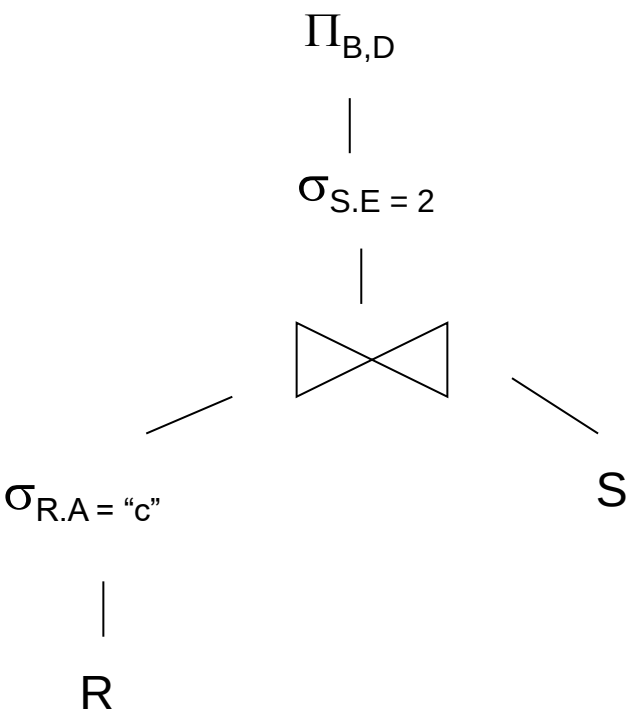
(1) Χρησιμοποίησε το ευρετήριο στο R.A για να επιλέξεις εγγραφές της R με τιμή R.A = “c”

(2) Για κάθε τέτοια εγγραφή, κοίτα την τιμή R.C και χρησιμοποίησε το ευρετήριο στο S.C για να βρεις εγγραφές από τη σχέση S που κάνουν join

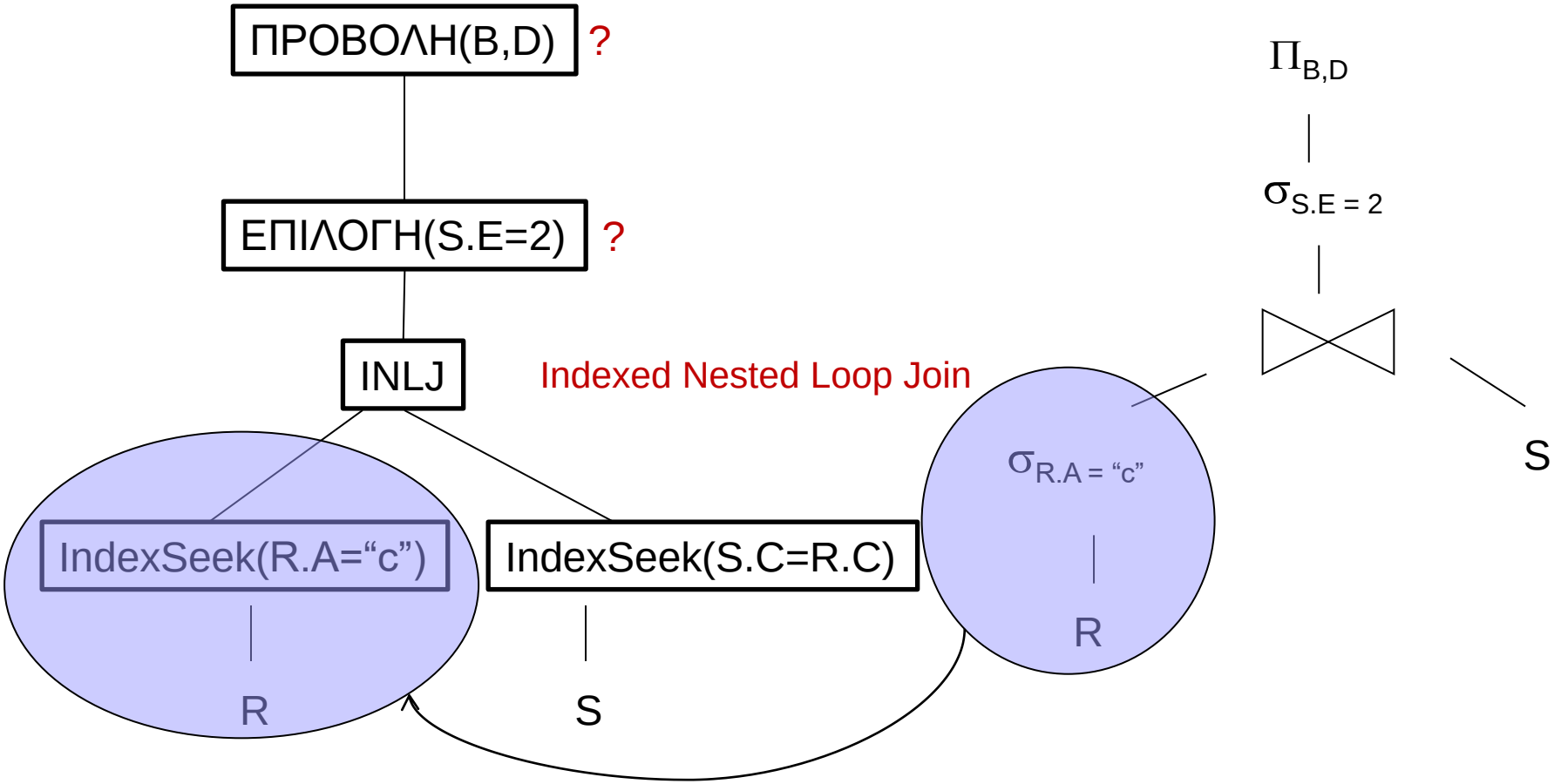
(3) Από τις εγγραφές που μας γυρίζει το ευρετήριο της σχέσης S, κράτα μόνο αυτές με τιμή S.E=2

(4) Επέστρεψε στο χρήστη τις τιμές R.B,S.D από τα ζεύγη εγγραφών που ικανοποιούν τη παραπάνω συνθήκη

B



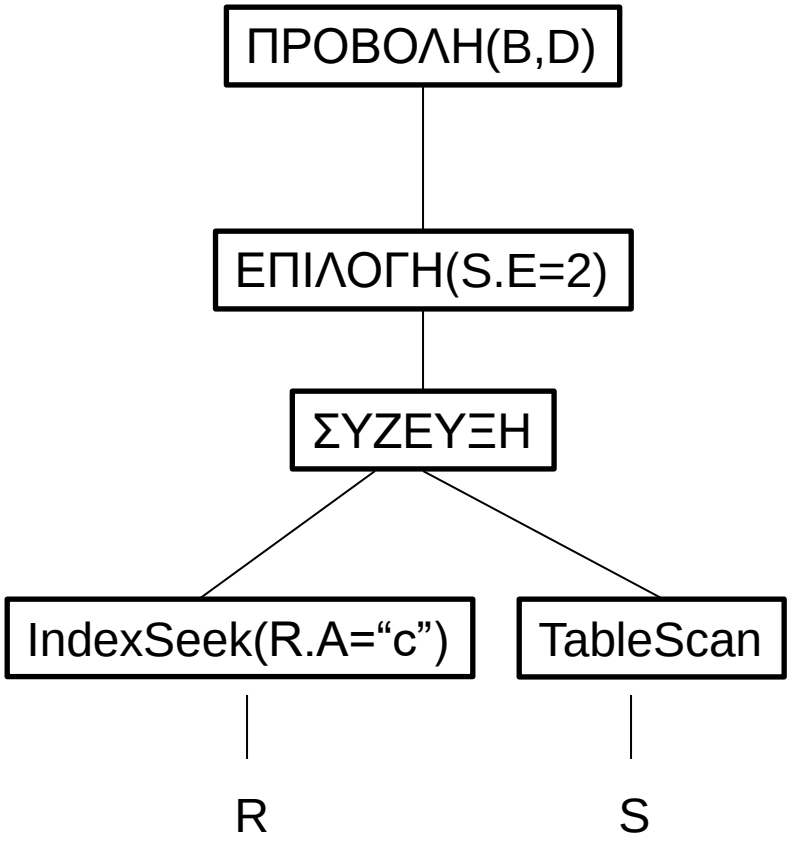
Φυσικό/Λογικό πλάνο



Ένας τρόπος υλοποίησης

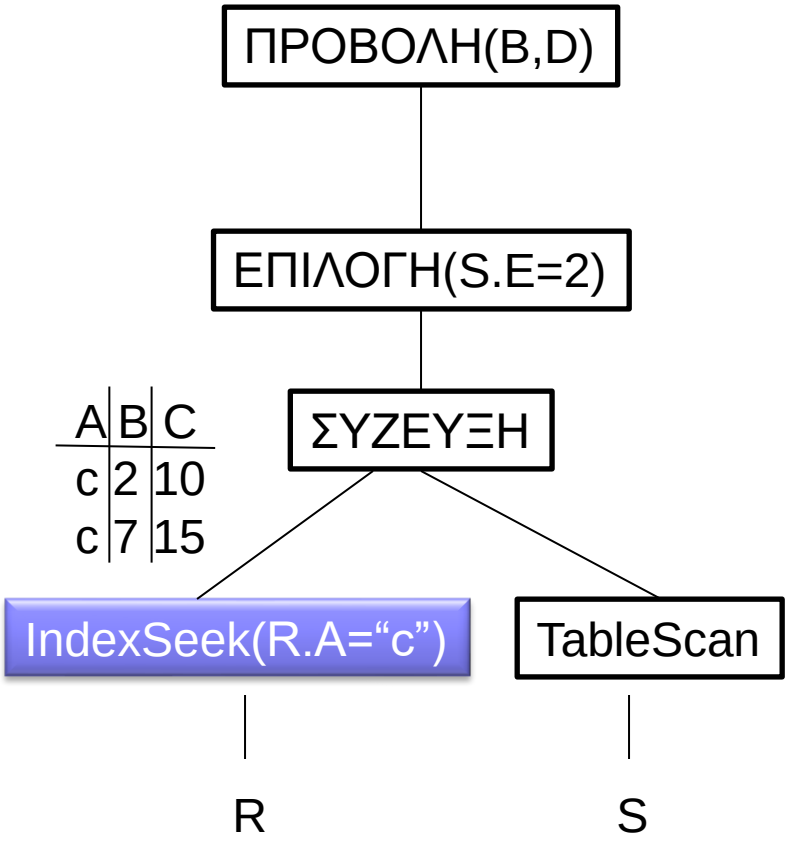
Φυσικό πλάνο: 2 μοντέλα αποτίμησης

1. Υλοποίηση (materialization)
2. Μοντέλο Επαναλήπτη (iterator model)



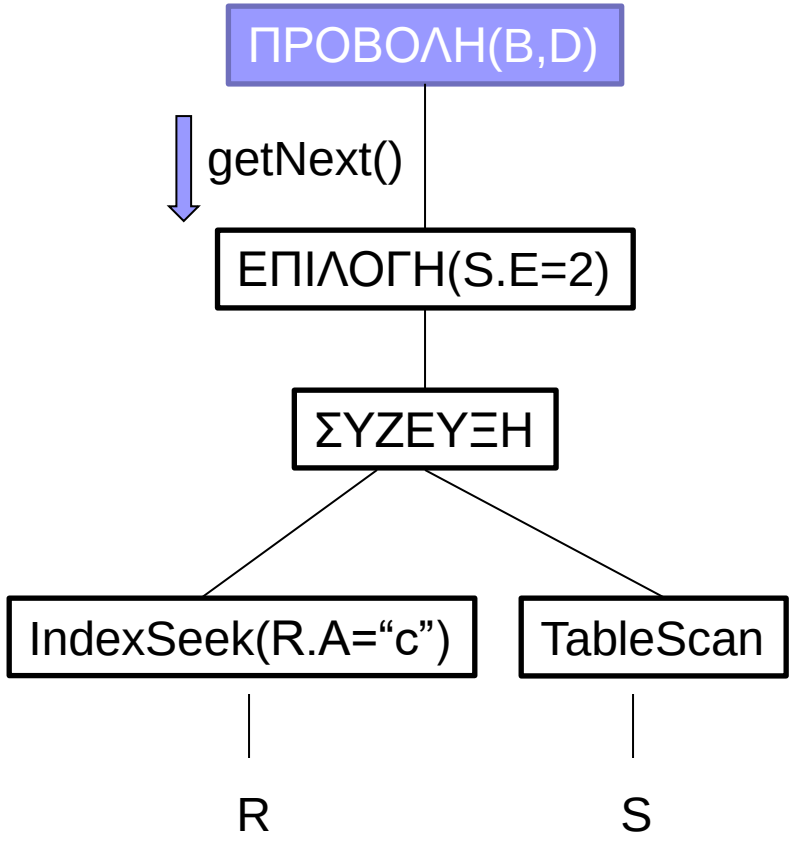
Υλοποίηση (materialization)

- Ο φυσικός τελεστής διαβάζει την είσοδο του και υπολογίζει το αποτέλεσμα (πχ σε προσωρινή σχέση στο δίσκο ή στη μνήμη) το οποίο λαμβάνει ο επόμενος (από πάνω) τελεστής (ή ο τελικός χρήστης)



