

Exercises on query plans


Data Management

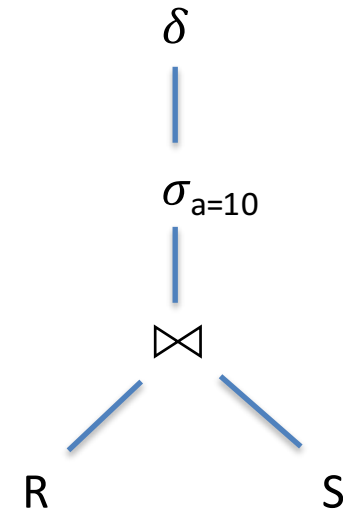
A.Y. 2017/18

Maurizio Lenzerini

Exercise 1

Let $R(a,b)$ and $S(b,c)$ two relations, and consider the following query and associated logical plan:

select distinct *
from R join S 
where $R.a = 10$



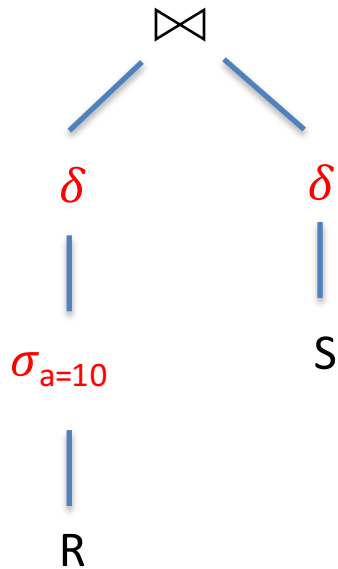
Also, let the statistics for $R(a,b)$ and $S(b,c)$ be as follows:

$T(R) = 5000$	$T(S) = 2000$
$V(R,a) = 50$	$V(S,b) = 200$
$V(R,b) = 100$	$V(S,c) = 100$

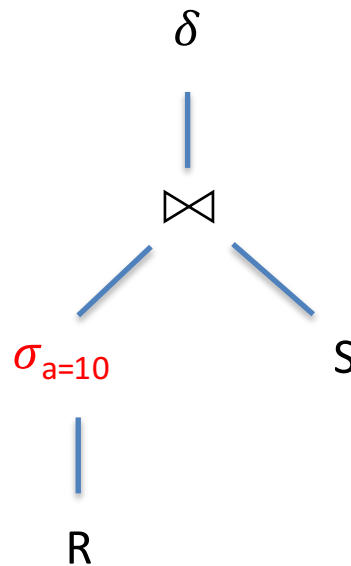
Would you transform the plan? If yes, how? 

Solution exercise 1

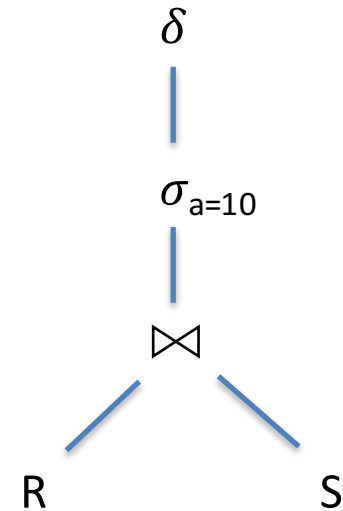
We surely push selection. And therefore we are left with two alternatives:
(a) and (b)



(a)



(b)



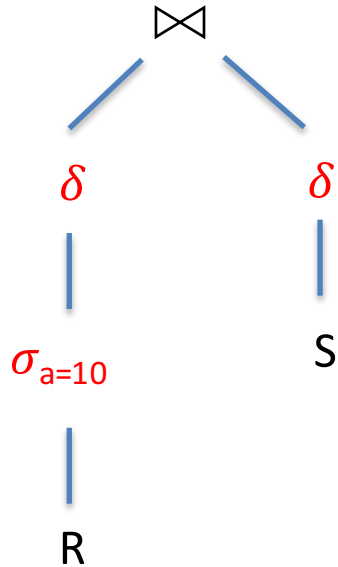
Solution exercise 1

Size of $\sigma_{a=10}(R)$: we divide $T(R)$ by $V(R,a) \rightarrow 5000/50 = 100$

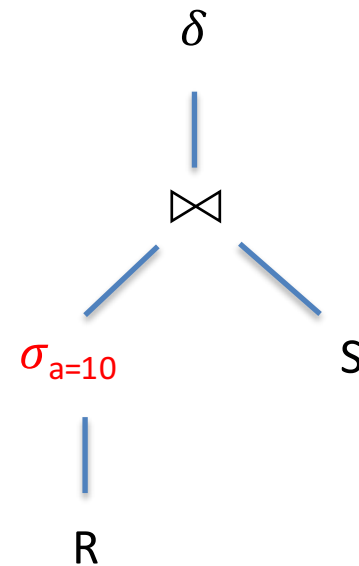
Size of join of tree (b): We multiple $T(R)/V(R,a)$ by $T(S)$, and divide by $\max\{V(R,b), V(S,b)\} \rightarrow 100 \times 2000 / 200 = 1000$

The first duplicate elimination in tree (a) \rightarrow half of the tuples in $\sigma_{a=10}(R) = 50$;

The second duplicate elimination in tree (a) \rightarrow half of the tuple of $S = 1000$.



(a)



(b)


Solution exercise 1

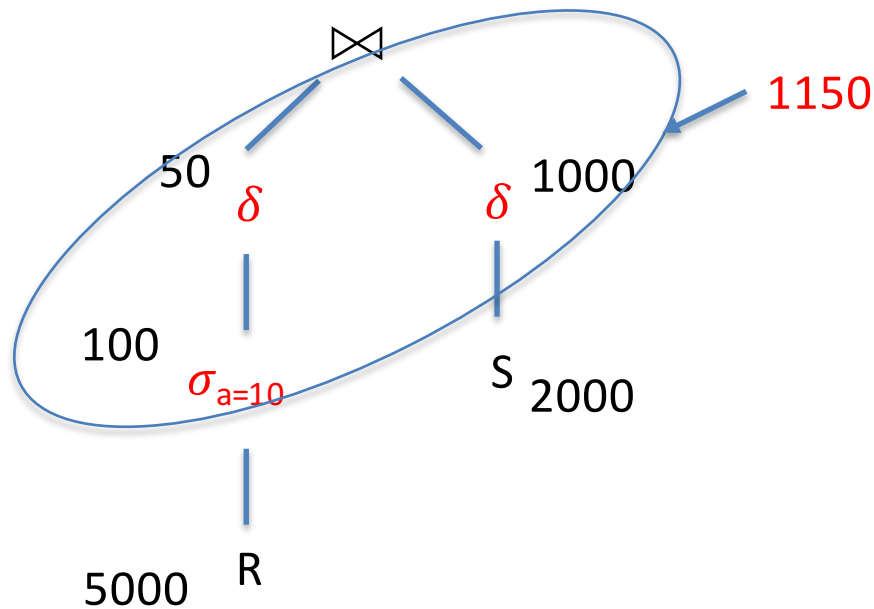
Size of $\sigma_{a=10}(R)$: we divide $T(R)$ by $V(R,a) \rightarrow 5000/50 = 100$

Size of join of tree (b): We multiple $T(R)/V(R,a)$ by $T(S)$, and divide by $\max\{V(R,b), V(S,b)\} \rightarrow 100 \times 2000 / 200 = 1000$

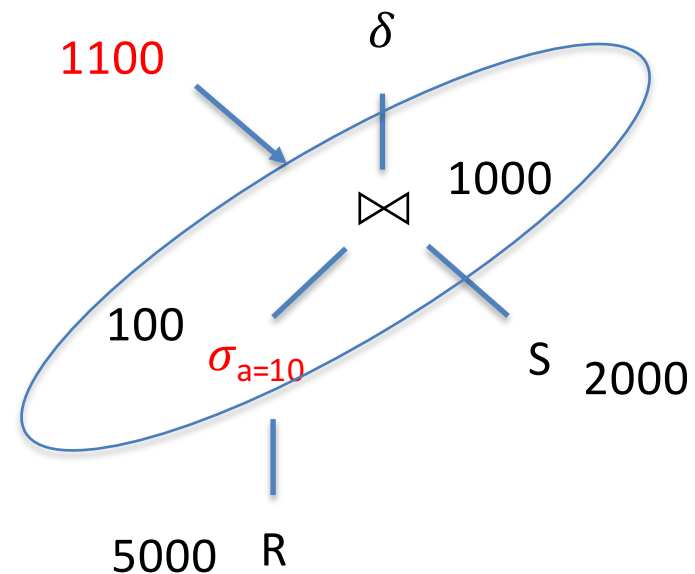
The first duplicate elimination in tree (a) \rightarrow half of the tuples in $\sigma_{a=10}(R) = 50$;

The second duplicate elimination in tree (a) \rightarrow half of the tuple of $S = 1000$.

We only consider the number of tuples in intermediate nodes (no root, no leaves)! 



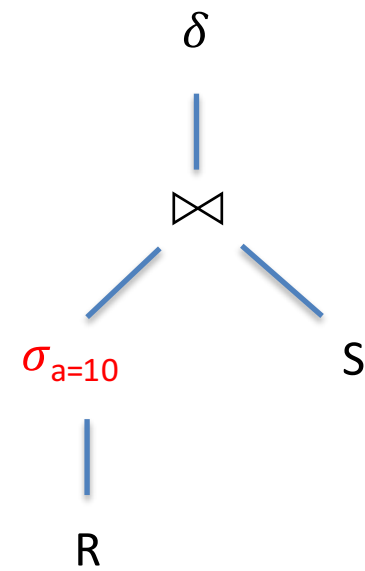
(a)



(b)

Solution exercise 1

This is the final logical query plan:




(b)

A note of the size estimates for projection

For the size estimate of a projection, we should remember that the number of tuples does not change with projection.

However, the size of each tuple may decrease, and therefore the number of pages needed to store the result of projection may decrease.

One way to take this into account is to measure the factor f by which the relation R shrinks with the projection, and consider the size of the projection to be $f \times T(R)$. 

Exercise 2

Consider the two relations: MovieStar(Name, Age, Gender, Birthdate),

StarsIn(Title, MovieYear, StarName)

written as

MovieStar(N, A, G, B),

StarsIn(T, Y, SN)

with statistics:

$T(\text{MovieStar}) = 250.000$

$T(\text{StarsIn}) = 2.500.000$

$V(\text{MovieStar}, N) = 250.000$

$V(\text{MovieStar}, B) = 10.000$

$V(\text{StarsIn}, SN) = 250.000$

and consider the query:


select T

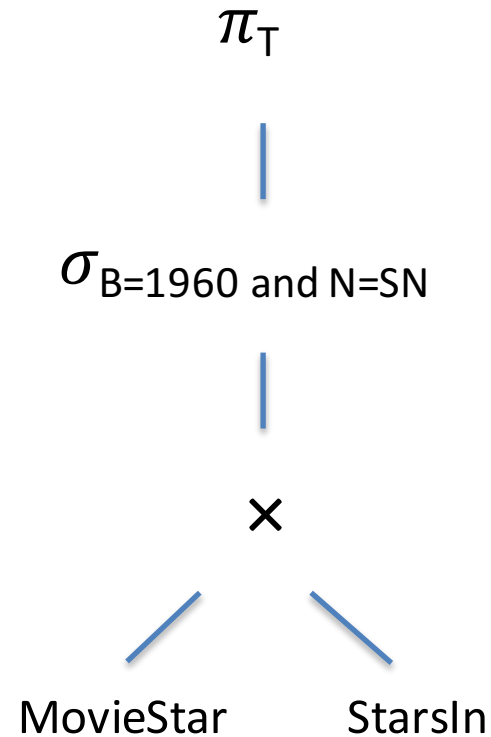
from MovieStar, StarsIn

where B = 1960 **and** N = SN

Tell which is the “best”
logical plan associated
to the query.


Solution exercise 2

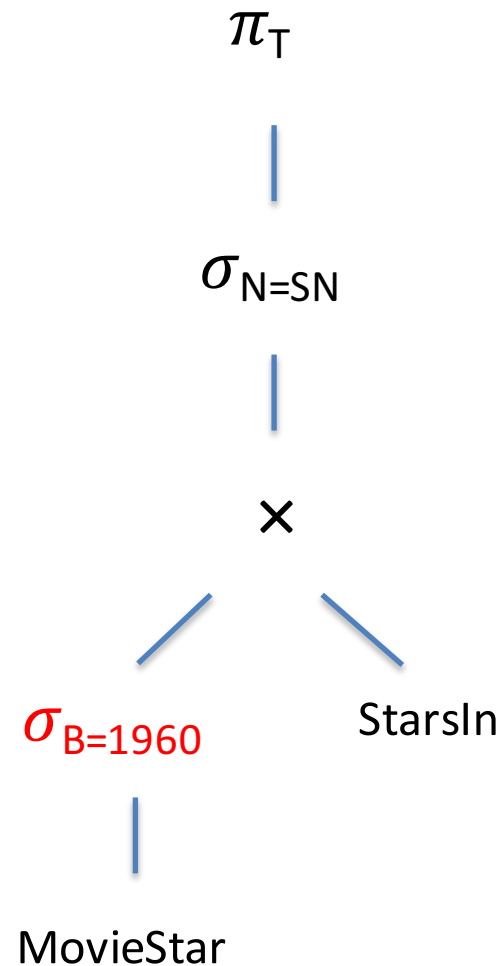
This is the logical plan
derived from the query
code: 



```
select T
from MovieStar, StarsIn
where B = 1960 and N = SN
```

Solution exercise 2

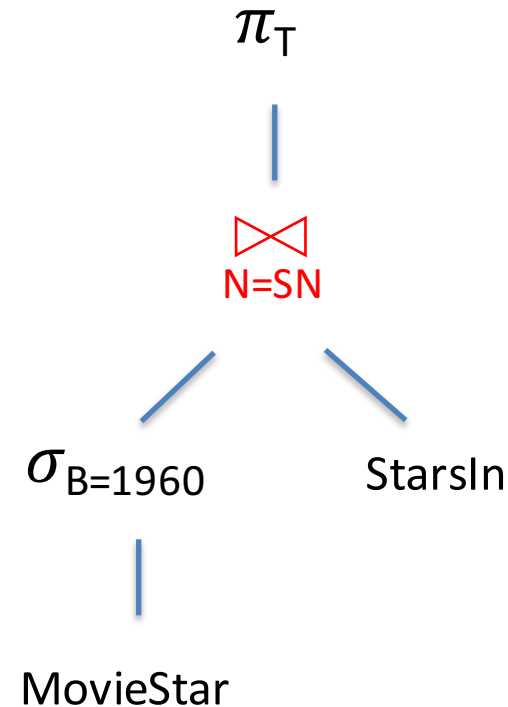
We obviously split and push selections. This is the logical plan obtained: 



`select T`
`from MovieStar, StarsIn`
`where B = 1960 and N = SN`

Solution exercise 2

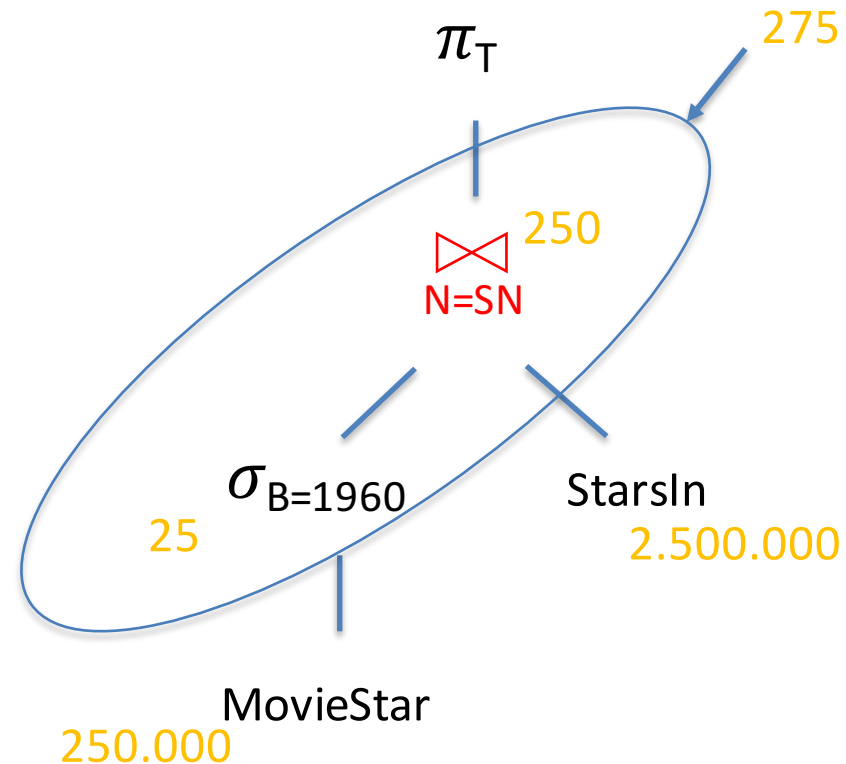
We transform product into equijoin. This is the logical plan obtained:



`select T`
`from MovieStar, StarsIn`
`where B = 1960 and N = SN`

Solution exercise 2


We transform product into equijoin. This is the logical plan obtained:

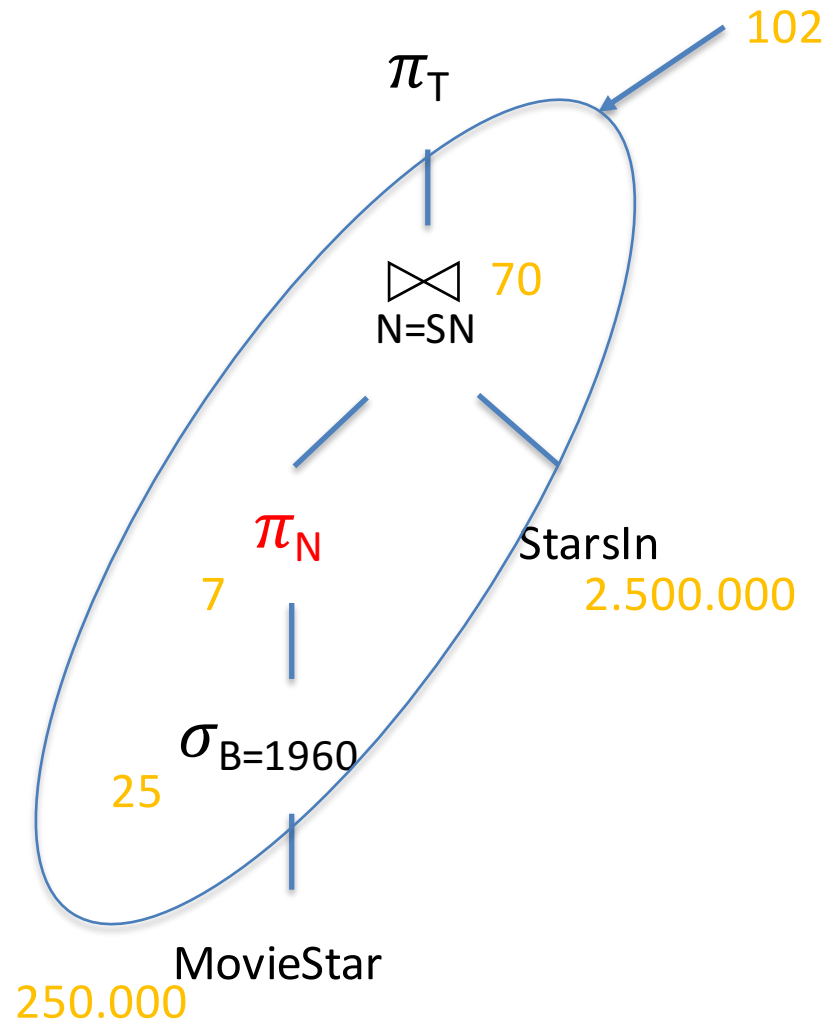


```
select T
from MovieStar, StarsIn
where B = 1960 and N = SN
```

Solution exercise 2

What if we add projection
on the first operand of join?


This is the logical
plan obtained: 

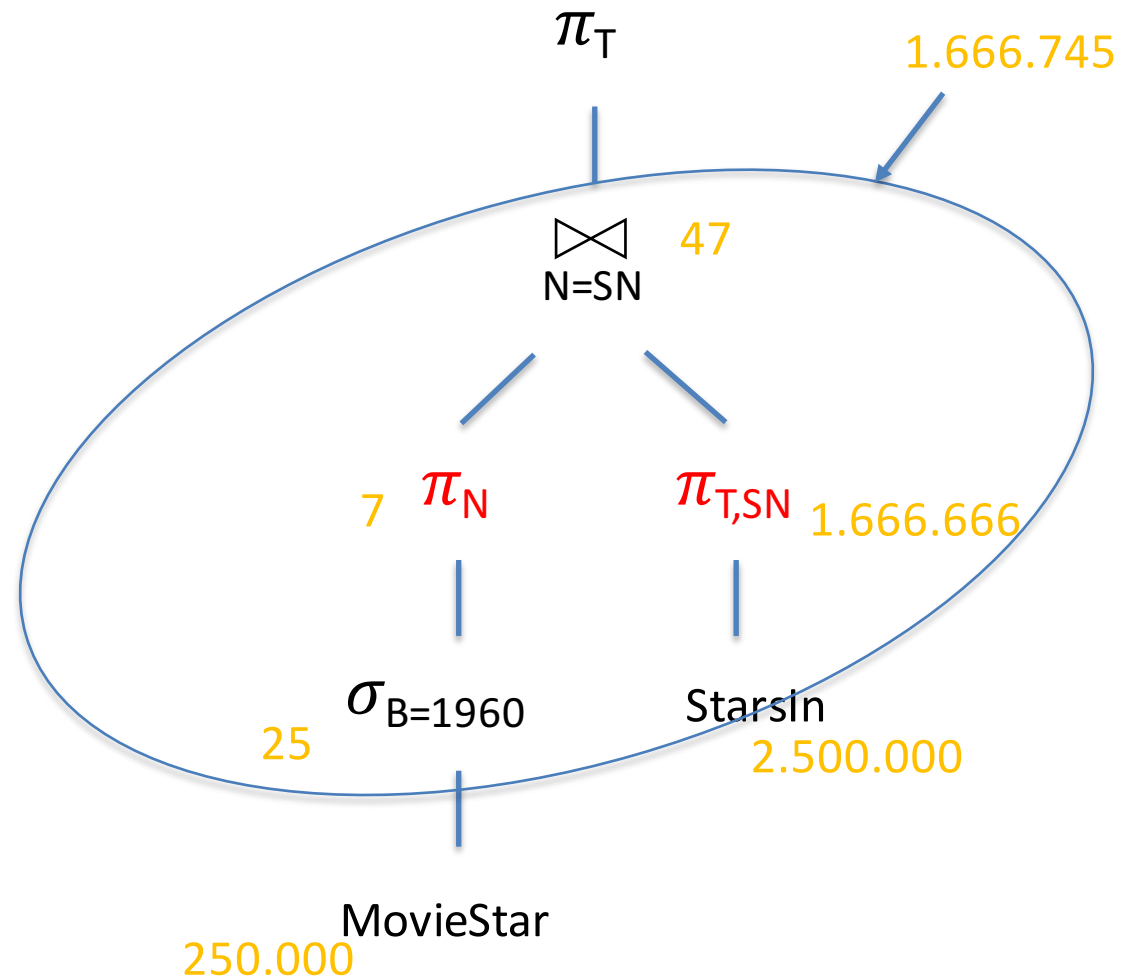


`select T`
`from MovieStar, StarsIn`
`where B = 1960 and N = SN`

Solution exercise 2

What if we add projection
on both operands of join?

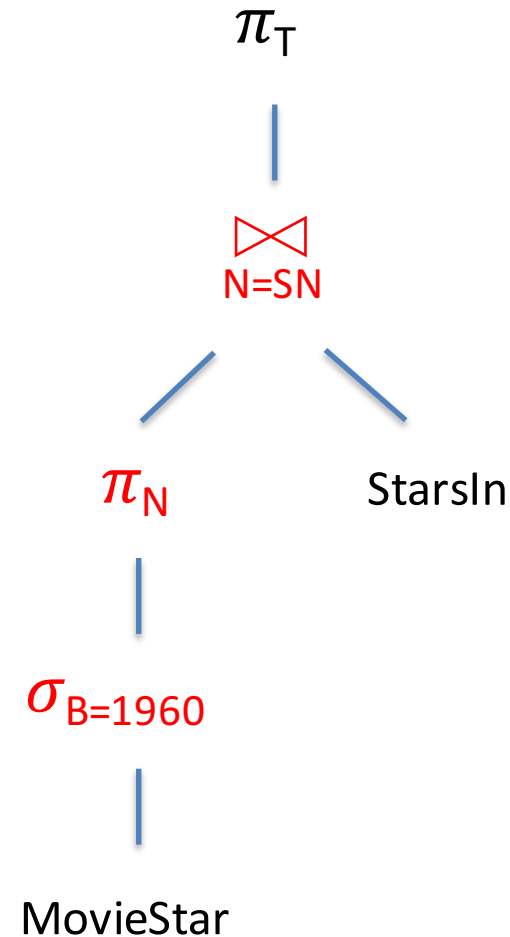
This is the logical
plan obtained: 



`select T`
`from MovieStar, StarsIn`
`where B = 1960 and N = SN`

Solution exercise 2

This is the final
logical plan: 💬





select T
from MovieStar, StarsIn
where B = 1960 and N = SN

Exercise 3

Let the relevant statistics associated to R1, R2, R3, and R4 be as follows:

R1(A,B,C),
R2(A,B,C),
R3(A,C,D,E,F,G),
R4(C,H,L)