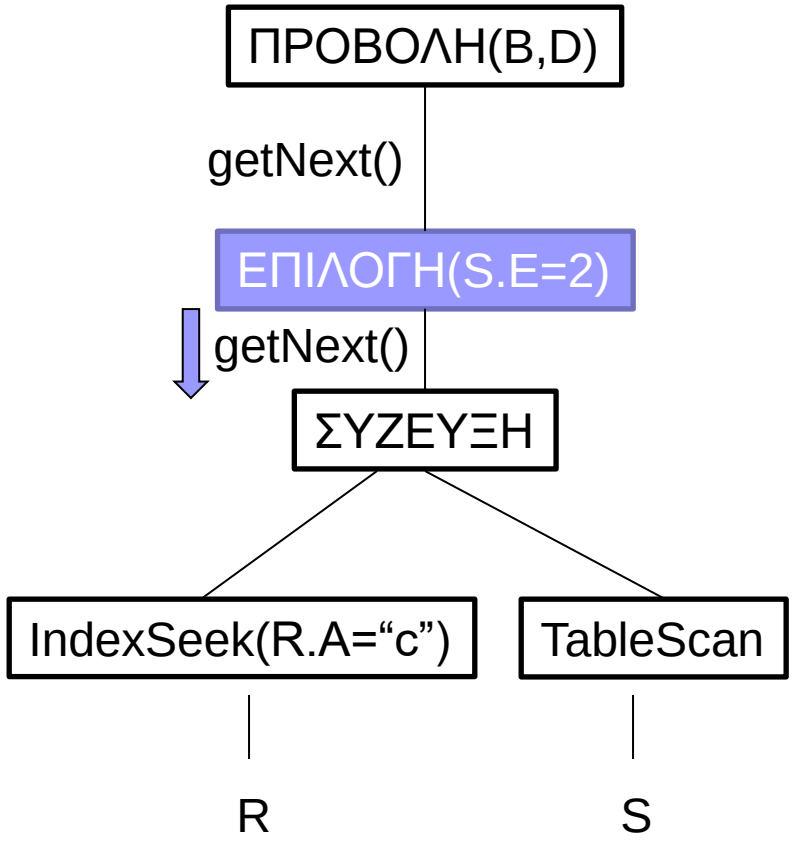
Μοντέλο Επαναλήπτη (iterator model)

- Ο κάθε τελεστής ζητάει από τον επόμενο του (παιδί του στο δέντρο) μία εγγραφή την οποία επεξεργάζεται και παραδίνει στον πατέρα του



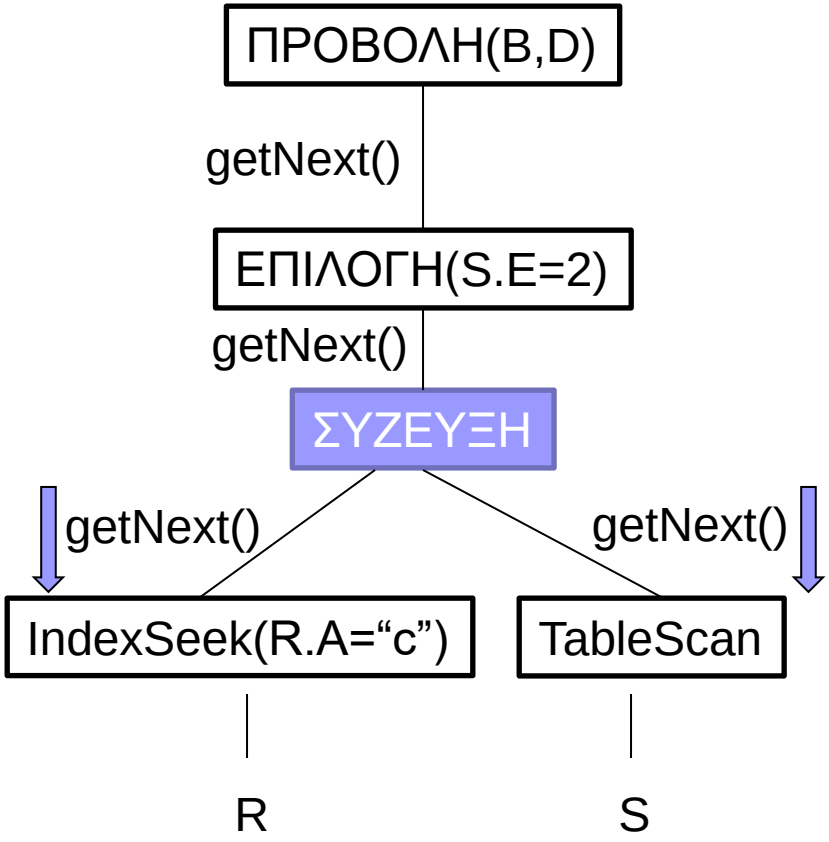
Μοντέλο Επαναλήπτη (iterator model)

- Ο κάθε τελεστής ζητάει από τον επόμενο του (παιδί του στο δέντρο) μία εγγραφή την οποία επεξεργάζεται και παραδίνει στον πατέρα του



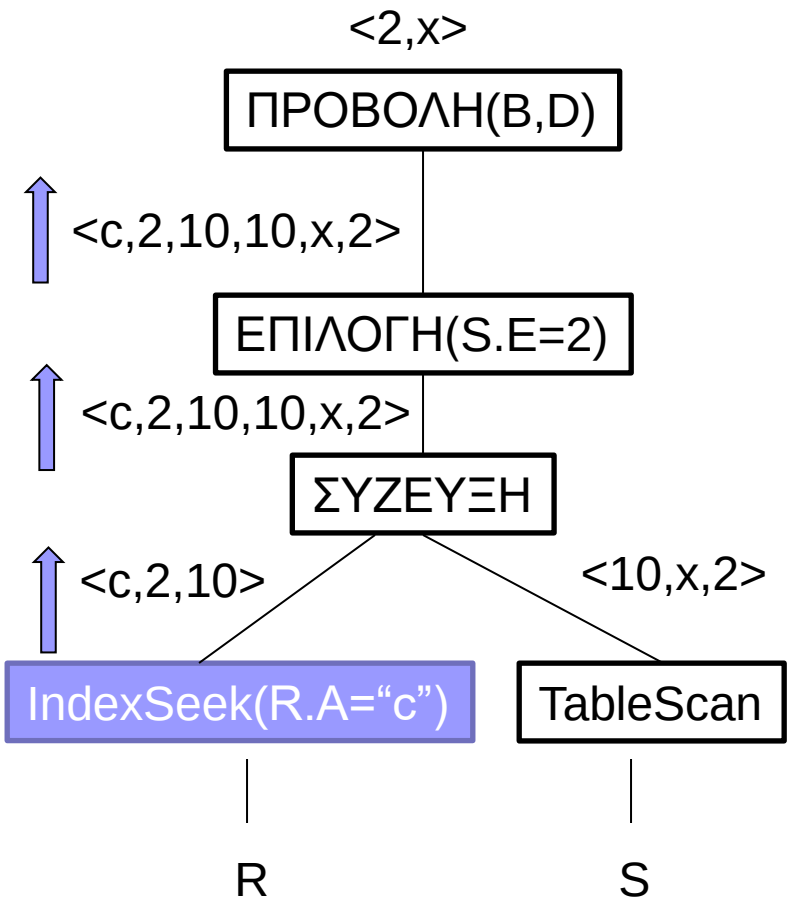
Μοντέλο Επαναλήπτη (iterator model)

- Ο κάθε τελεστής ζητάει από τον επόμενο του (παιδί του στο δέντρο) μία εγγραφή την οποία επεξεργάζεται και παραδίνει στον πατέρα του



Μοντέλο Επαναλήπτη (iterator model)

- Ο κάθε τελεστής ζητάει από τον επόμενο του (παιδί του στο δέντρο) μία εγγραφή την οποία επεξεργάζεται και παραδίνει στον πατέρα του



Υλοποίηση μοντέλου επαναλήπτη

- Κάθε τελεστής υλοποιεί 3 συναρτήσεις
  - open(): αρχικοποιεί βοηθητικές δομές, συνδέεται με τον αποκάτω τελεστή, αν υπάρχει
  - getNext(): επιστρέφει την επόμενη εγγραφή στο αποτέλεσμα του
  - close(): κλείνει αρχεία, διαγράφει βοηθητικές δομές, κλείνει τον/τους τελεστές παιδιά του στο δέντρο

Αποτίμηση μοντέλου επαναλήπτη

- Η εκτέλεση της επερώτησης ξεκινάει από τον τελεστή που βρίσκεται στη ρίζα του δέντρου
- Κάθε τελεστής καλεί τις συναρτήσεις του επόμενου (προς τα κάτω) τελεστή
  - “Pull” model
  - Κλήση συναρτήσεων από πάνω προς τα κάτω
  - Ροή αποτελεσμάτων από κάτω προς τα πάνω

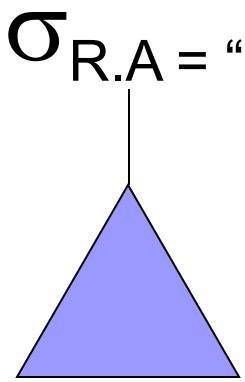
Επαναλήπτης για TableScan

```
Open() {
  /** initialize variables */
  b = first block of R;
  t = first tuple in block b;
}
```

```
Close() {
  /** nothing to be done */
}
```

```
GetNext() {
  IF (t is past last tuple in block b) {
      set b to next block;
      IF (there is no next block)
          /** no more tuples */
          RETURN EOT;
      ELSE t = first tuple in b;
  }
  /** return current tuple */
  oldt = t;
  set t to next tuple in block b;
  RETURN oldt;
}
```

Επαναλήπτης για Select

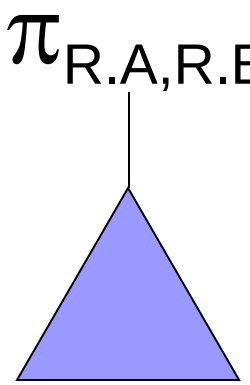



```
Open() {
  /** initialize child */
  Child.Open();
}

Close() {
  /** inform child */
  Child.Close();
}
```

```
GetNext() {
  LOOP:
      t = Child.GetNext();
      IF (t == EOT) {
        /** no more tuples */
        RETURN EOT;
      }
      ELSE IF (t.A == “c”)
        RETURN t;
  ENDLOOP:
}
```

Επαναλήπτης για Project




```
Open() {
  /** initialize child */
  Child.Open();
}

Close() {
  /** inform child */
  Child.Close();
}
```

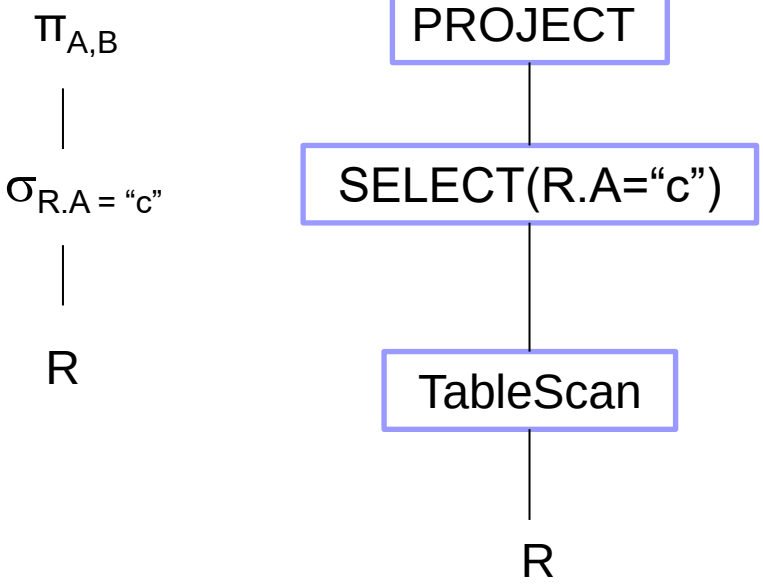
```
GetNext() {
   t = Child.GetNext();
   IF (t == EOT) {
        /** no more tuples */
        RETURN EOT;
    }
    ELSE
        RETURN t.A,t.B;
}
```

Παράδειγμα

R:

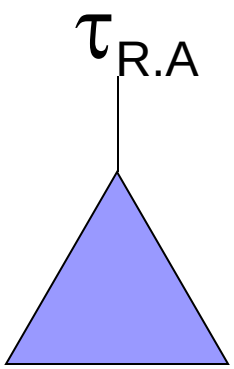
A	B	C
a	1	10
c	2	20
d	3	30
e	4	40

Select A,B From R Where A="c"




```
Project.open()
  Select.open()
    TableScan.open()
Project.getNext()          ←  <c,2>
  Select.getNext()         ←  <c,2,20>
    TableScan.getNext()    ←  <a,1,10>
    TableScan.getNext()    ←  <c,2,20>
Project.getNext()          ←  EOT
  Select.getNext()         ←  EOT
    TableScan.getNext()    ←  ???
    TableScan.getNext()    ←  ???
    TableScan.getNext()    ←  EOT
Project.close()
  Select.close()
    TableScan.close()
```

Επαναλήπτης για Sort




```
GetNext() {
   IF (more tuples)
      RETURN next tuple in order;
   ELSE RETURN EOT;
}
```