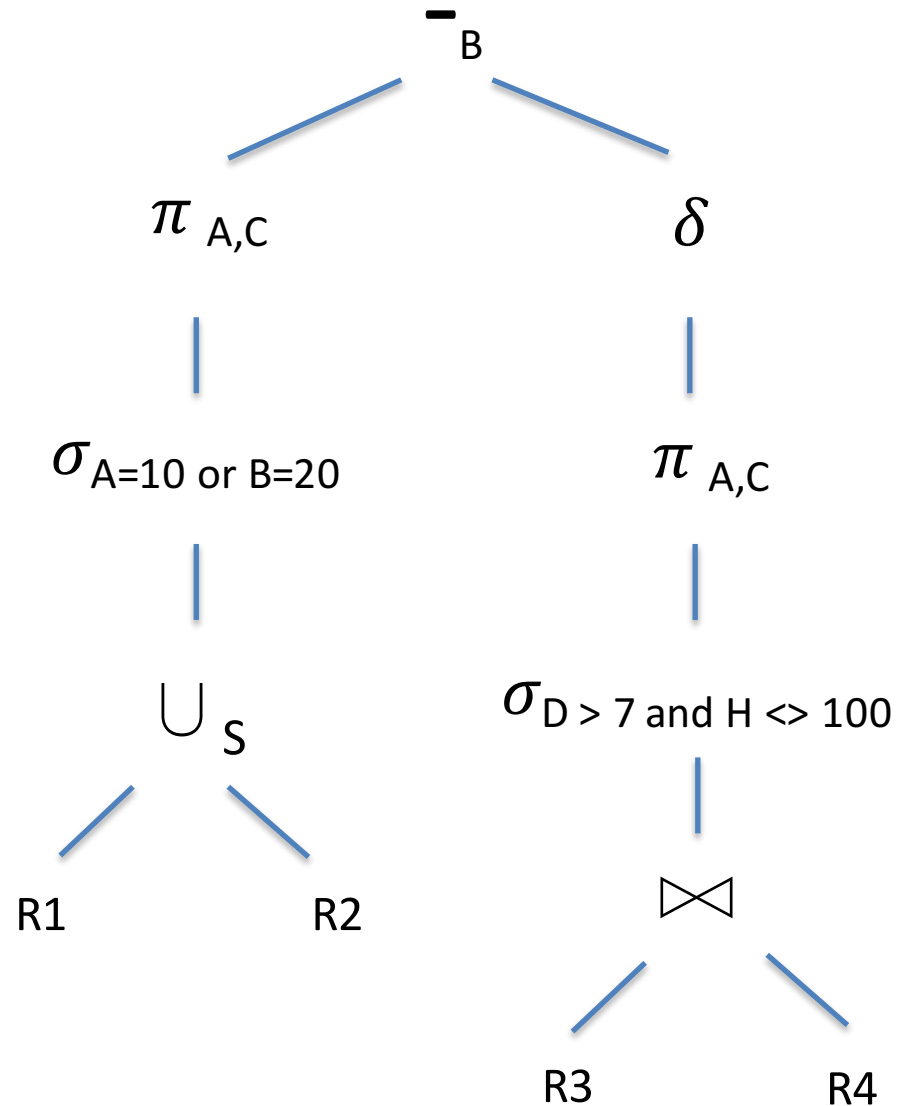
$T(R1) = 7000$	$V(R1,A) = 100$	$V(R1,B) = 200$
$T(R2) = 5000$	$V(R2,A) = 100$	$V(R2,B) = 200$
$T(R3) = 3000$	$V(R3,C) = 100$	
$T(R4) = 8000$	$V(R4,C) = 120$	

```
select T1.A, T1.C
from (select * from R1
       union
      select * from R2) T1
where T1.A = 10 or T1.B = 20
 minus all
select distinct A,C
from R3 natural join R4
where R3.D > 7 and R4.H <> 100
```

Tell which is the “best” logical plan associated to the query.

Solution exercise 3

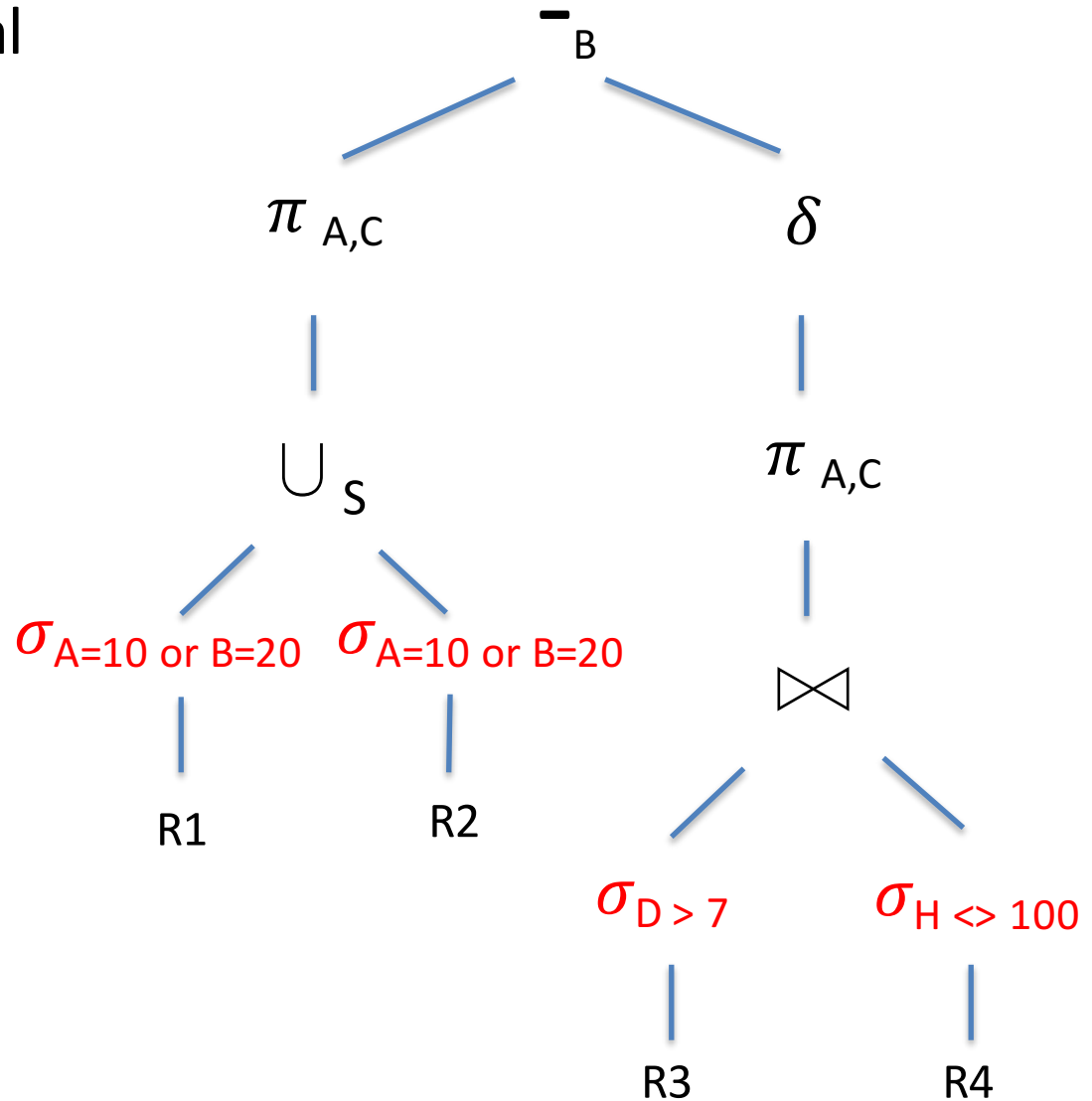
This is the logical plan derived
from the query code: 



R1(A,B,C),
R2(A,B,C),
R3(A,C,D,E,F,G),
R4(C,H,L)

Solution exercise 3

We obviously split and push selections. This is the logical plan obtained: 

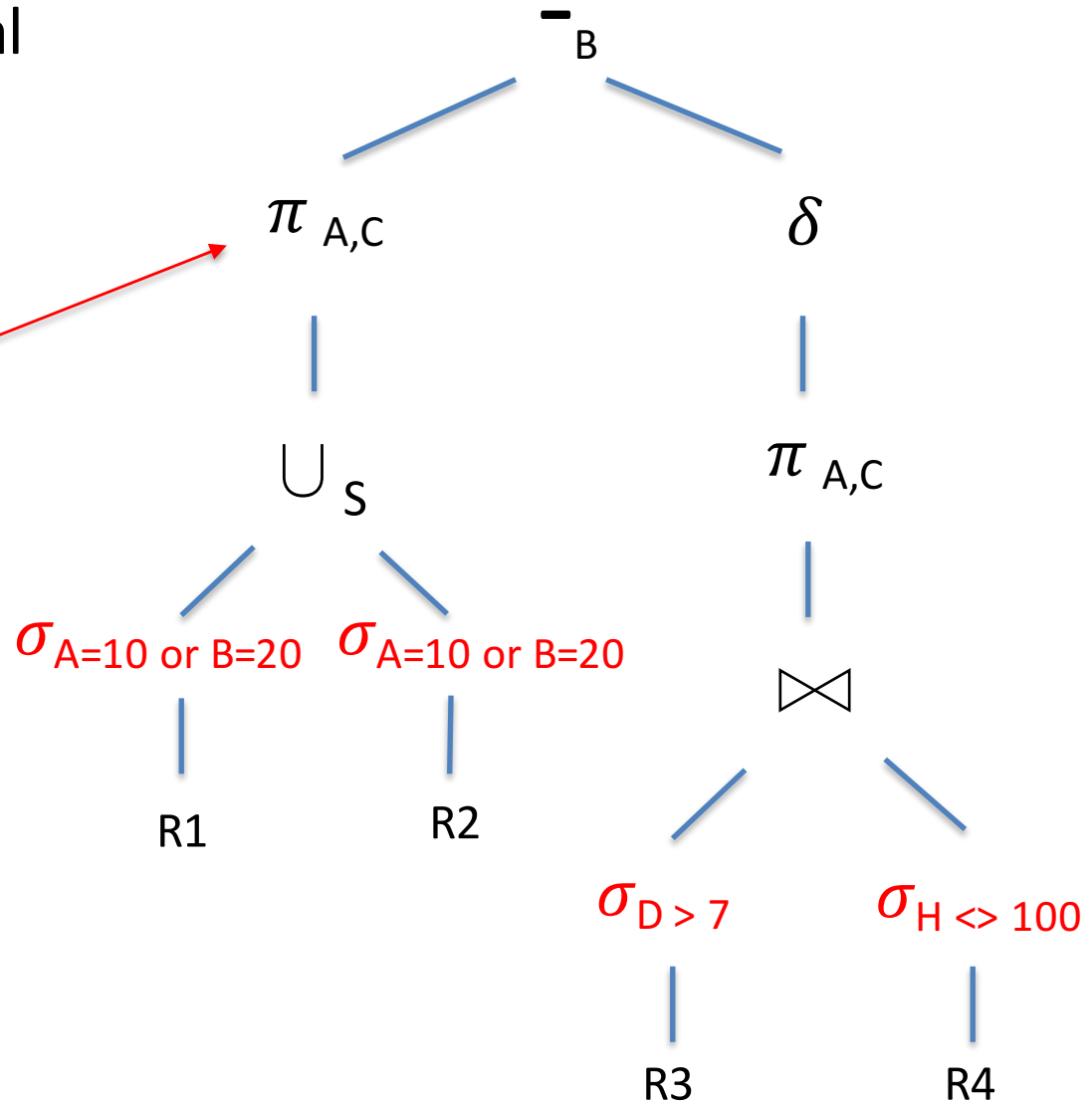


$R1(A,B,C),$
 $R2(A,B,C),$
 $R3(A,C,D,E,F,G),$
 $R4(C,H,L)$

Solution exercise 3

We obviously split and push selections. This is the logical plan obtained:

We cannot push this projection without changing the semantics of the query



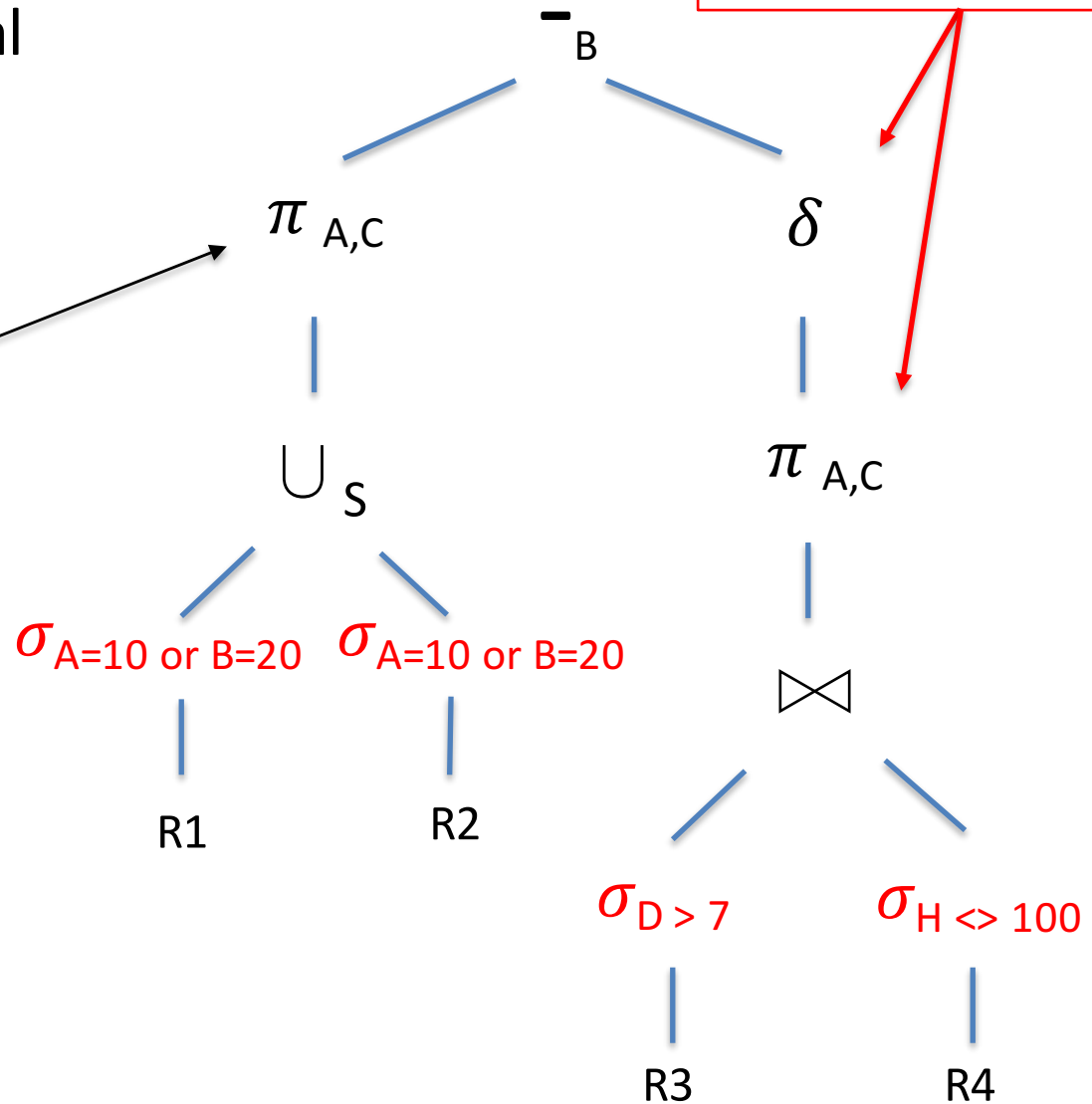
$R1(A,B,C),$
 $R2(A,B,C),$
 $R3(A,C,D,E,F,G),$
 $R4(C,H,L)$

Solution exercise 3

We obviously split and push selections. This is the logical plan obtained: 

We cannot push this projection without changing the semantics of the query

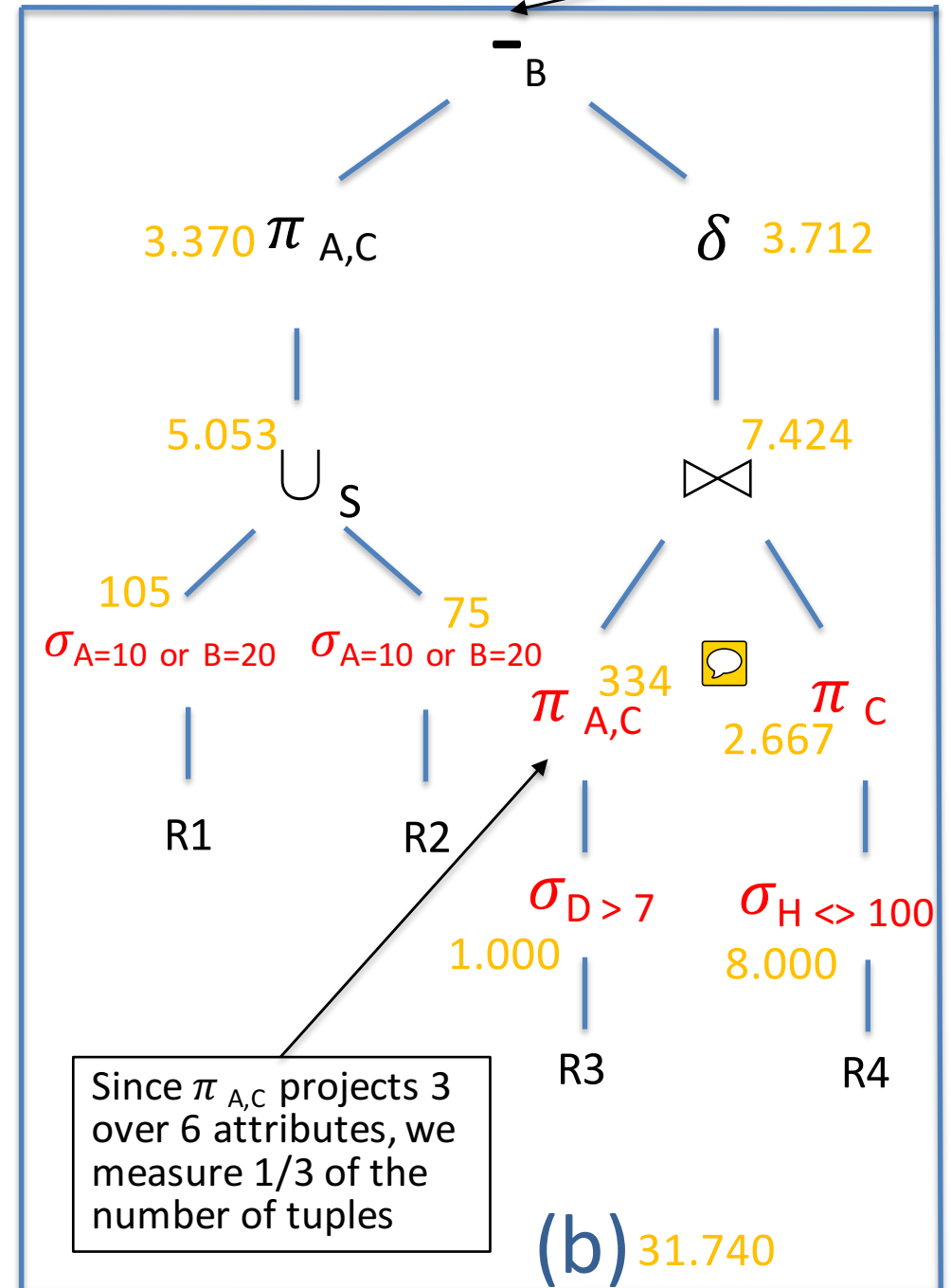
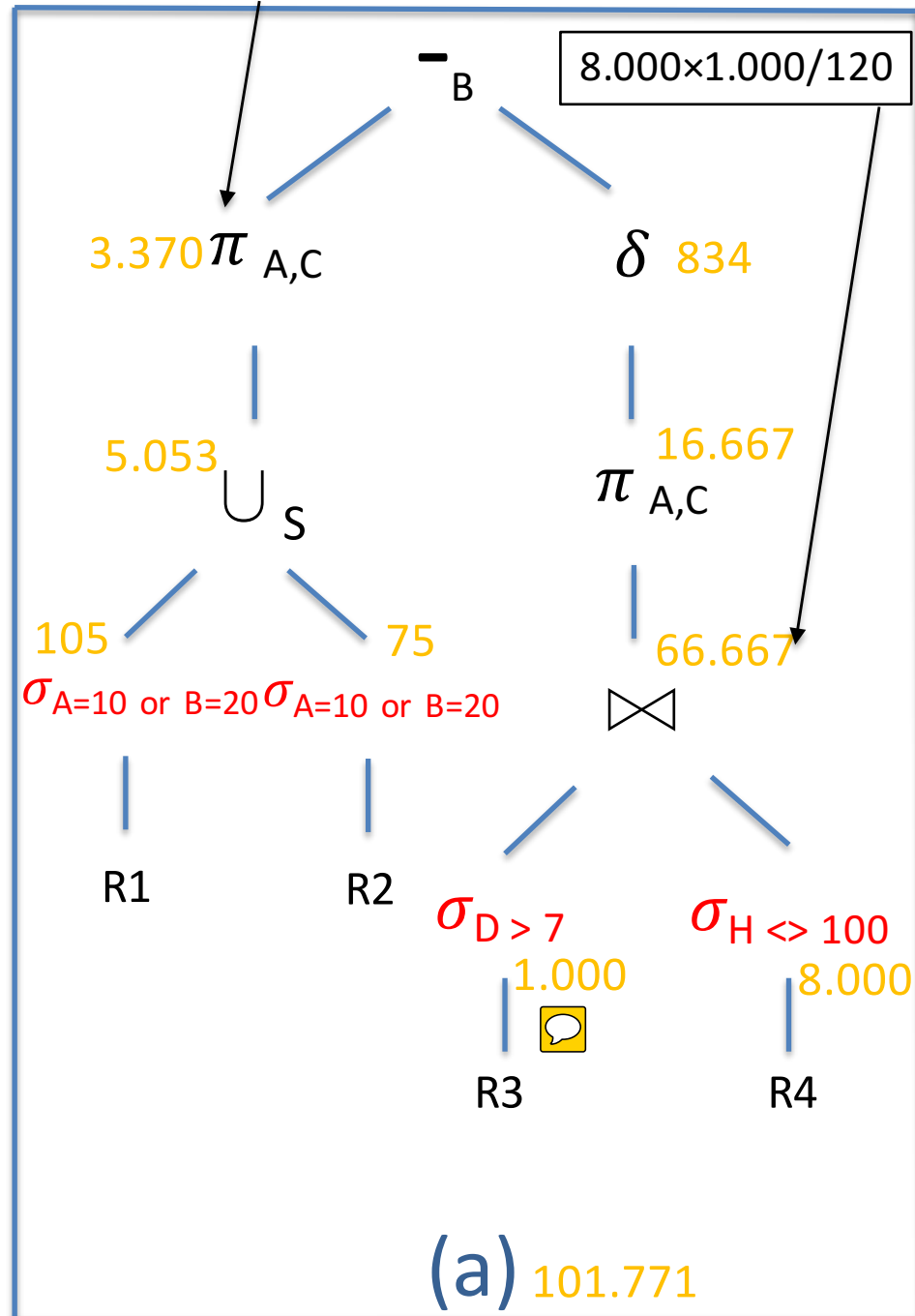
R1(A,B,C),
R2(A,B,C),
R3(A,C,D,E,F,G),
R4(C,H,L)



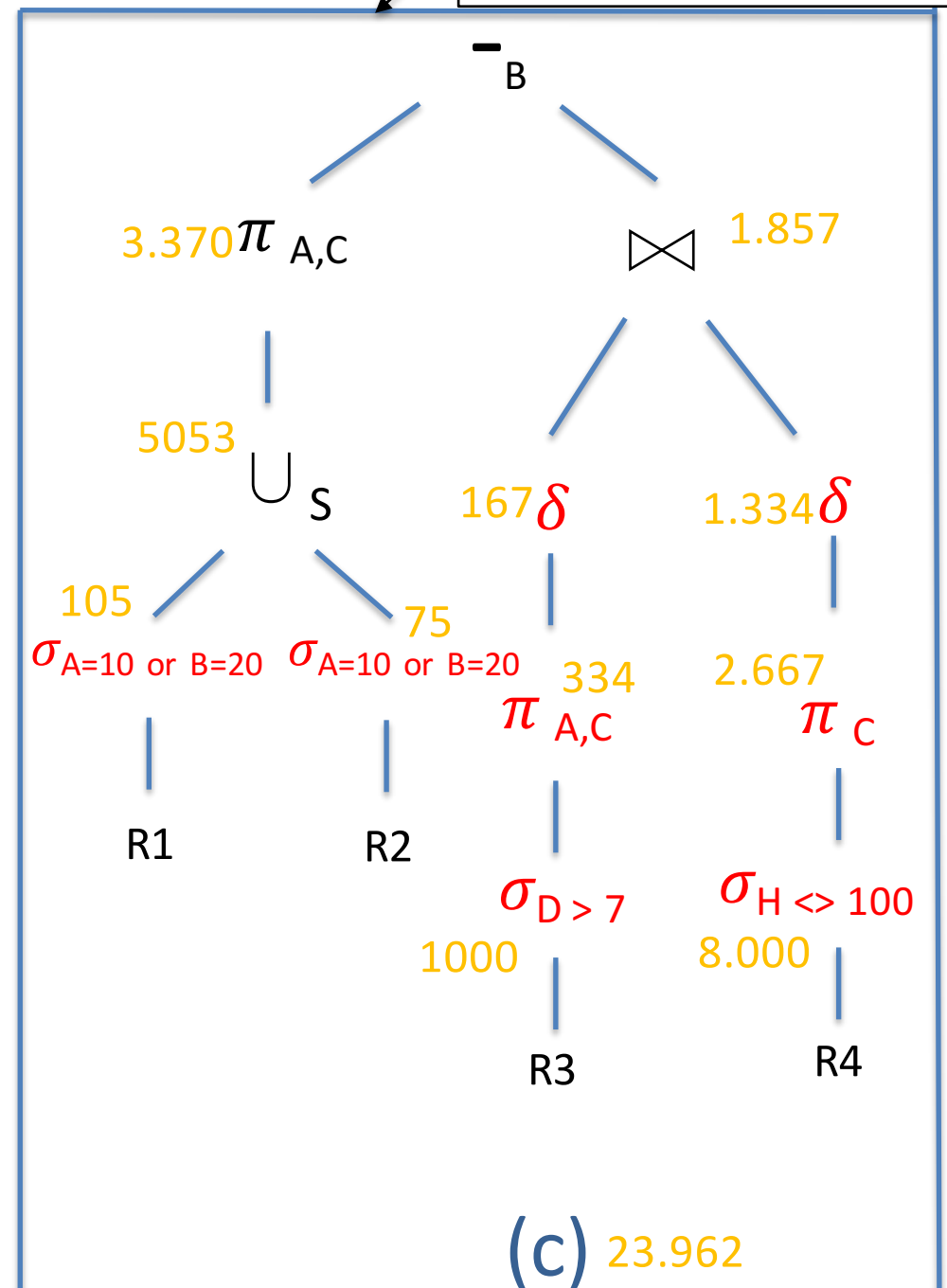
Since $\pi_{A,C}$ projects 2 over 3 attributes, we measure 2/3 of the number of tuples