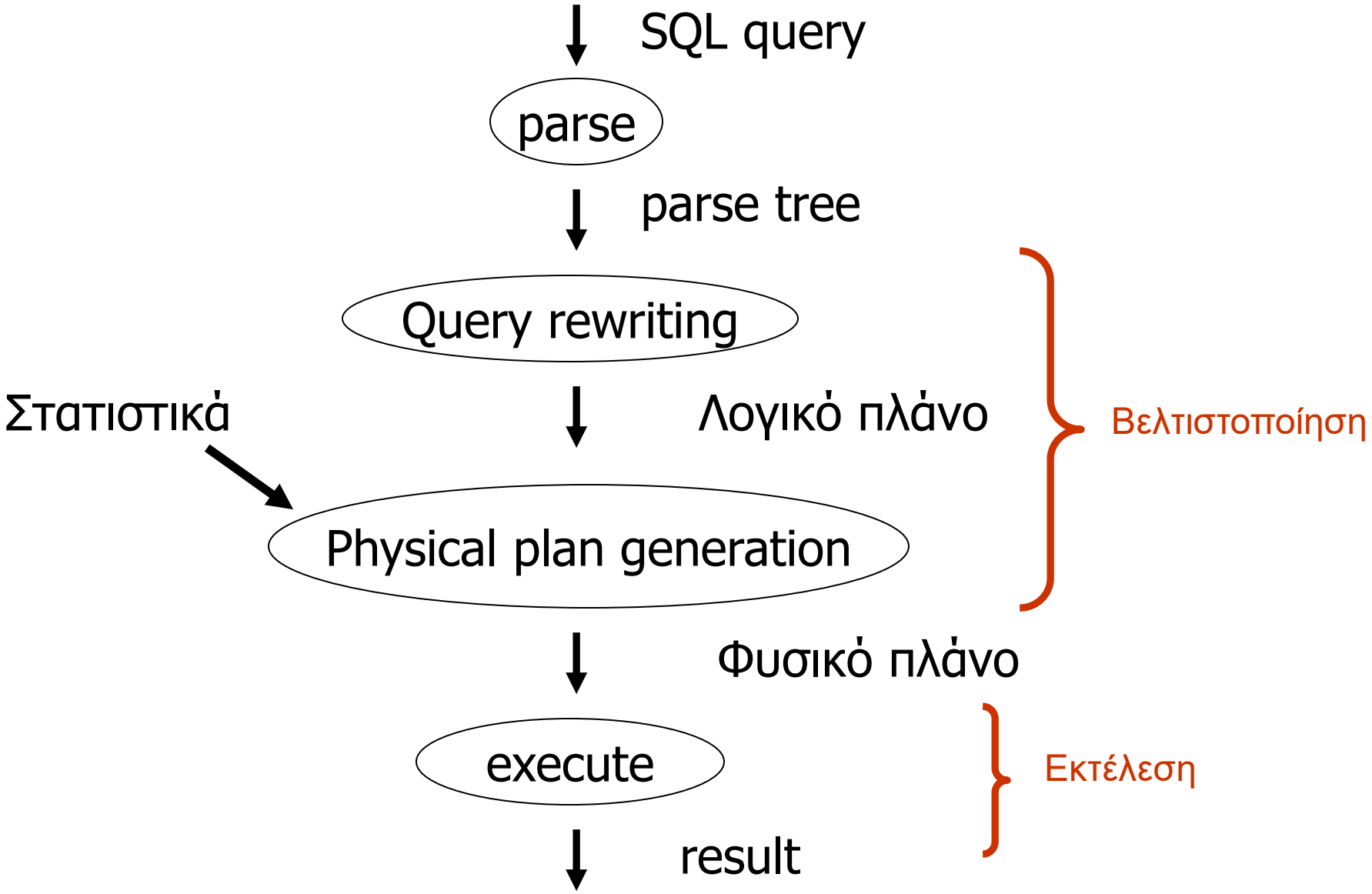
```
Open() {
  /** Bulk of the work is here */
  Child.Open();
  Read all tuples from Child
     and sort them
}
```

```
Close() {
  /** inform child */
  Child.Close();
}
```

Ταξινόμηση Τελεστών

	Tuple-at-a-time	Full-relation
Unary	Select	Sort
Binary		Difference

Επισκόπηση Επεξεργασίας Επερωτήσεων


SQL query
parse
parse tree
Query rewriting
Στατιστικά
Λογικό πλάνο
Βελτιστοποίηση
Physical plan generation
Φυσικό πλάνο
execute
Εκτέλεση
result

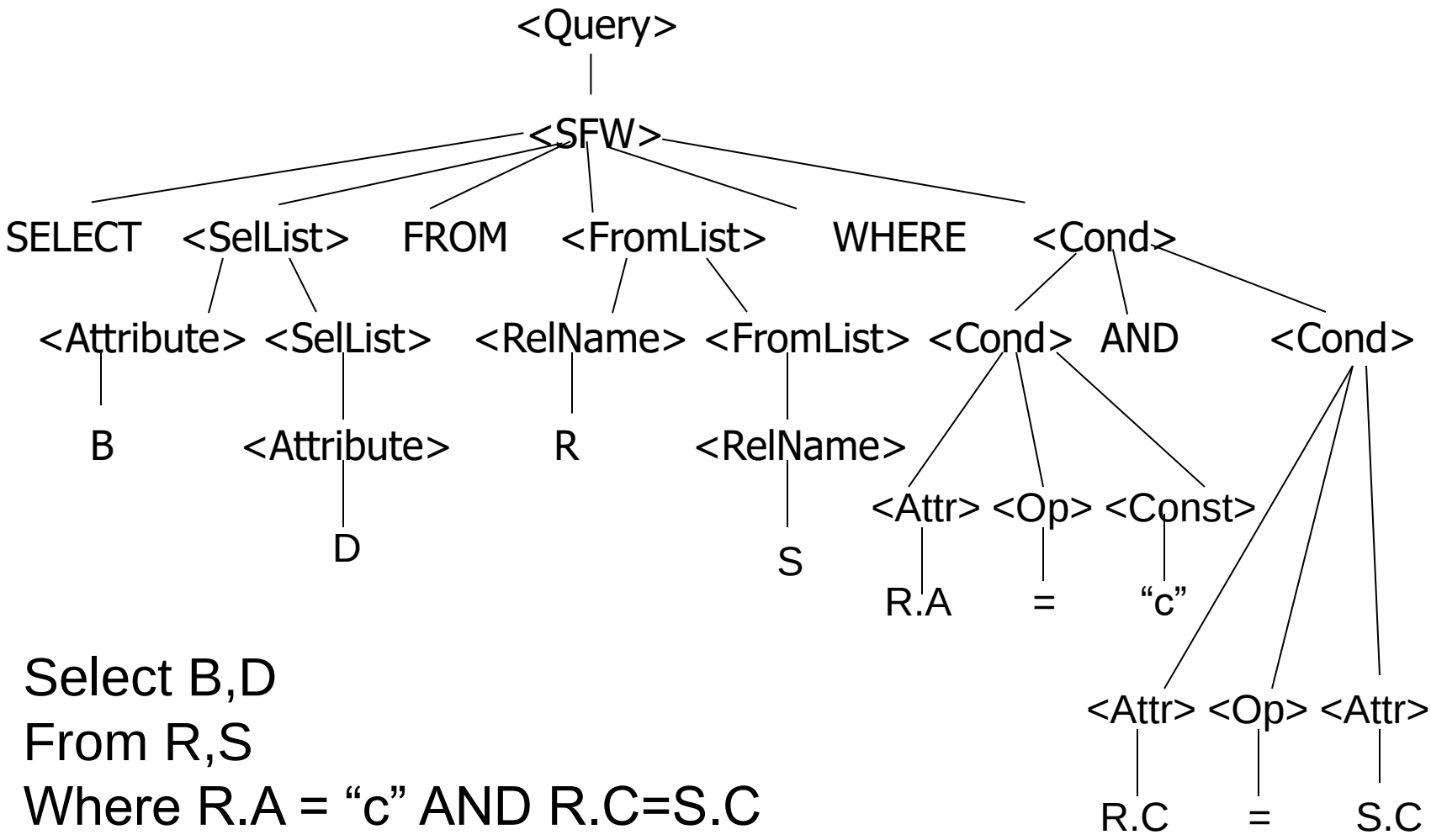
Παράδειγμα

Select B,D

From R,S

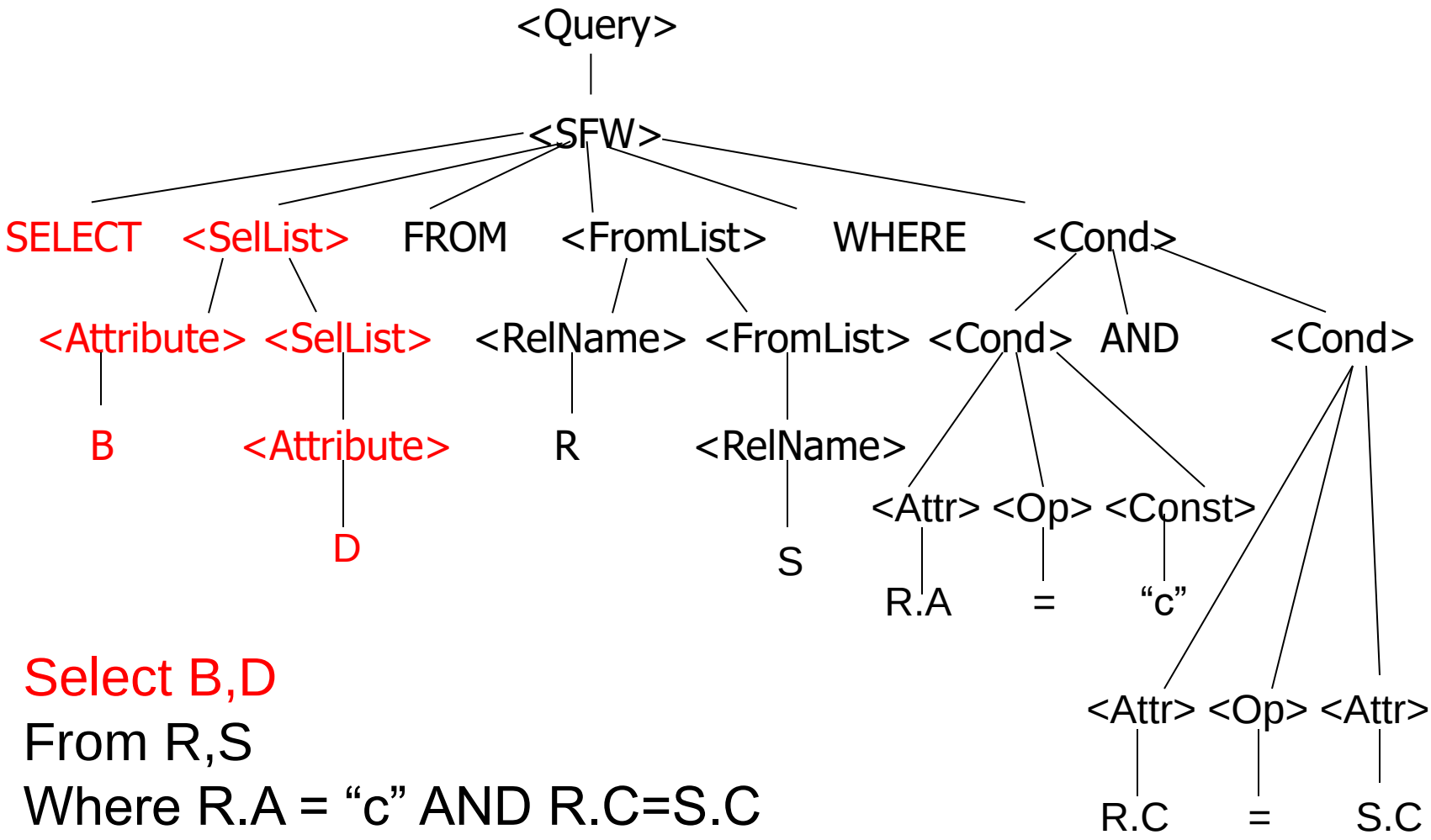
Where R.A = “c” AND R.C=S.C

Parse Tree



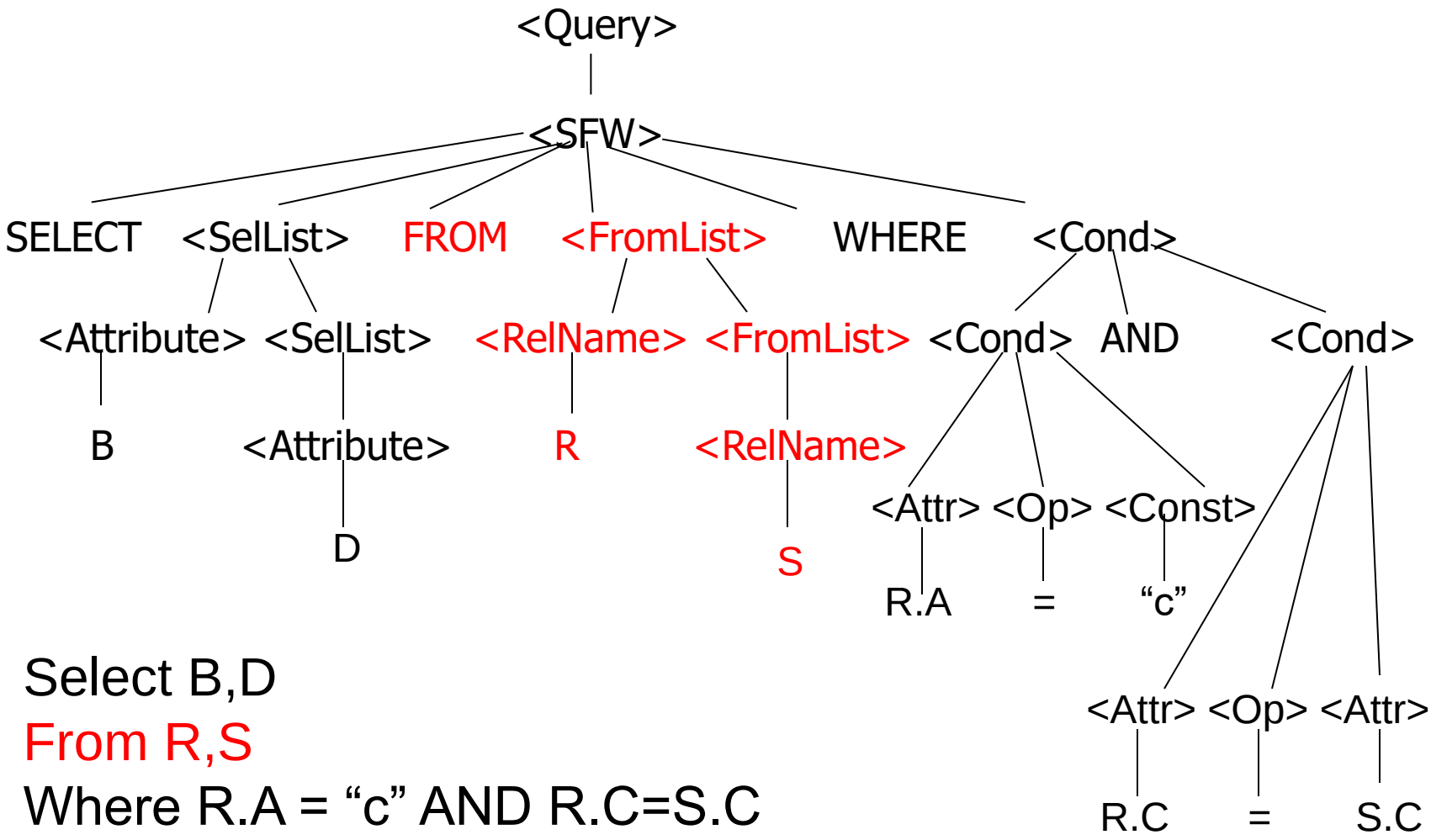
Select B,D
From R,S
Where R.A = "c" AND R.C=S.C

Parse Tree



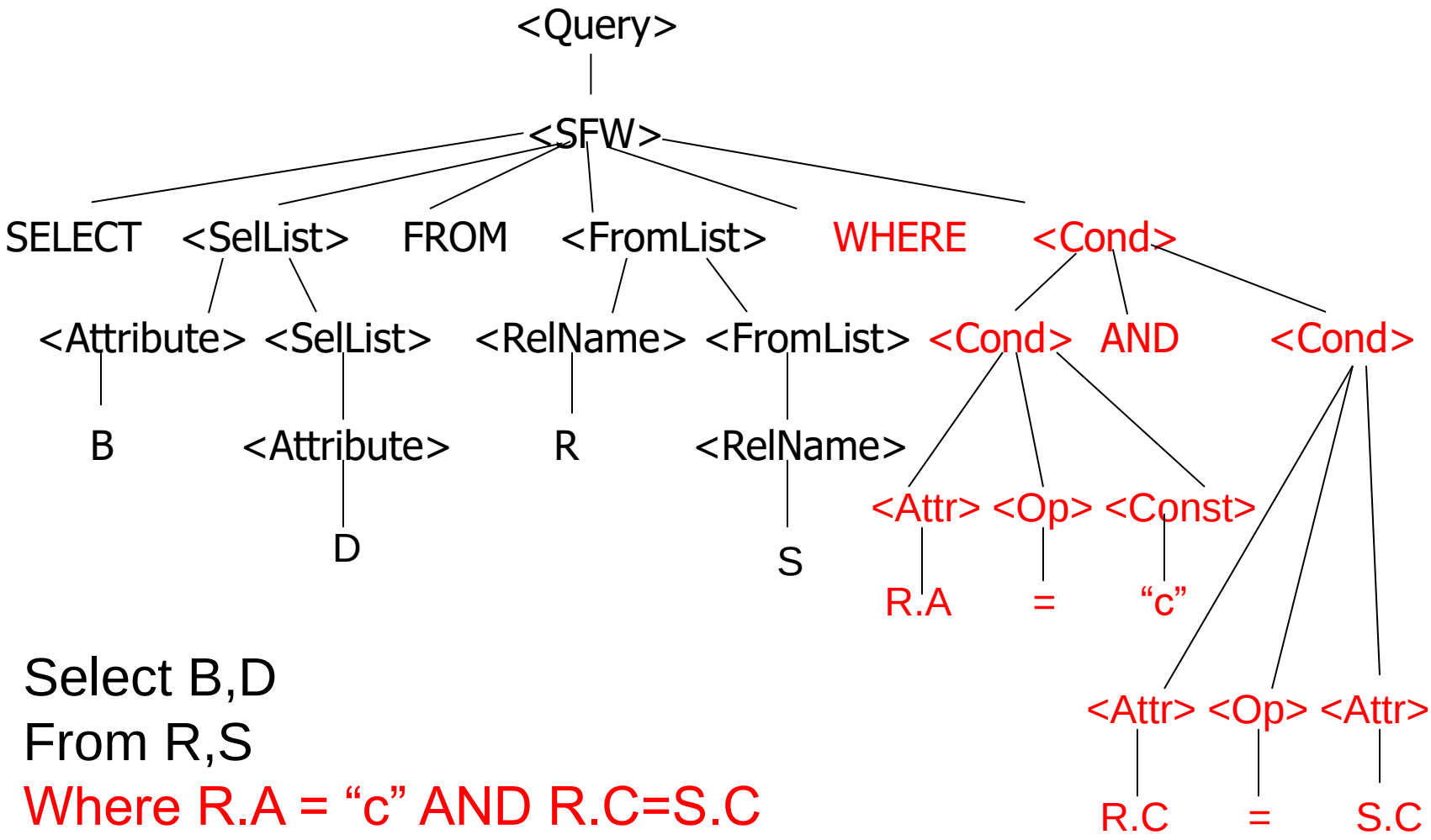
Select B,D
From R,S
Where R.A = "c" AND R.C=S.C

Parse Tree



Select B,D
From R,S
Where R.A = "c" AND R.C=S.C

Parse Tree



Select B,D
From R,S
Where R.A = "c" AND R.C=S.C

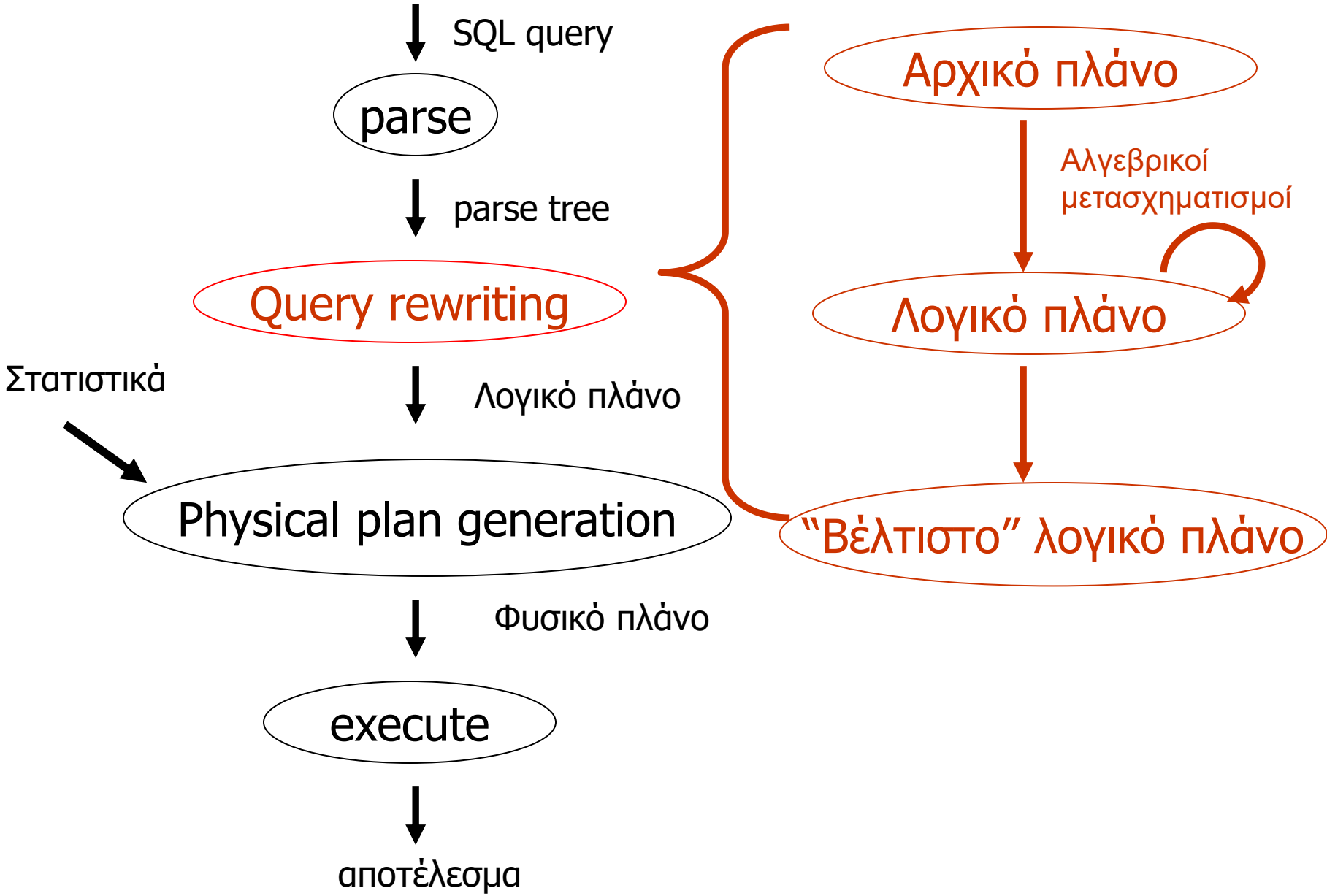
Μαζί με το Parsing …

- Αντικατάσταση όψεων με τον ορισμό τους (expand views)
- Σημασιολογικοί έλεγχοι
  - Τα γνωρίσματα που αναφέρονται στην επερώτηση υπάρχουν στις δηλωθέντες σχέσεις?
  - Type checking, πχ: R.A > 17.5
    - Τι γίνεται αν το R.A έχει ορισθεί ως string?

INFORMATION_SCHEMA Views

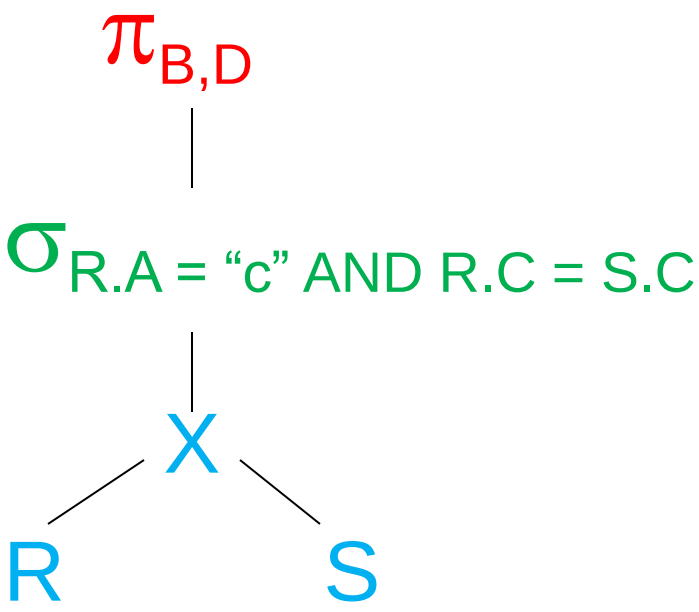
- ANSI-Standard views that provide information about the database schema
- Example (list all columns and their data types for table Employees):