Solution exercise 3

Logical plan obtained by pushing projection

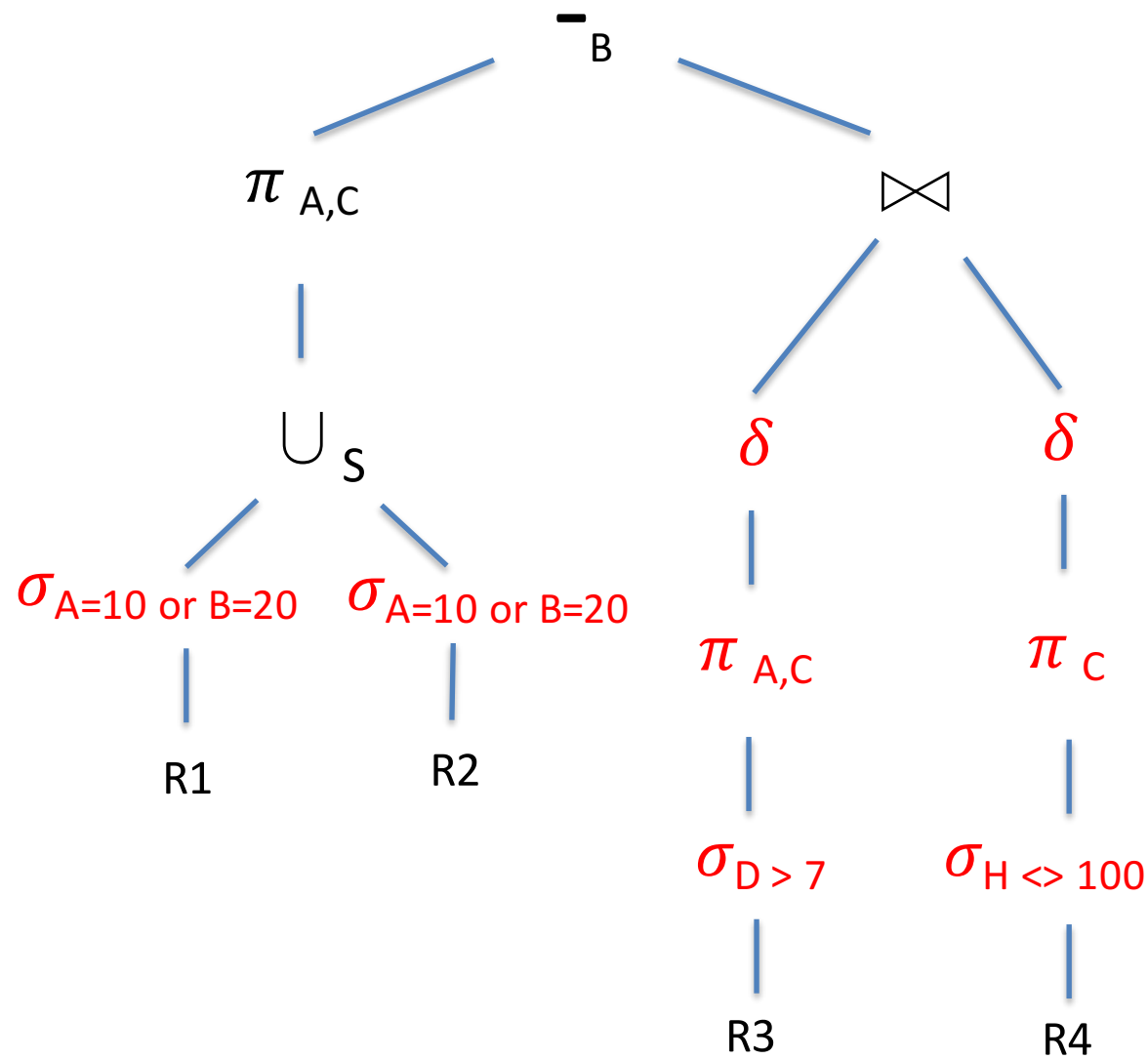


Logical plan obtained by pushing projection and duplicate elimination



Solution exercise 3

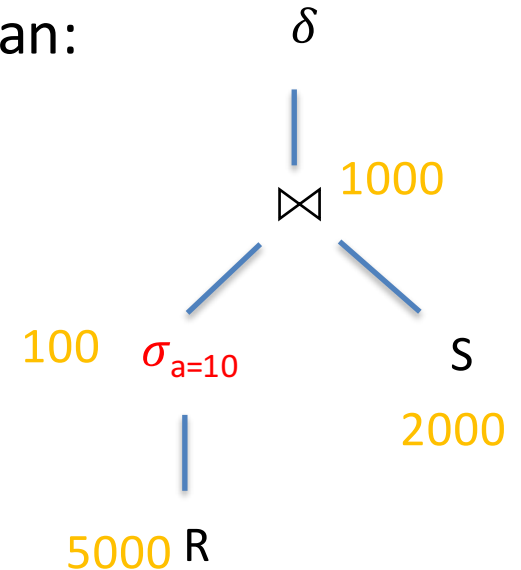
Final logical plan:



(c)

Exercise 4

Consider the logical query plan:



Also, let the statistics for $R(a,b)$ and $S(b,c)$ be as follows:

$$T(R) = 5000$$


$$T(S) = 2000$$

$$V(R,a) = 50$$

$$V(S,b) = 200$$

$$V(R,b) = 100$$

$$V(S,c) = 100$$

We know 10 tuples of both R and S fit in one page, and the buffer has 12 free frames. 

Which physical query plan would you choose, and which is the cost of executing such plan in terms of number of page accesses?

Solution exercise 4

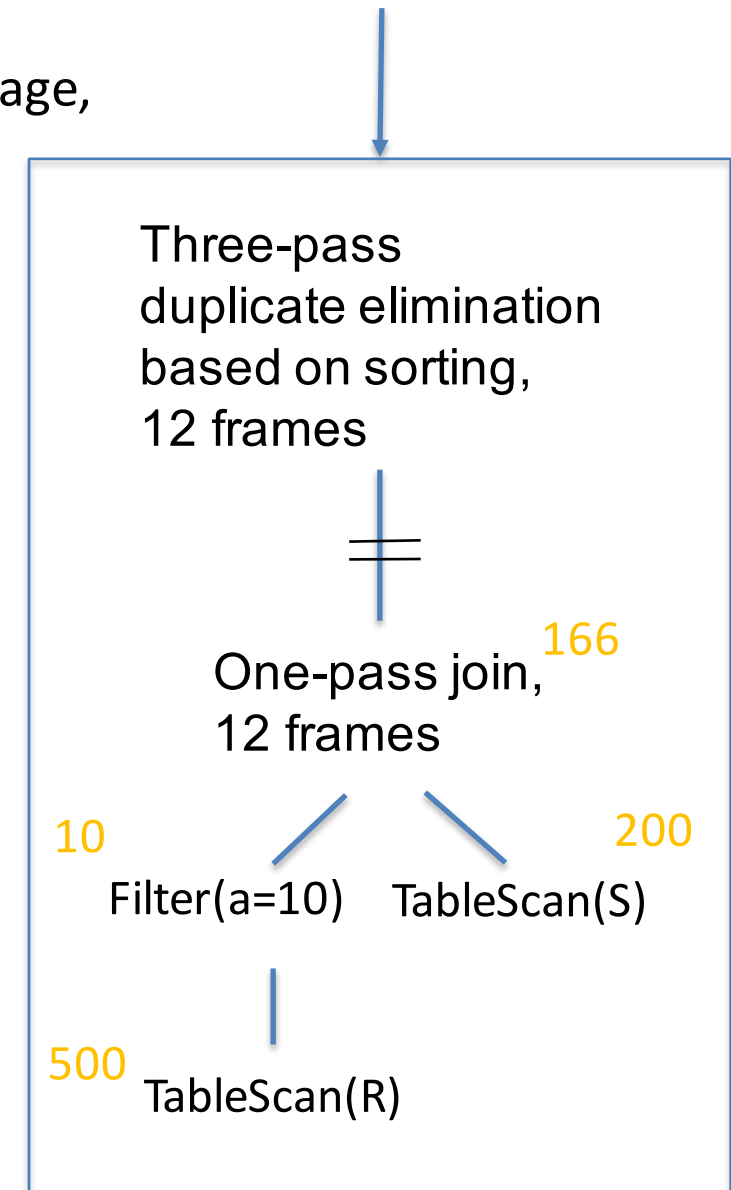
We know 10 tuples of both R and S fit in one page, and the buffer has 12 free frames. So, the pages after the selection are 10, and we can compute the join in one pass. If 10 tuples of 2 values fit in one page, 6 tuples of 3 values fit in one page. It follows that the $100 \times 2000 / 200$ tuples of the result of the join must be materialized in $1000 / 6 = 166$ pages. Unfortunately, we cannot do duplicate elimination in one or two passes. We need three passes.

Cost:

$$500 + 200 + 1000/6 + 5 \times 1000/6 =$$
$$500 + 200 + 166 + 830 = 1.696$$

TableScan(R) TableScan(S) writing of join three-pass duplicate elimination

Physical query plan
(with number of pages)



Exercise 5

Consider the logical query plan:

`select T`
`from MovieStar, StarsIn`
`where B = 1960 and N = SN`

MovieStar(N, A, G, B),
StarsIn(T, Y, SN)

$T(\text{MovieStar}) = 250.000$

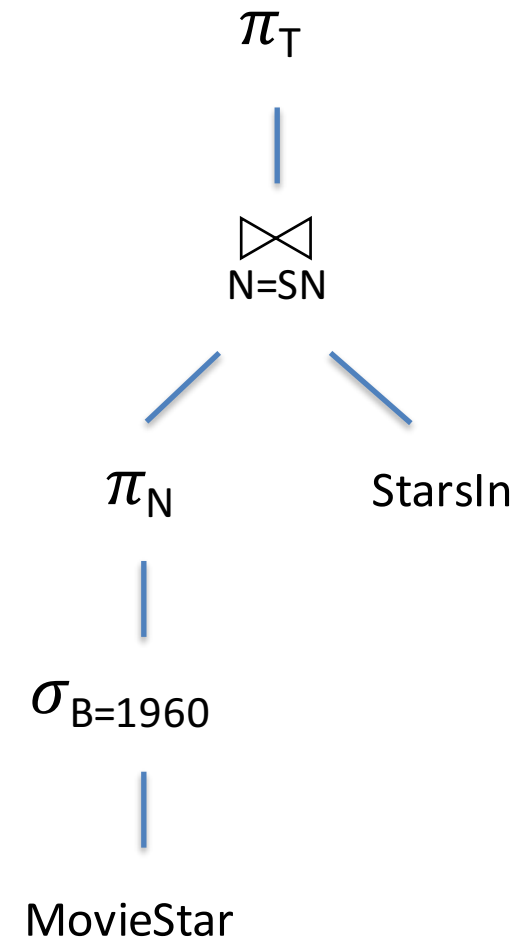
$T(\text{StarsIn}) = 2.500.000$

$V(\text{MovieStar}, N) = 250.000$

$V(\text{MovieStar}, B) = 10.000$

$V(\text{StarsIn}, SN) = 250.000$

We know that 100 tuples of both R and S fit in one page,
and the buffer has 12 free frames.

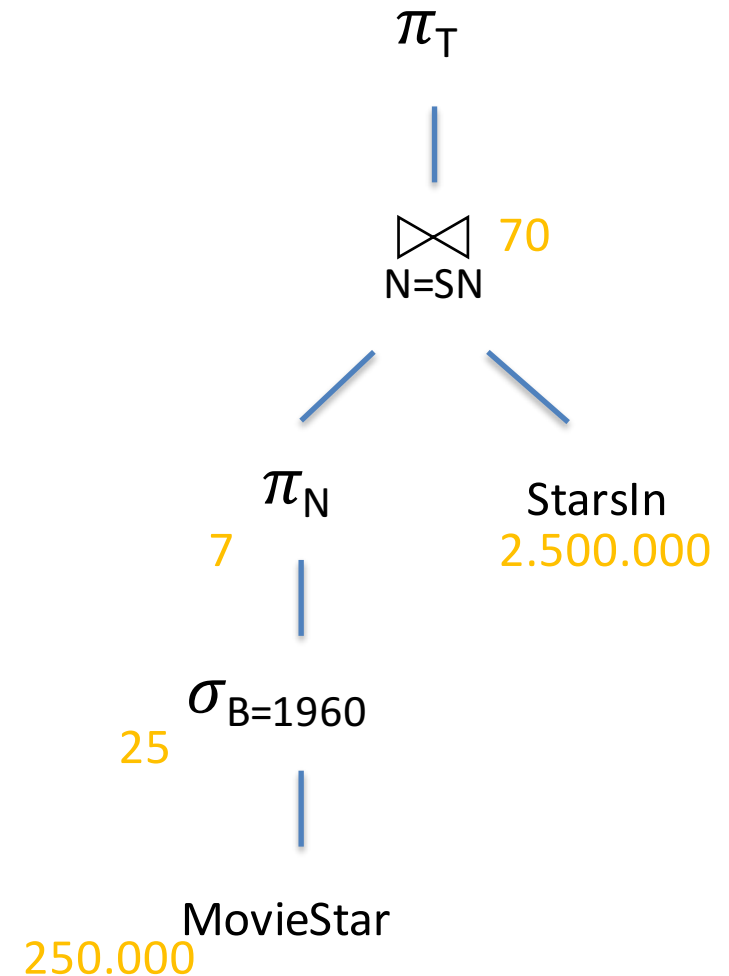


Which physical query plan would you choose, and which is the cost of executing such plan in terms of number of page accesses?