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name='Employees';
```


SQL query
parse
parse tree
Query rewriting
Στατιστικά
Λογικό πλάνο
Physical plan generation
Φυσικό πλάνο
execute
αποτέλεσμα
Αρχικό πλάνο
Αλγεβρικοί μετασχηματισμοί
Λογικό πλάνο
"Βέλτιστο" λογικό πλάνο

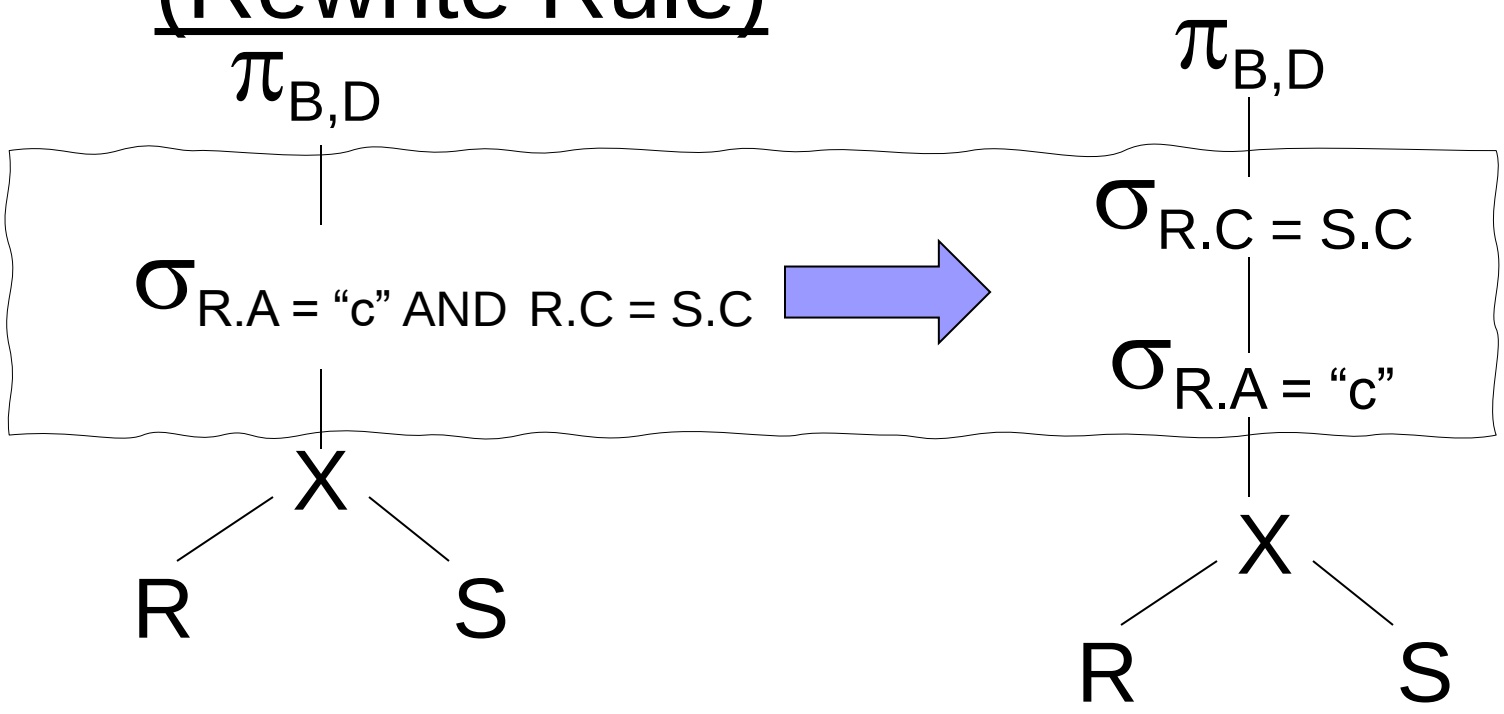
Αρχικό λογικό πλάνο

Select B,D
From R,S
Where R.A = “c”
AND R.C=S.C



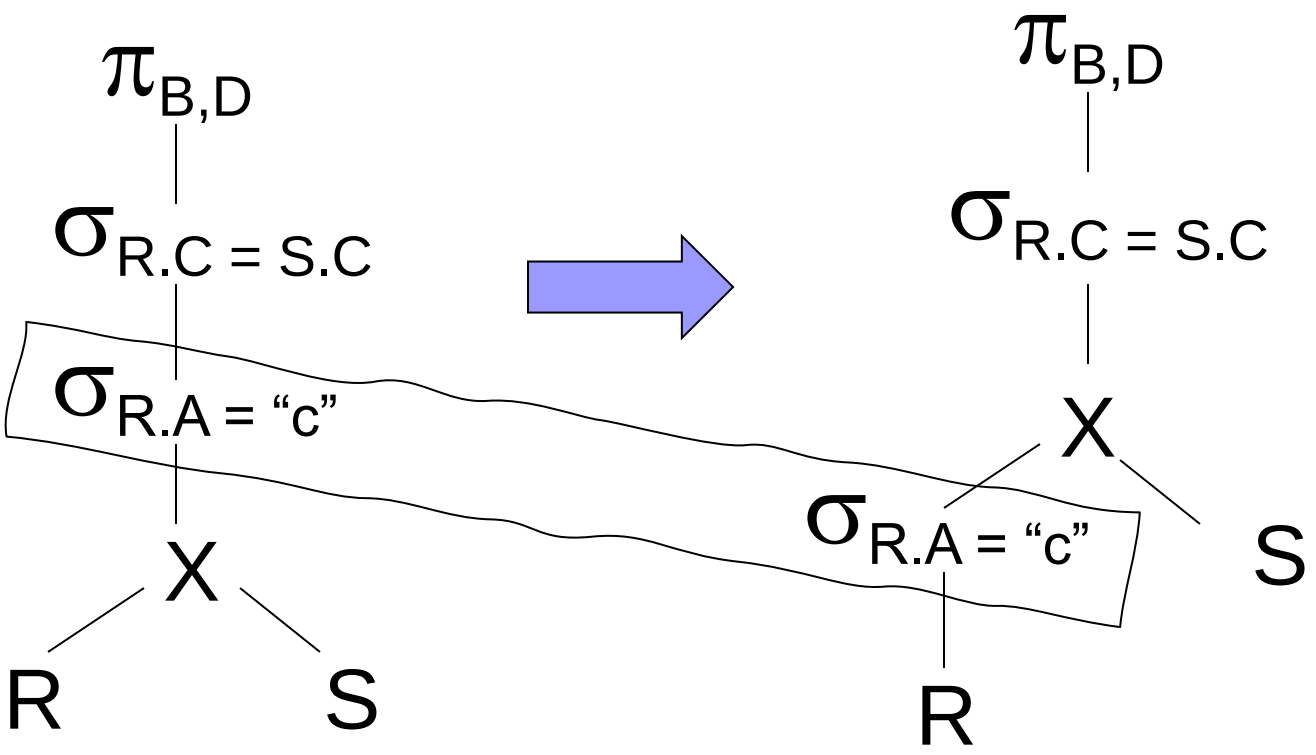
$$\Pi_{B,D}\,[\,\sigma_{R.A=\text{“c”}\ \text{AND}\ R.C\,=\,S.C}\,(RXS)]$$

Αλγεβρικός Μετασχηματισμός (Rewrite Rule)



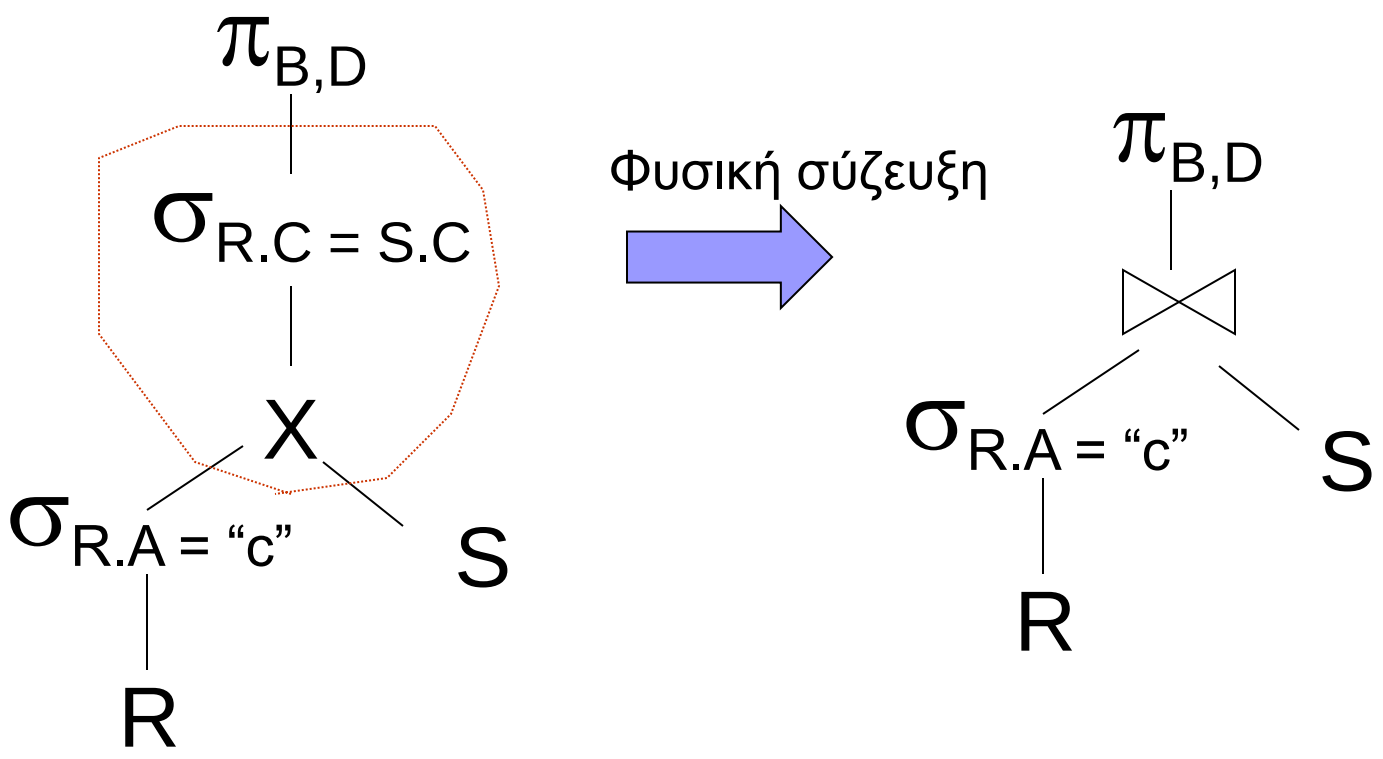
$$\Pi_{B,D}\,[\,\sigma_{R.C=S.C}\,[\sigma_{R.A=\text{“c”}}(R \times S)]]$$

Rewrite Rule (2)

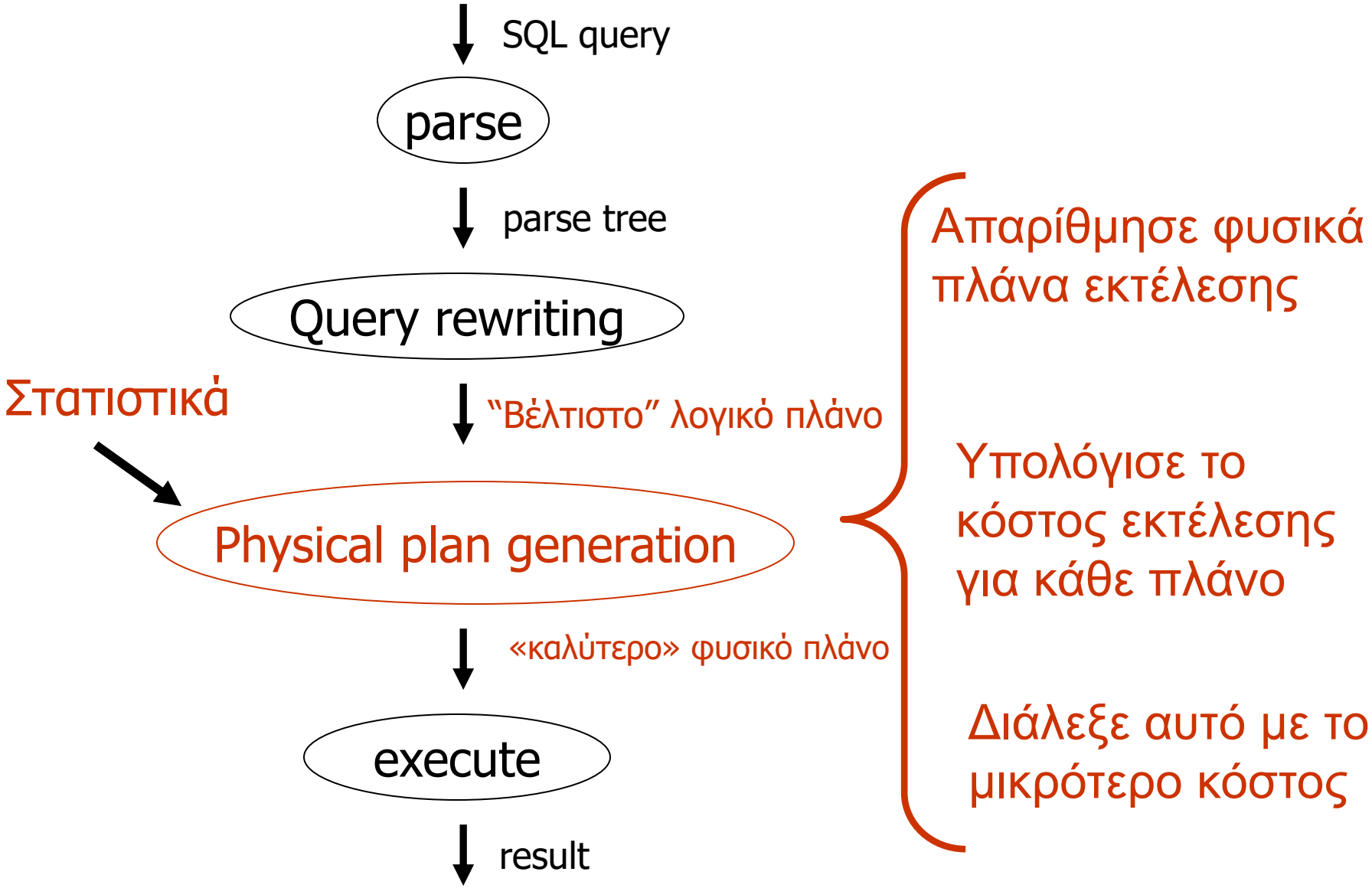


$$\Pi_{B,D}\,[\,\sigma_{R.C=S.C}\,[\sigma_{R.A=\text{“c”}}(R)]\ X\ S]$$

Rewrite Rule (3)



$$\Pi_{B,D}\,[[\sigma_{R.A=\text{“c”}}(R)] \bowtie S]$$


SQL query
parse
parse tree
Query rewriting
Στατιστικά
"Βέλτιστο" λογικό πλάνο
Physical plan generation
«καλύτερο» φυσικό πλάνο
execute
result
Απαρίθμησε φυσικά πλάνα εκτέλεσης
Υπολόγισε το κόστος εκτέλεσης για κάθε πλάνο
Διάλεξε αυτό με το μικρότερο κόστος

Απαρίθμηση φυσικών πλάνων

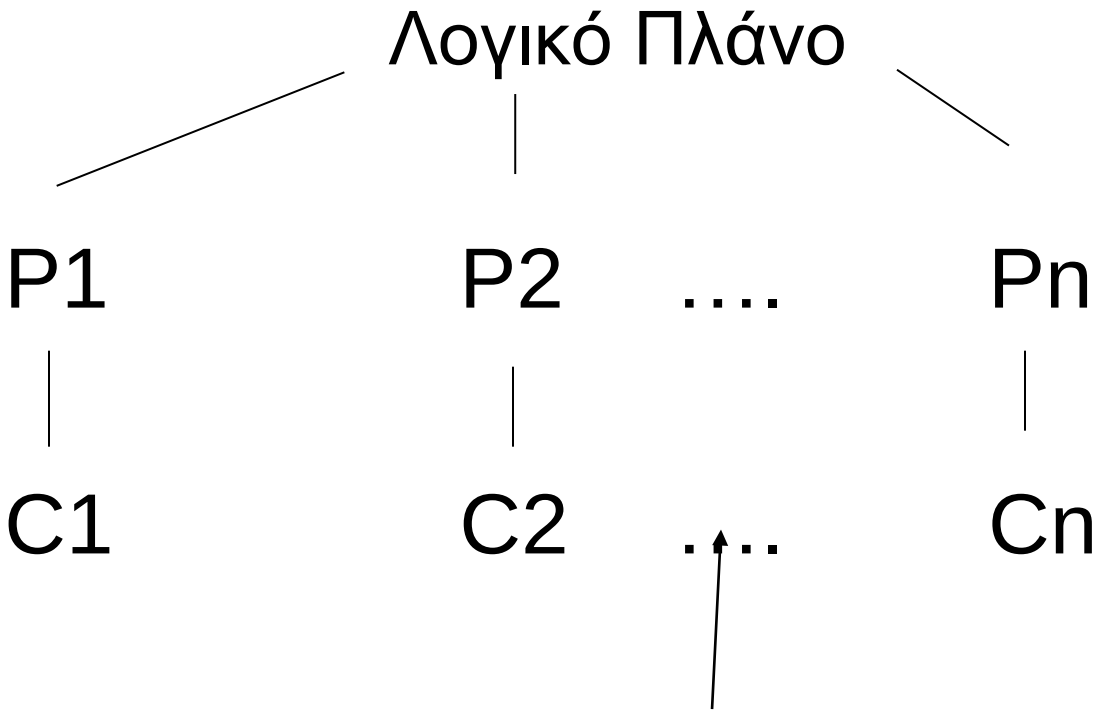
- Το ΣΔΒΔ έχει στις βιβλιοθήκες του έναν ή περισσότερους αλγόριθμους υλοποίησης για κάθε τελεστή της σχεσιακής άλγεβρας
- Πχ Σύζευξη (Join)
  - Nested Loop Join (NLJ)
  - Sort Merge Join (SMJ)
  - Hash Join
  - Zig-Zag Join
  - …

Πως επιλέγω ανάμεσα τους?

Κόστος εκτέλεσης

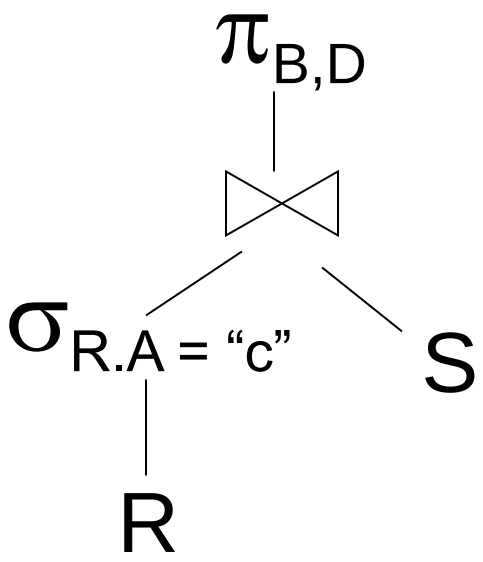
- Πως ορίζεται;
  - Χρόνος εκτέλεσης επερώτησης;
  - Χρόνος = CPU time + I/O time
- Εξαρτάται από πολλούς παράγοντες
  - Διάταξη δεδομένων στη μνήμη ή στο δίσκο
  - Αριθμός εγγραφών/μέγεθος ενδιάμεσων αποτελεσμάτων
  - Διαθέσιμη μνήμη, το διαθέσιμο υλικό (hardware)
  - Ευρετήρια
  - ….

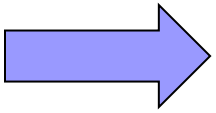
Απαρίθμηση Φυσικών Πλάνων

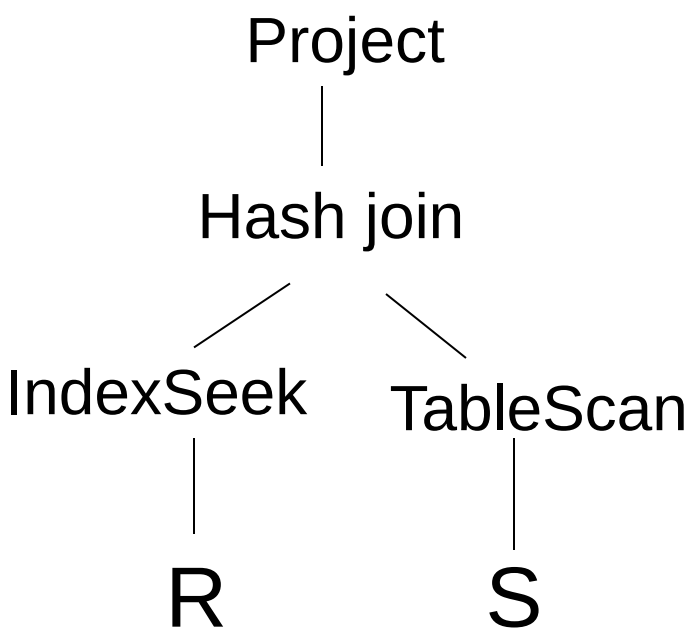


Διάλεξε αυτό με το μικρότερο κόστος

Δημιουργία Φυσικού Πλάνου







“Καλύτερο” Λογικό Πλάνο

Γιατί επιλέγουμε 1 λογικό πλάνο;

- Θα μπορούσαμε να απαριθμήσουμε όλα τα λογικά πλάνα τα οποία προκύπτουν από τους αλγεβρικούς μετασχηματισμούς που γνωρίζουμε
  - Για κάθε ένα από αυτά χρειάζεται απαρίθμηση όλων των φυσικών πλάνων που μπορεί να προκύψουν
  - Τεράστιος αριθμός πλάνων

Γιατί επιλέγουμε 1 λογικό πλάνο;

- Ουσιαστικά η βελτιστοποίηση είναι ένα πρόβλημα αναζήτησης σε ένα μεγάλο χώρο λύσεων
  - Δεν έχει νόημα να σπαταλήσουμε πχ 5 λεπτά για να βρούμε ένα πλάνο το οποίο βελτιώνει το χρόνο εκτέλεσης της επερώτησης κατά 10 δευτερόλεπτα!
  - Συχνά, η τελική επιλογή πρέπει να γίνει σε κλάσματα του δευτερολέπτου

Τι γίνεται στην πράξη

- Υπάρχουν διαφορετικοί αλγόριθμοι ευρετικής αναζήτησης
- Συχνά εφαρμόζονται πρώτα αλγεβρικοί μετασχηματισμούς που (*σχεδόν πάντα*) οδηγούν σε καλύτερα φυσικά πλάνα εκτέλεσης
  - Πχ αντικατάσταση καρτεσιανού γινομένου με σύζευξη
  - Αποτίμηση των επιλογών όσο ποιο νωρίς γίνεται (push selection down)

Αλγεβρικοί μετασχηματισμοί (Query rewrite)

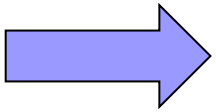
- Μετασχημάτισε το λογικό πλάνο σε ένα ισοδύναμο
- Συνήθως **δεν** κοιτάμε τα στατιστικά σε αυτό τα βήμα (μέγεθος σχέσεων, κατανομή τιμών γνωρισμάτων)
- Λαμβάνουμε υπόψη περιορισμούς που γνωρίζουμε
  - Κλειδιά, περιορισμοί (constraints),…

Όσα αναφέρουμε είναι γενικές κατευθύνσεις. Οι σχεδιαστές ενός ΣΔΒΔ μπορούν να κάνουν τα δικά τους «κόλπα»

Μετασχηματισμοί με επιλογή (select - σ)

$$\sigma_{p1 \wedge p2}(R) = \sigma_{p1}\,[\,\sigma_{p2}\,(R)]$$

Πχ:

$\sigma_{R.A = \text{“c”} \text{ AND } R.C = S.C}$ 

$$\sigma_{R.C = S.C}$$
$$|$$
$$\sigma_{R.A = \text{“c”}}$$

∧ = λογικό «και»

Ισχύει το παρακάτω?

$$\sigma_{p1 \vee p2}(R) = \sigma_{p1}(R) \cup \sigma_{p2}(R)$$

Έστω ο πίνακας R με εγγραφές R={a,a,b,b,b,c}

Συνθήκη P1 ικανοποιείται από τις εγγραφές a,b
Συνθήκη P2 ικανοποιείται από τις εγγραφές b,c

$$\sigma_{p1 \vee p2}(R) = \{a,a,b,b,b,c\}$$

$$\sigma_{p1}(R) = \{a,a,b,b,b\}$$

$$\sigma_{p2}(R) = \{b,b,b,c\}$$

$$\sigma_{p1}(R) \cup \sigma_{p2}(R) = \{a,a,b,b,b,b,b,b,c\}$$

Μετασχηματισμοί με χρήση προβολής (project – π)

Έστω: X = υποσύνολο γνωρισμάτων σχέσης R

Y = υποσύνολο* γνωρισμάτων σχέσης R

(*με μη κοινά γνωρίσματα με το X)

και XY = X U Y (η ένωση τους)

$\pi_{xy}(R) = \pi_x[\pi_y(R)]$

σ μαζί με ⋈

Έστω p = συνθήκη μόνο στα γνωρίσματα της R

q = συνθήκη μόνο στα γνωρίσματα της S

m = συνθήκη σε γνωρίσματα της R **και** της S

$$\sigma_p (R \bowtie S) = [\sigma_p(R)] \bowtie S$$

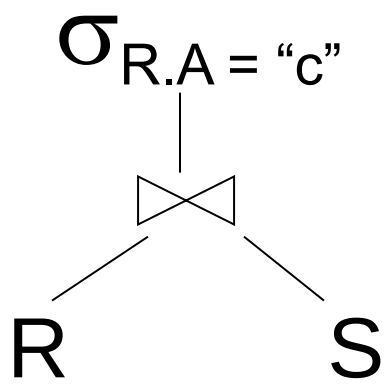
$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q(S)]$$

Στη βιβλιογραφία αναφέρεται ως: Pushing selections down

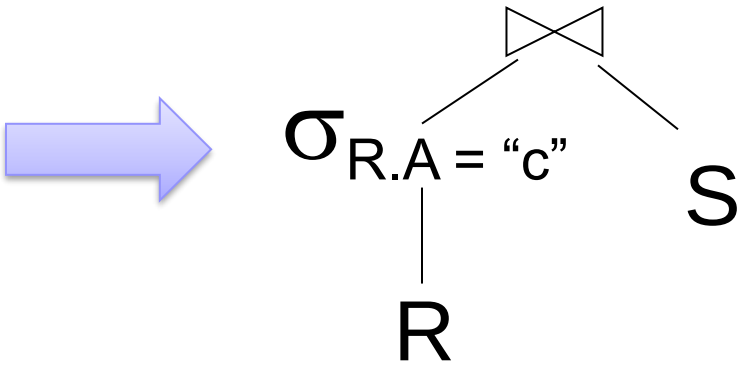
Σε τι μας χρησιμεύει?

Ας σκεφτούμε

Λογικό πλάνο Α:



Λογικό πλάνο Β:



- Έστω ότι η σχέση R έχει χιλιάδες εγγραφές αλλά μόνο μερικές δεκάδες ικανοποιούν τη συνθήκη R.A=“c”
  - Το πλάνο Β οδηγεί σε σημαντική μείωση του αριθμού των εγγραφών που θα επεξεργαστεί η σύζευξη.
  - Με το πλάνο Β μπορώ να χρησιμοποιήσω ευρετήριο στο γνώρισμα R.A, εφόσον υπάρχει
    - Γιατί αυτό δε γίνεται στο πλάνο Α;

Μερικά ακόμα παραδείγματα

$\sigma_{p \wedge q} (R \bowtie S) = ?$

$\sigma_{p \wedge q \wedge m} (R \bowtie S) = ?$

Λύση