Solution exercise 5

Consider the logical query plan where we show the number of tuples for each node:

`select T`
`from MovieStar, StarsIn`
`where B = 1960 and N = SN`

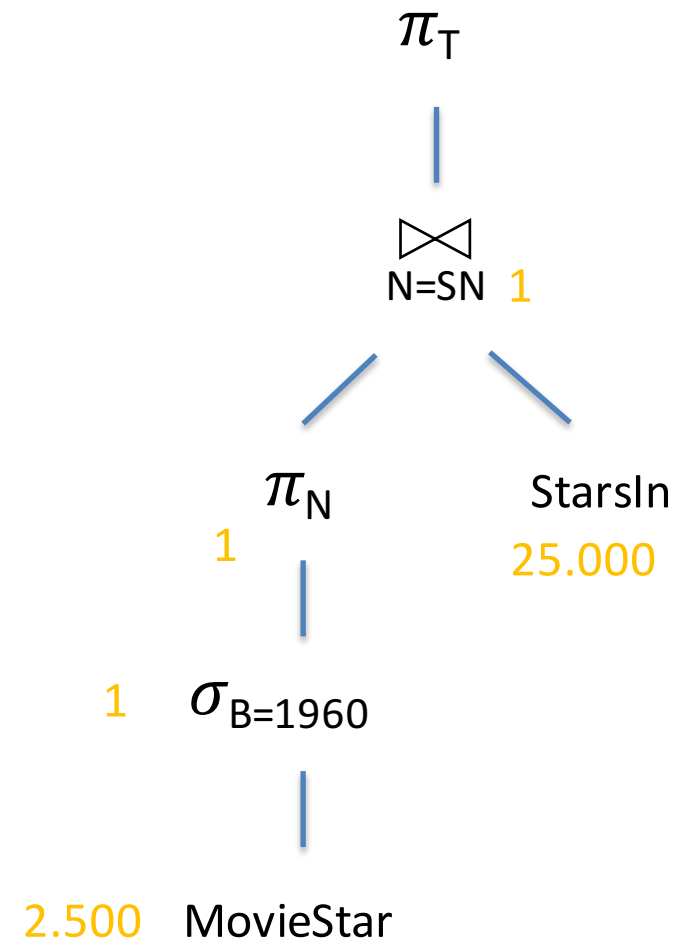


We know that 100 tuples of both R and S fit in one page, and the buffer has 12 free frames.

Solution exercise 5

Consider the logical query plan
where now we indicate the number
of pages associated to the various nodes:

select T
from MovieStar, StarsIn
where B = 1960 and N = SN

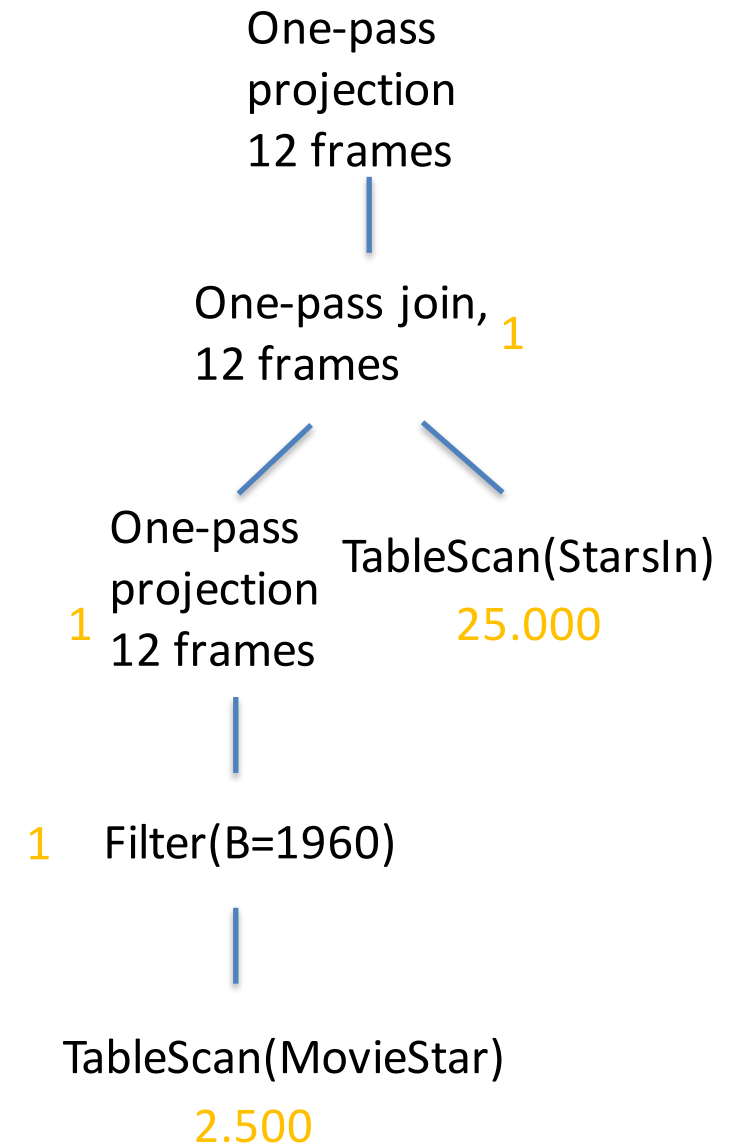


Solution exercise 5

Here is the physical query plan:

`select T`
`from MovieStar, StarsIn`
`where B = 1960 and N = SN`

Cost
 $2.500 + 25.000 = 27.500$



Exercise 6

Consider the relations

Match(Team1,Team2,Date,Result,Referee,Attendees),

Rivalry(FirstTeam,SecondTeam,Degree,Official,Timestamp)

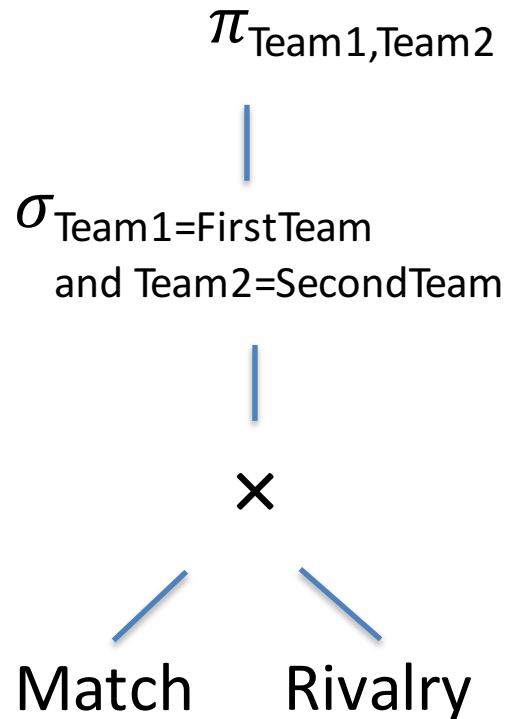
where Match occupies 2.000 pages, and Rivalry occupies 16.000 pages.
We know that we have an unclustering tree index on Match with search key (Team1,Team2,Date), and we have 100 free frames in the buffer.
Consider the query:

```
select Team1, Team2, Date, Degree
from Match, Rivalry
where Team1 = FirstTeam
      and Team2 = SecondTeam
```

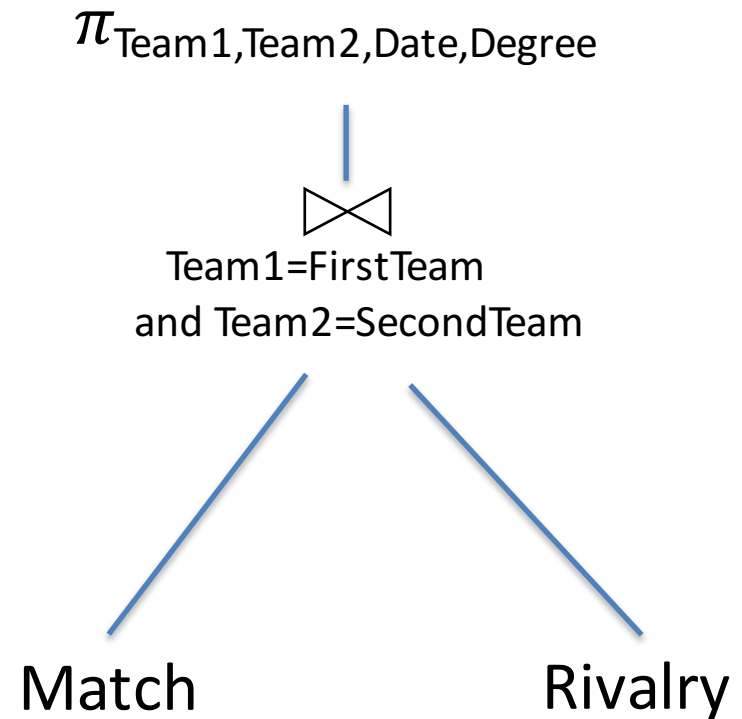
Tell which physical plan
would you associate
to the query, and which
is its cost

Solution exercise 6

Here is the logical query plan associated to the query code:



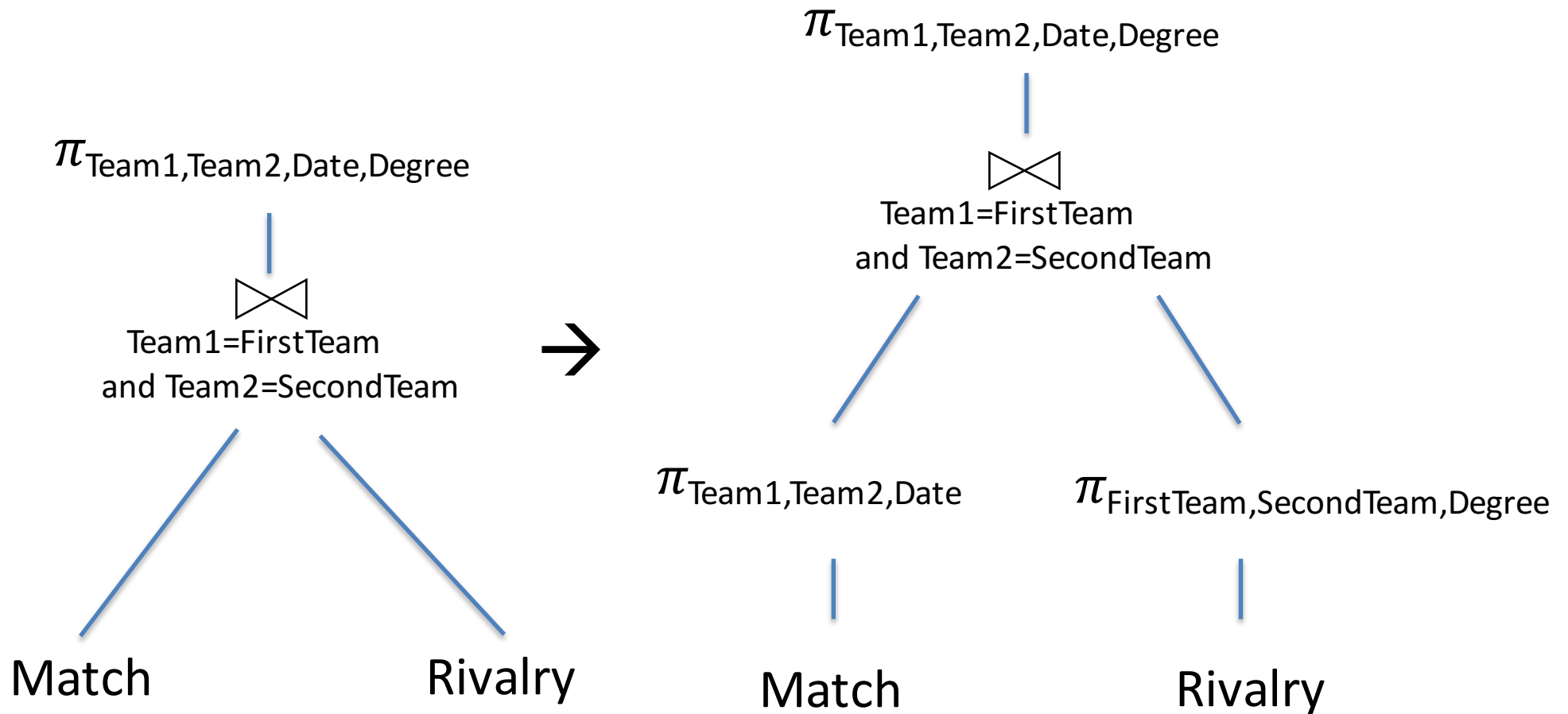
Transformed immediately as follows:



Notice that the size of every data entry is a half of a tuple, and therefore the tree index on Match occupies 1.500 pages (taking into account the 66% rule). Since Rivalry occupies 16.000 pages, we cannot use the one-pass algorithm for the join, even if we opt for index-only scan for Match. Nor we can use the two-pass join based on sorting, because $1.500 + 16.000 > 99 \times 100$ (i.e., 9.900).

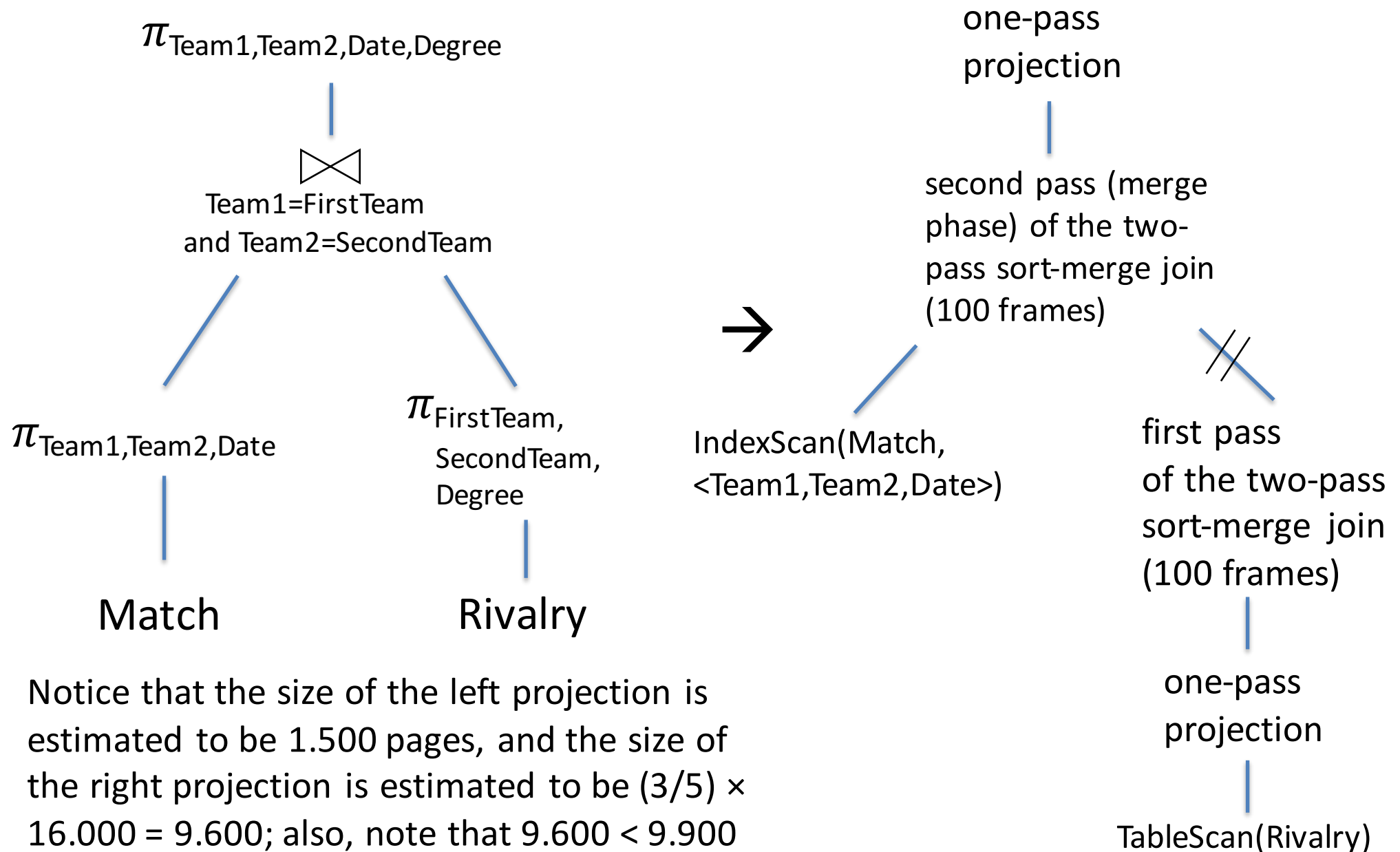
Solution exercise 6

Let us try to push projection:



Solution exercise 6

Let us now concentrate on the physical query plan.



Solution exercise 6

The total cost is:

$$16.000 + 9.600 + 9.600 + 1.500 =$$

36.700 page accesses

reads 97 sorted files (1.500 + 9.600 pages) using 97 buffer frames, merge them, using one buffer frames for computing the final projection

1.500 pages

IndexScan(Match,
<Team1,Team2,Date>)

reads 16.000 pages, projects the tuples while reading, and writes the $9.600/100 = 96$ sorted sublists of 100 pages each

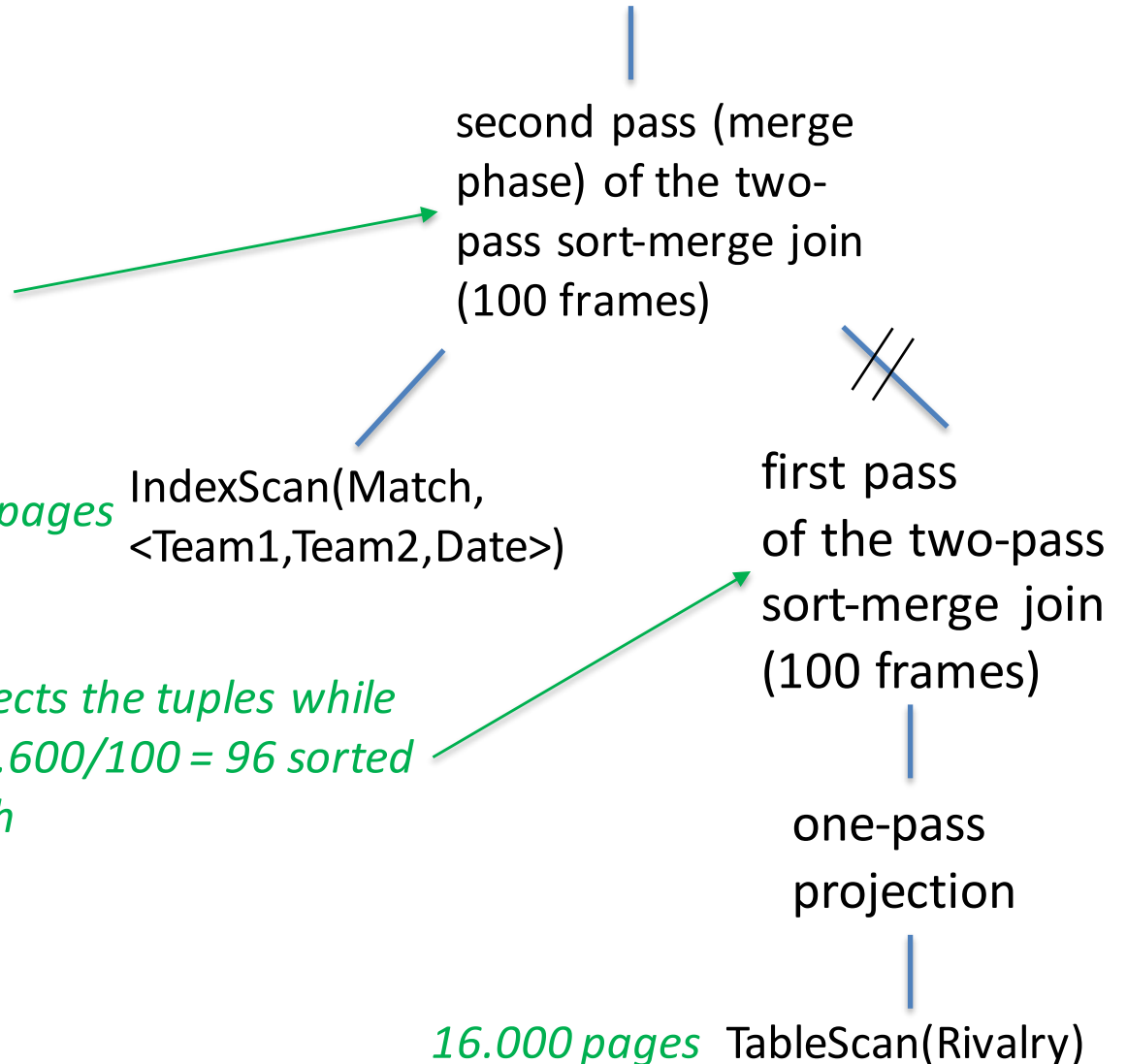
one-pass
projection

second pass (merge
phase) of the two-
pass sort-merge join
(100 frames)

~~first pass
of the two-pass
sort-merge join
(100 frames)~~

one-pass
projection

16.000 pages TableScan(Rivalry)



Exercise 7

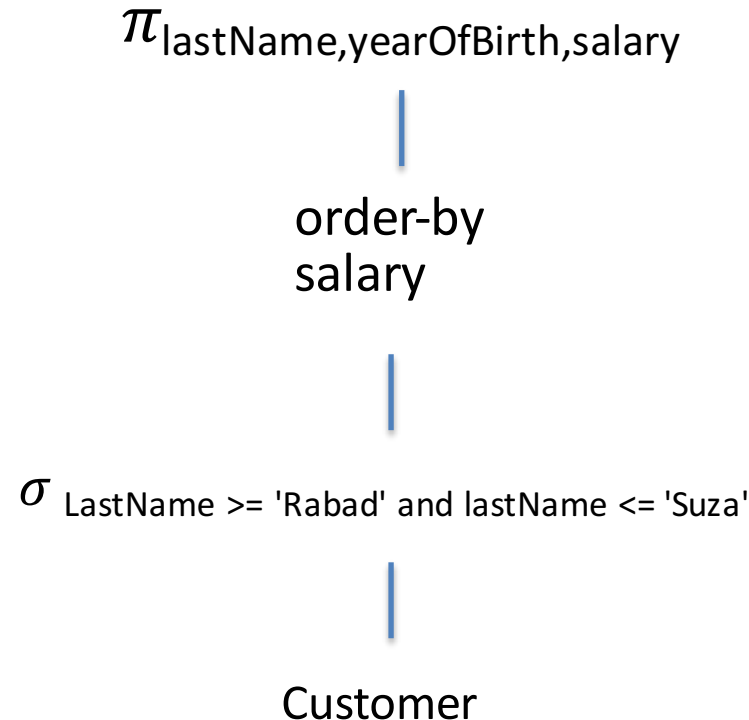
Assume that relation **Customer(firstName,lastName,yearOfBirth,salary)** (where lastName is a key), has 650.000 tuples, each attribute and each pointer in the system occupies 100 Bytes, each page has space for 4.000 Bytes, the last names of customers are equally distributed on the first letter over the 26 letters of the alphabet, and the most frequent query on Customer asks for all customers whose last name falls into a given range.

1. Tell which method would you use to store the relation.
2. Tell which logical plan and physical plan would you use to answer the following query Q

```
select lastName, yearOfBirth, salary
from Customer
where lastName >= 'Rabad' and lastName <= 'Suza'
order by salary
```
3. Assuming that 65 free buffer frames are available, tell which is the cost of executing the physical query plan you have defined for item 2, in terms of the number of page accesses.

Solution exercise 7

The logical plan associated to the query code is as follows:



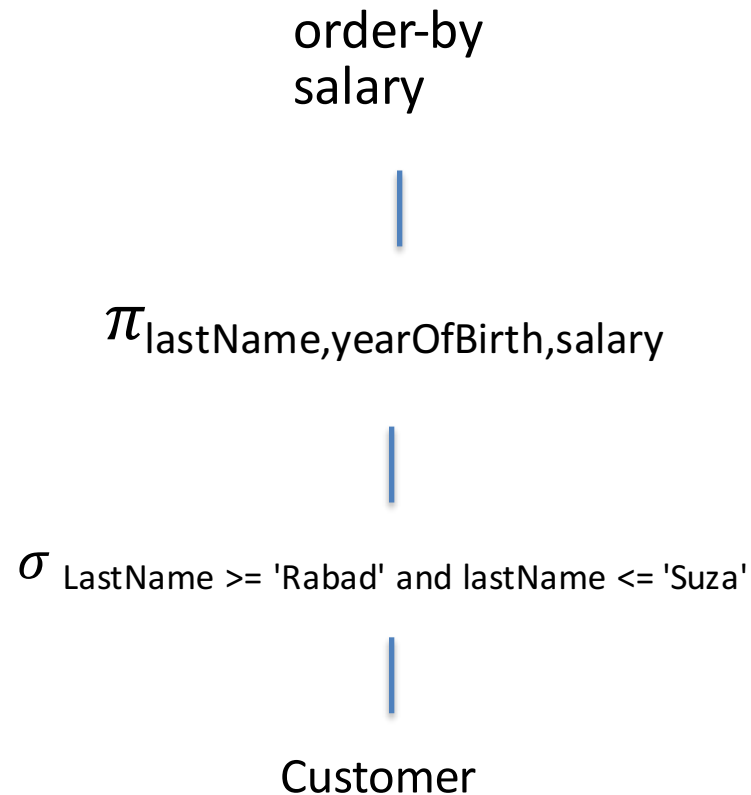
Since the most frequent query on Customer asks for all customers whose last name falls into a given range, we focus our attention on the method that stores the relation in a sorted file with sorting key lastName, with an associated clustering, sparse tree-based index on search key lastName. Indeed, it is well known that range queries are well supported by a clustering B+-tree index.