$$\sigma_{p \wedge q} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) =$$

$$\sigma_m \left[(\sigma_p R) \bowtie (\sigma_q S) \right]$$

Ο πρώτος μετασχηματισμός αναλυτικά

$$\sigma_{p \wedge q} (R \bowtie S) =$$

$$\sigma_{p} [\sigma_{q} (R \bowtie S)] =$$

$$\sigma_{p} [R \bowtie \sigma_{q} (S)] =$$

$$[\sigma_{p} (R)] \bowtie [\sigma_{q} (S)]$$

Προβολή με επιλογή (π,σ)

Έστω x = υποσύνολο γνωρισμάτων σχέσης R

z = γνωρίσματα της R στη λογική έκφραση P

$$\pi_x[\sigma_p(R)] = \pi_x\{\sigma_p[\overset{\pi_{xz}}{\cancel{\pi_x}}(R)]\}$$

Παράδειγμα

- Movies(Title,Actor,Trailer)
- $\pi_{Title}[\sigma_{Actor=\text{“Pitt”}}(Movies)]$

  =

  $\pi_{Title}[\sigma_{Actor=\text{“Pitt”}}[\pi_{Title,Actor}(Movies)]]$

Τι κερδίζουμε με τον παραπάνω μετασχηματισμό?

Rules: π, $\bowtie$ combined

Let x = subset of R attributes

y = subset of S attributes

z = intersection of R,S attributes

$$\pi_{xy}(R \bowtie S) =$$

$$\pi_{xy}\{[\pi_{xz}(R)] \bowtie [\pi_{yz}(S)]\}$$

$$\pi_{xy}\{\sigma_p\ (R \bowtie S)\} =$$

$$\pi_{xy}\{\sigma_p\ [\pi_{xz'}(R) \bowtie \pi_{yz'}(S)]\}$$

$$z' = z\ U\ \{\text{attributes used in P}\}$$

Ανάλογα οι συνδυασμοί σ, π με Χ

Πχ: p = συνθήκη σε γνωρίσματα της R

$$\sigma_p (R \times S) = \ ?$$

σ μαζί με U:

p συνθήκη σε γνωρίσματα των R,S

$\sigma_p(R \cup S) = \sigma_p(R) \cup \sigma_p(S)$

$\sigma_p(R - S) = \sigma_p(R) - S = \sigma_p(R) - \sigma_p(S)$

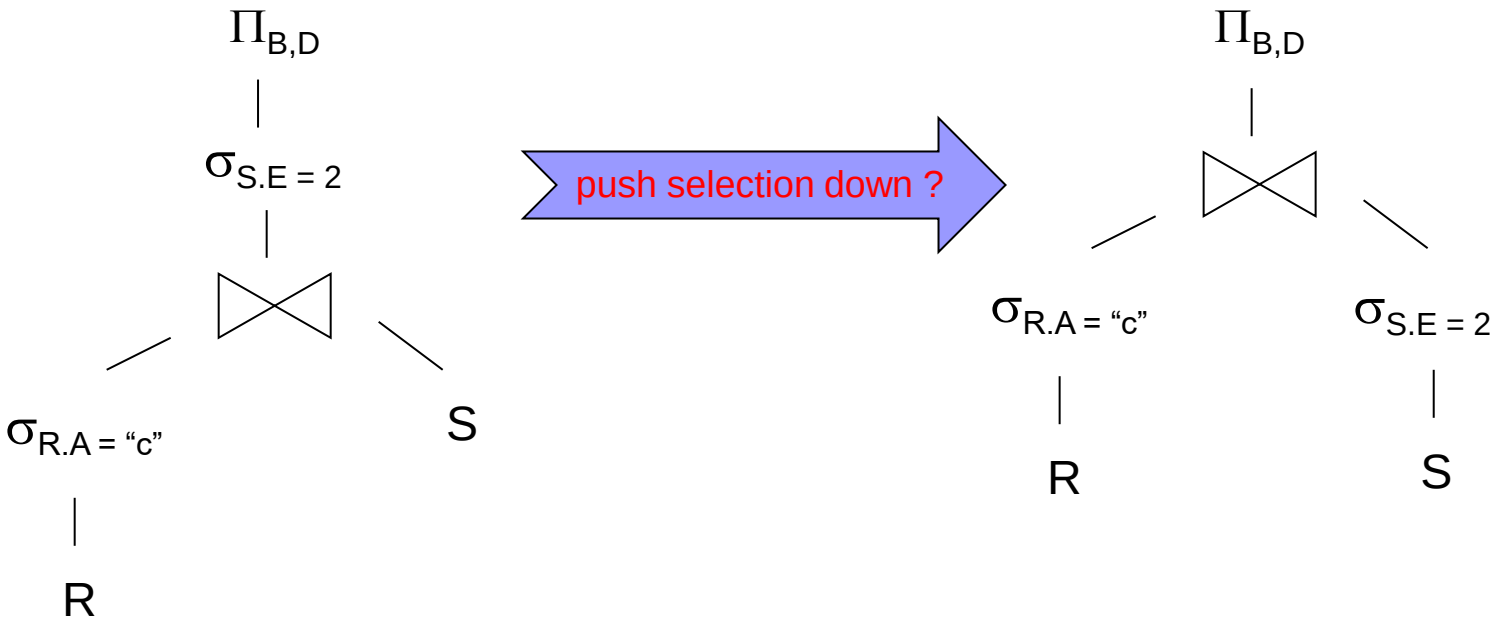
Ανακεφαλαίωση: γιατί τα παρακάτω είναι χρήσιμα?

- $\sigma_{p1 \wedge p2}(R) \rightarrow \sigma_{p1}[\sigma_{p2}(R)]$
- $\sigma_{p}(R \bowtie S) \rightarrow [\sigma_{p}(R)] \bowtie S$
- $R \bowtie S \rightarrow S \bowtie R$

Ανάλογα με τον αλγόριθμο υλοποίησης, το κόστος μπορεί να αλλάζει αν αλλάξουμε τη σειρά των σχέσεων

- $\pi_{x}[\sigma_{p}(R)] \rightarrow \pi_{x}\{\sigma_{p}[\pi_{xz}(R)]\}$

Συμφέρει πάντα?



Select B,D

From R,S

Where R.A = “c” AND S.E = 2 AND R.C=S.C

Προσοχή

- Κανένας μετασχηματισμός δεν είναι πάντα καλός
  - Δεν μπορώ να αποδείξω κάτι αν δεν ξέρω το περιεχόμενο και τη δομή της βάσης στην οποία αναφέρομαι, τα χαρακτηριστικά του δίσκου, τη μνήμη, τους αλγορίθμους υλοποίησης των τελεστών κοκ
- Συνήθως
  - Push selections down
  - Subqueries → Joins (Next)

SQL Query with an Uncorrelated Subquery (εντός ύλης)

Find the movies with stars born in 1960

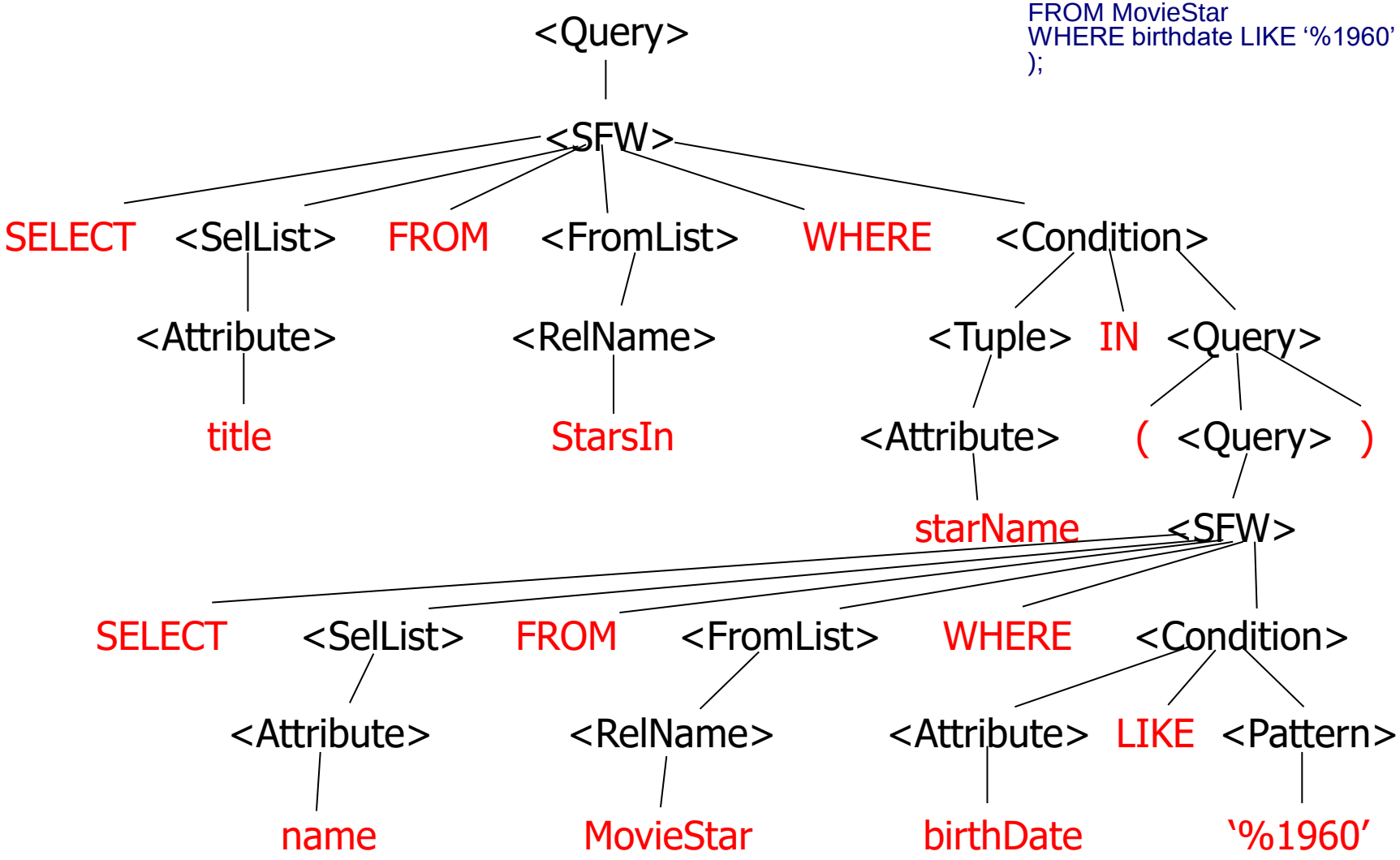
MovieStar(name, address, gender, birthdate)

StarsIn(title, year, starName)

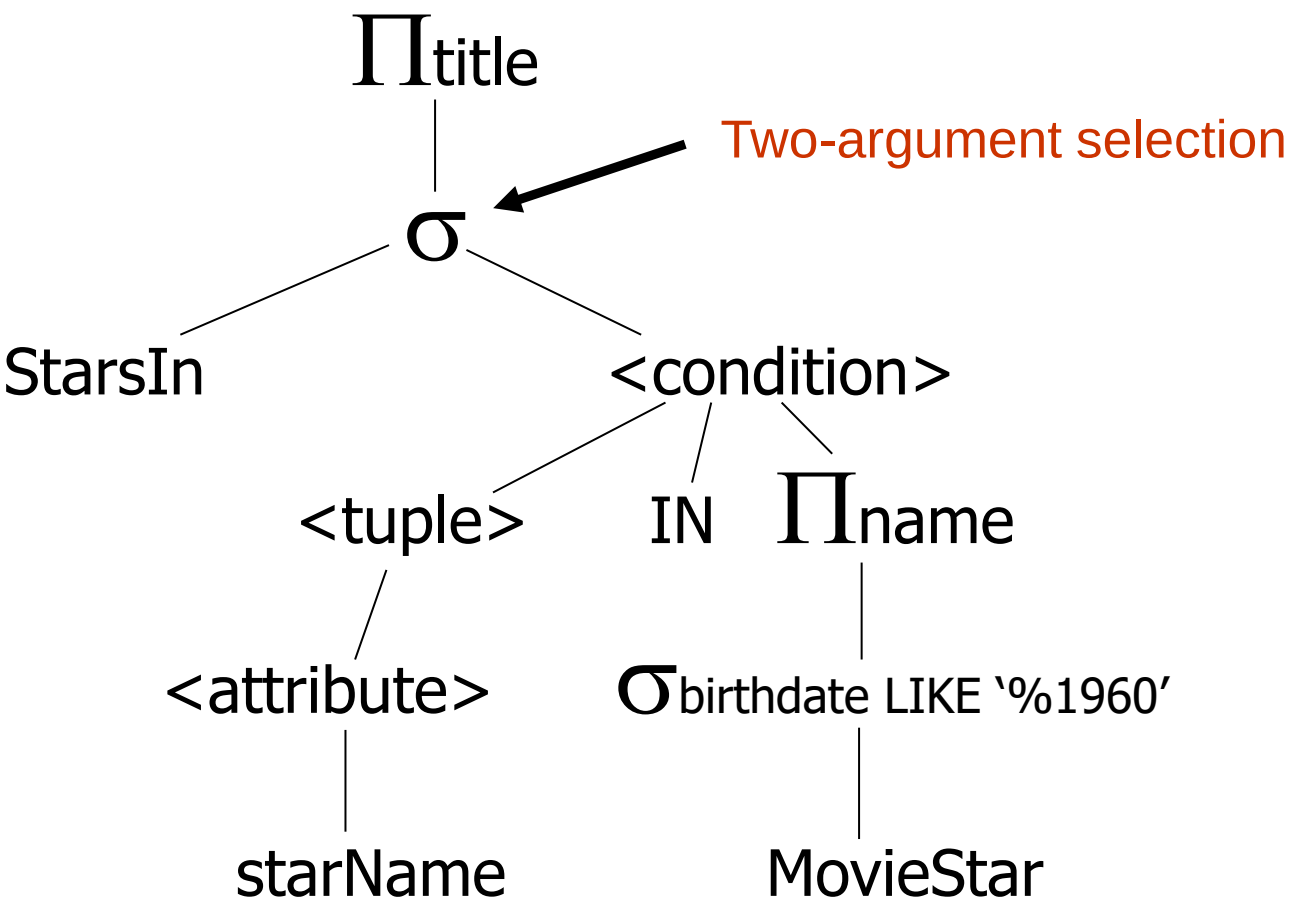
```
SELECT title
FROM StarsIn
WHERE starName IN (
        SELECT name
        FROM MovieStar
        WHERE birthdate LIKE '%1960'
);
```

Parse Tree

```
SELECT title
FROM StarsIn
WHERE starName IN (
SELECT name
FROM MovieStar
WHERE birthdate LIKE '%1960'
);
```

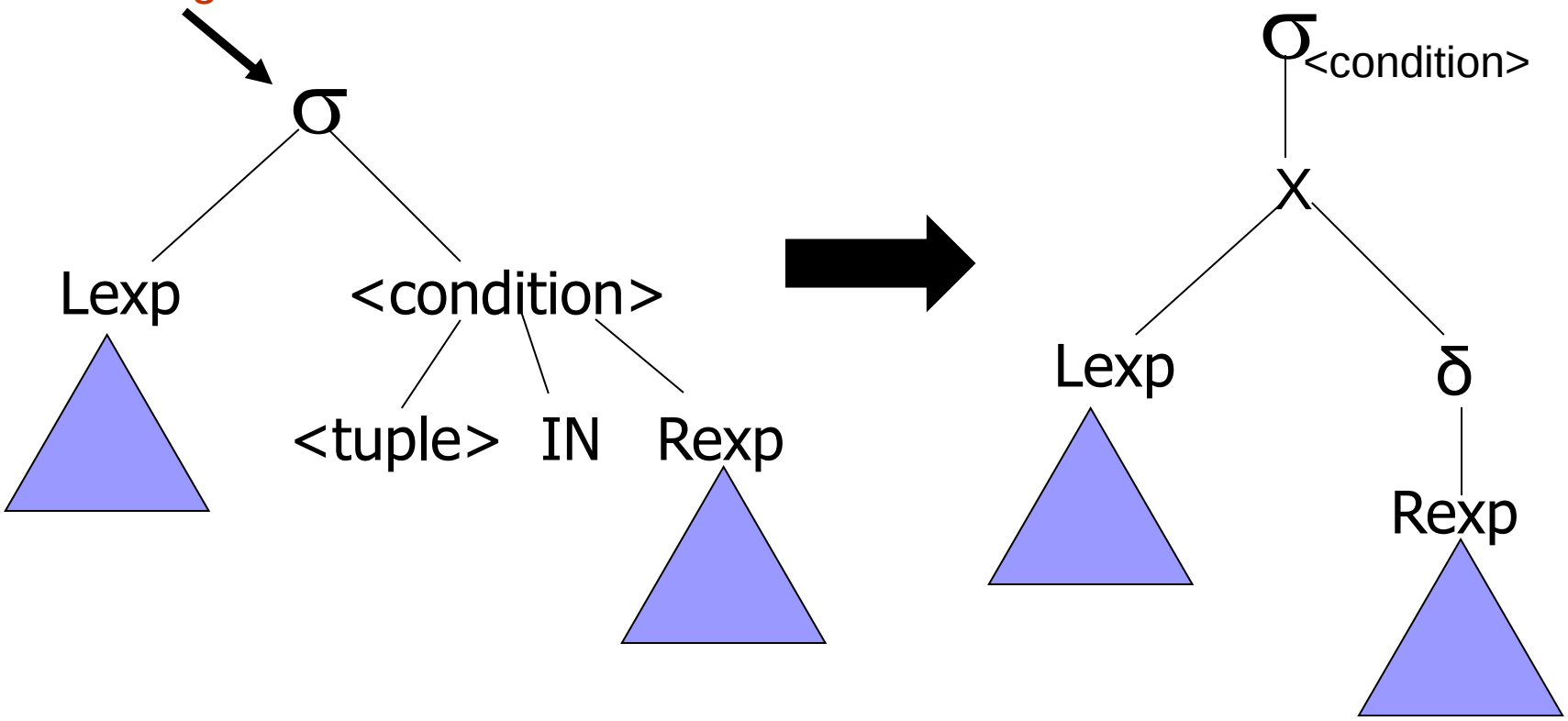


Generating Relational Algebra

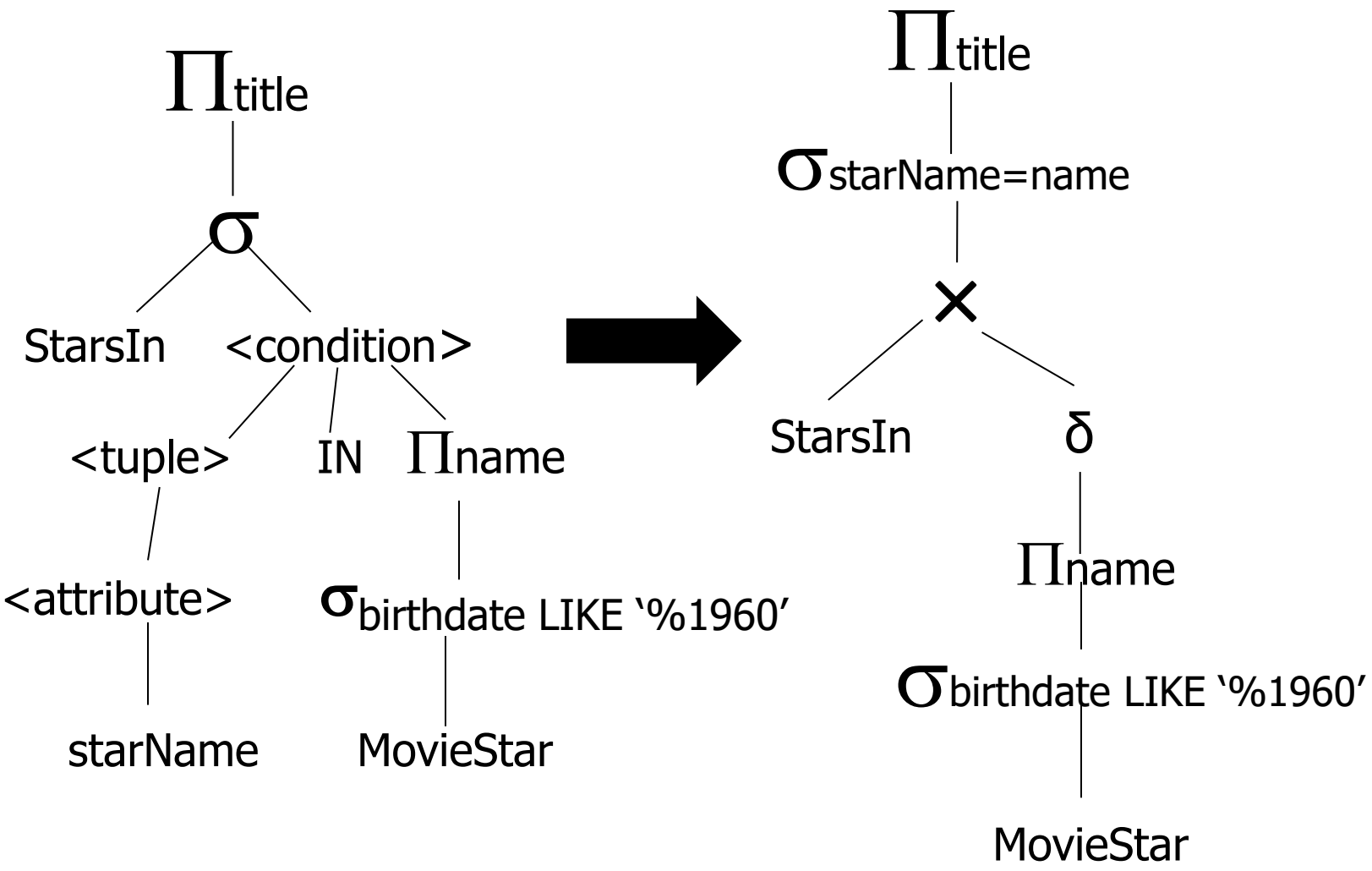


Rewrite Rule for Two-argument Selection with Conditions Involving IN

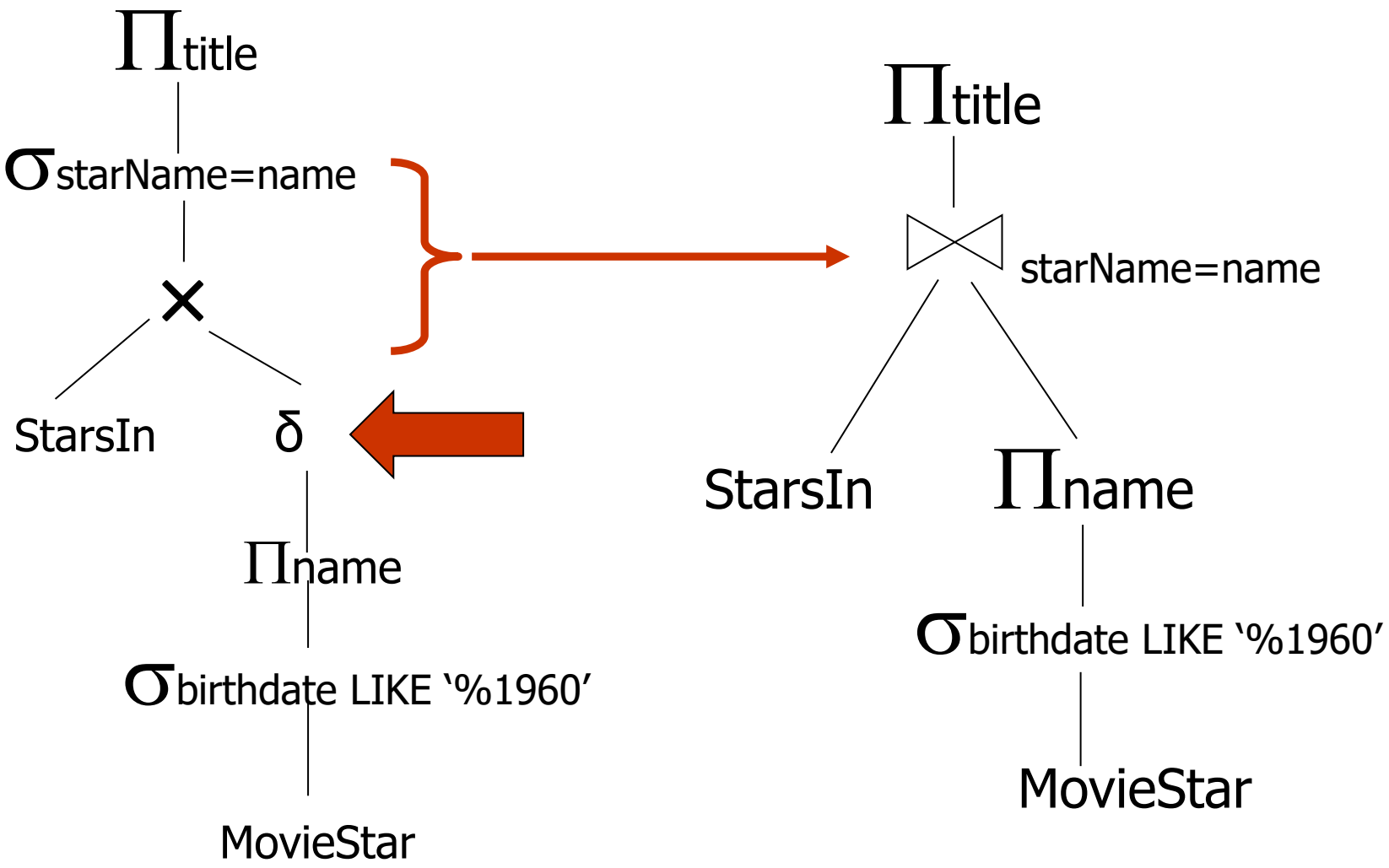
Two-argument selection



Applying the Rewrite Rule

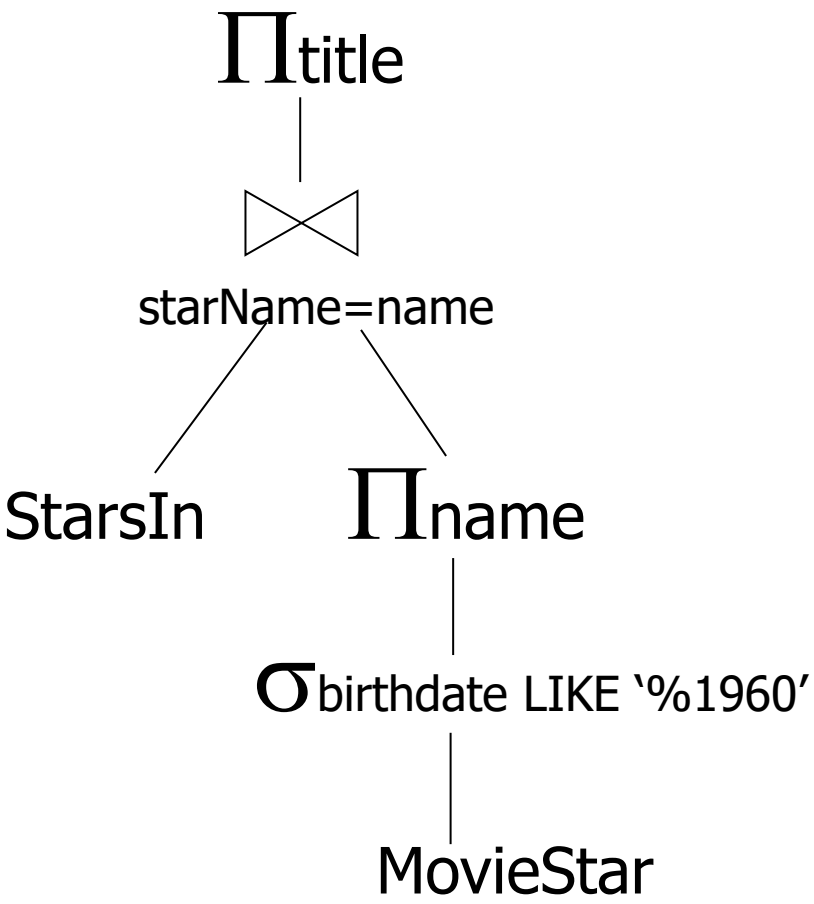


Improving the Logical Query Plan



Equivalent query

```
SELECT title
FROM StarsIn, MovieStar
WHERE starName=name
and birthdate LIKE '%1960';
```



Original Query

```
SELECT title
FROM StarsIn
WHERE starName IN (
  SELECT name
  FROM MovieStar
  WHERE birthdate LIKE '%1960'
);
```