Solution exercise 7

- Having the clustering B+-tree index, we can use the index for the “selection” operator, thus finding the first page of Customer with the tuples satisfying the where condition. We then exploit the fact that the index is clustering, and sort on salary all pages with tuples satisfying the condition, so as to get the final result. Note that we do not materialize the result of the “selection” operator. Rather, a pipeline approach is used to pass the result of the “selection” operator to the sorting operator.
- Note that it seems promising to push projection under the order by, because the size of the operand will decrease.
- Note also that we cannot push projection under selection, because otherwise we cannot use the index.

Solution exercise 7

The logical plan of query Q
is now as follows:



Let us now choose the physical
query plan.

Solution exercise 7

- **Algorithm for selection.** Since each value or pointer occupies 100 Bytes, every tuple has 4 values, and since each page has space for 4000 Byte, it follows that every tuple occupies 400 Bytes, which means that each page has space for 10 tuples of Customer, and the relation is stored in 65.000 pages. Also, we have that each data entry requires 200 Bytes, and each page has space for 20 data entries. Taking into account the 67% occupancy rule for the leaves of the tree-based index, we have that each leaf contains 13 data entries. Also, since each page has space for 20 index entries, we can assume the value 15 for the fan-out. Note that, since the index is sparse, it stores one value of lastName for each page of Customer, and therefore the number of leaves are $65.000 / 13 = 5.000$. It follows that reaching the right leaf requires $\log_{15} 5.000 = 4$ page accesses.
- **Projection.** After reaching the right leaf, we then have to compute the projection of three over 4 attributes. Since we have to access the tuples with last names starting with two letters ('R' and 'S'), and we know that the last names of customers are equally distributed on the first letter over the 26 letters of the alphabet, the number of qualifying records are $(650.000/26) \times 2 = 50.000$, stored in $50.000/10 + 1 = 5.001$ pages. Since we choose $\frac{3}{4}$ of attributes, we count $(3/4) \times 5.001 = 3.751$ pages to pass to the sorting operator.
- **Sorting.** The records of such pages must be sorted having 65 buffer frames available. Since $3.751 < 65 \times 64$ (i.e., 4.160), we can use the two-pass sorting algorithm, with a cost of $3 \times 3.751 = 11.253$ page accesses.

Solution exercise 7

Here is the very simple physical query plan:

Two-pass
sort(salary)
65 frames

Projection in
one-pass

IndexScan(Customer, lastName >= 'Rabad' and lastName <= 'Suza')

Notice that if we had not pushed projection under order-by, we could have not used a two-pass algorithm for sorting, because $5.001 > 4160$.

Exercise 8

Assume we have the relation

WORK(employeeNo,lastName,firstName,birthYear,company,salary)

with 1.000.000 tuples, where the size of each attribute value is 10 Bytes, the size of each page is 600 Bytes, and the number of different companies is no more than 900. Suppose we have 350 free buffer frames available.

- Consider the query

select max(salary) from WORK group by company

computing, for each company, the maximum salary given to its employees. Describe in detail the algorithm you would use to compute the answer to the query and the cost of the execution of such algorithm in terms of number of page accesses.

- Consider the query

select * from WORK order by employeeNo

and describe in detail the algorithm you would use to compute the answer to the query and the cost of the execution of such algorithm in terms of number of page accesses.

Solution exercise 8

- The size of each tuple of relation WORK is 60 Bytes, and the size of each page is 600 Bytes. Therefore 10 tuples fit in one page, and WORK has 100.000 pages.
- Since 900 values of company are present on WORK, and we have to form one group for each value of company, and also, for each group that we encounter, we have to keep in the buffer the value of the corresponding company and the value of the current maximum value of salary for that group, we need 18.000 Bytes in the buffer, corresponding to $18.000/600 = 30$ pages, much less than the available 350. So, we can answer the query by a one pass algorithm that reads the pages of WORK, and for each tuple update the information about the current tuple. At the end, it writes the pages to the output. The cost is 100.000 (ignoring the cost of writing the output).
- We have to sort the relation WORK, and since $100.000 < 350 * 350$, we can sort it in two passes. Therefore, the cost is $3 * 100.000 = 300.000$ (ignoring the cost of writing the output).

Exercise 9

Assume we have the relations

P1(cno, lastName, firstName, age, dateBirth, cityBirth, region)

P2(cno, lastName, firstName, age, dateBirth, cityBirth, region)

storing information about customers of two Internet providers.

Each relation has 50.000.000 tuples, and each page contains 100 tuples. There are 20 processors, and both relations are uniformly distributed in such a way that the tuples of each of the 20 existing regions is stored in a specific processor.

We have to answer the following query:

select *

from (select * from P1

 union

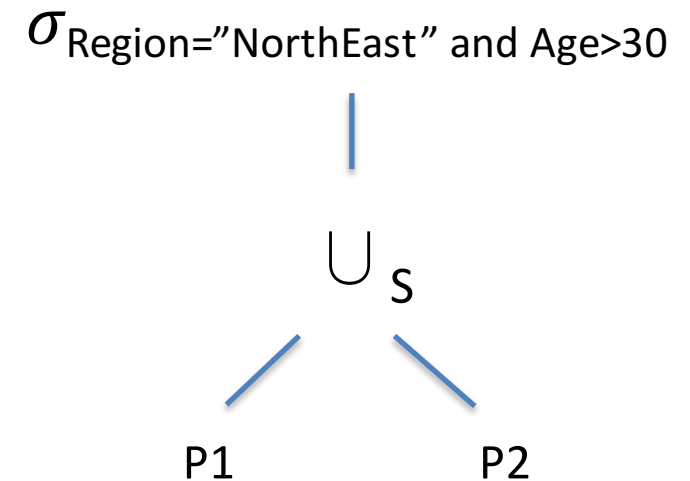
select * from P2) T

where T.region = "NorthEast" and T.age > 30

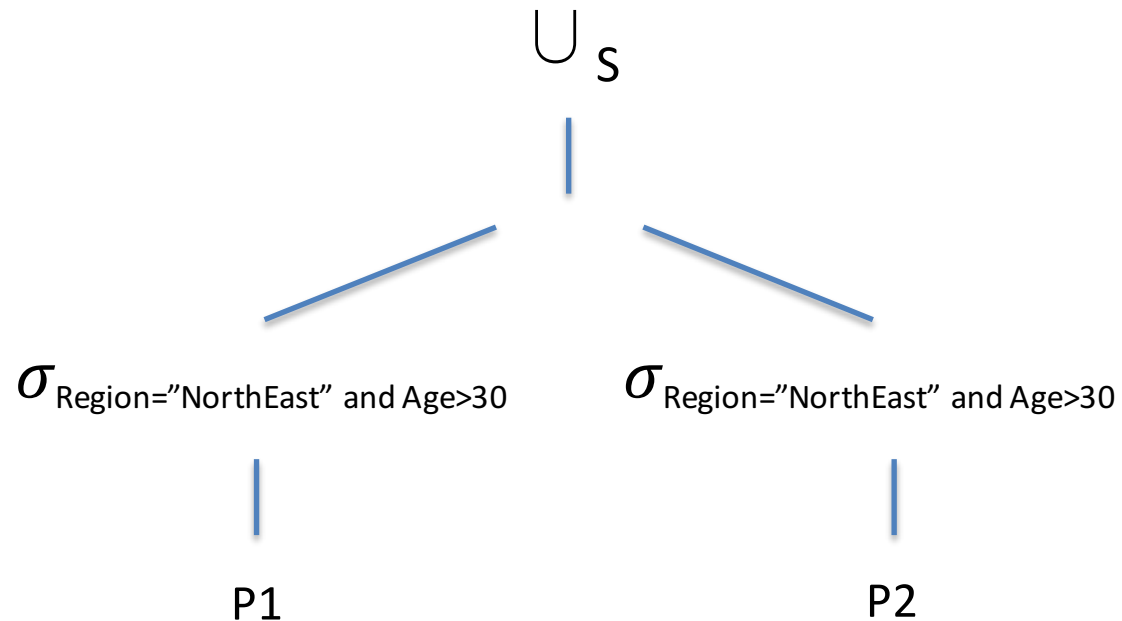
Assuming that half of the nodes has 300 buffer frames available, and the other half has 500 buffer frames available, tell which is the method that we should use to answer the query, and tell which is the elapsed time to compute the answer.

Solution exercise 9

```
select *  
from (select * from P1  
      union  
      select * from P2) T  
where T.region = "NorthEast" and T.age > 30
```

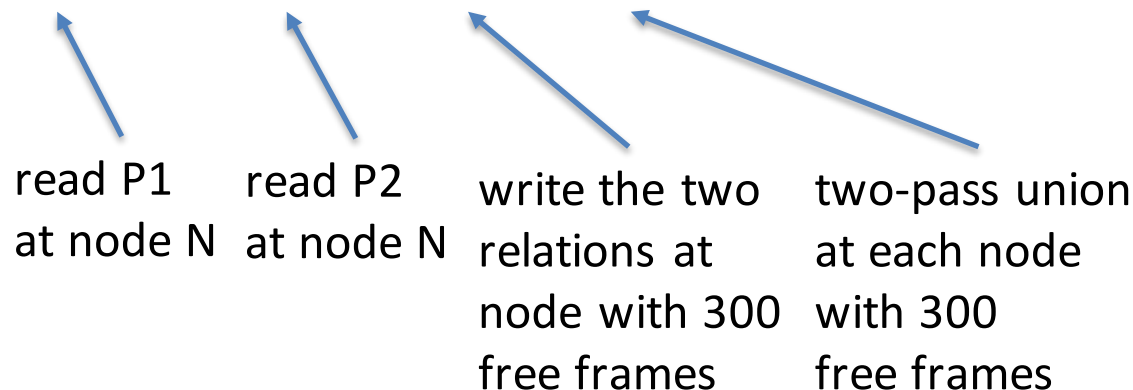


Immediately transformed as follows:



Solution exercise 9

- We know that all tuples relevant for the queries are stored in one node, say N. We assume that customers are uniformly distributed to regions and we conclude that number of pages for each relation at node N is $(50.000.000/20)/100 = 25.000$.
- At node N, we read one page at a time of P1, we select only those tuples with age > 30 (we assume that the selectivity factor is 1/3), we hash them, and we uniformly distribute the buckets in the 20 nodes. We do the same for P2. At the end, in each node we will have $(25.000/3)/20 = 417$ pages for each relation, 834 in total.
- In all nodes with 500 buffer frames, we can do the union in one-pass, by storing the tuples of P1 in the buffer, and then reading in the buffer the tuples of P2 while receiving them. In the other nodes, we have to do it in two-pass, after storing the tuples of P1 and P2 received. So, the cost (the elapsed time) is dominated by the nodes where we have to use the two-pass algorithm. In particular:
- $25.000 + 25.000 + 834 + 3 \times 834 = 53.336$




Exercise 10

Assume we have the relations

$R(\underline{A}, B, C, D)$

$S(E, F, G, H, I, \underline{L})$

R occupies 1.500 pages, with 20 tuples per page, and has a dense clustering tree-based index on A using alternative 2.

S occupies 4.800 pages. 

We have to answer the following query:

$\text{select distinct } A, B, C, D, F, G$

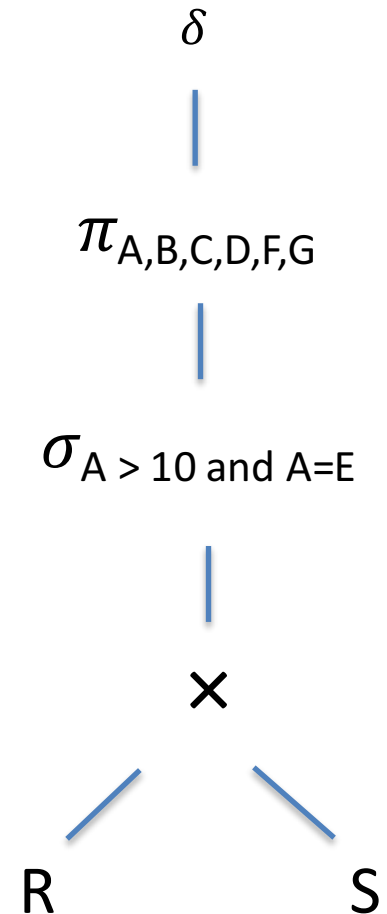
$\text{from } R, S$

$\text{where } A > 10 \text{ and } A = E$

Under the assumption that we have 50 free frames in the buffer.

Solution exercise 10

Here is the logical plan associated
with the query code:



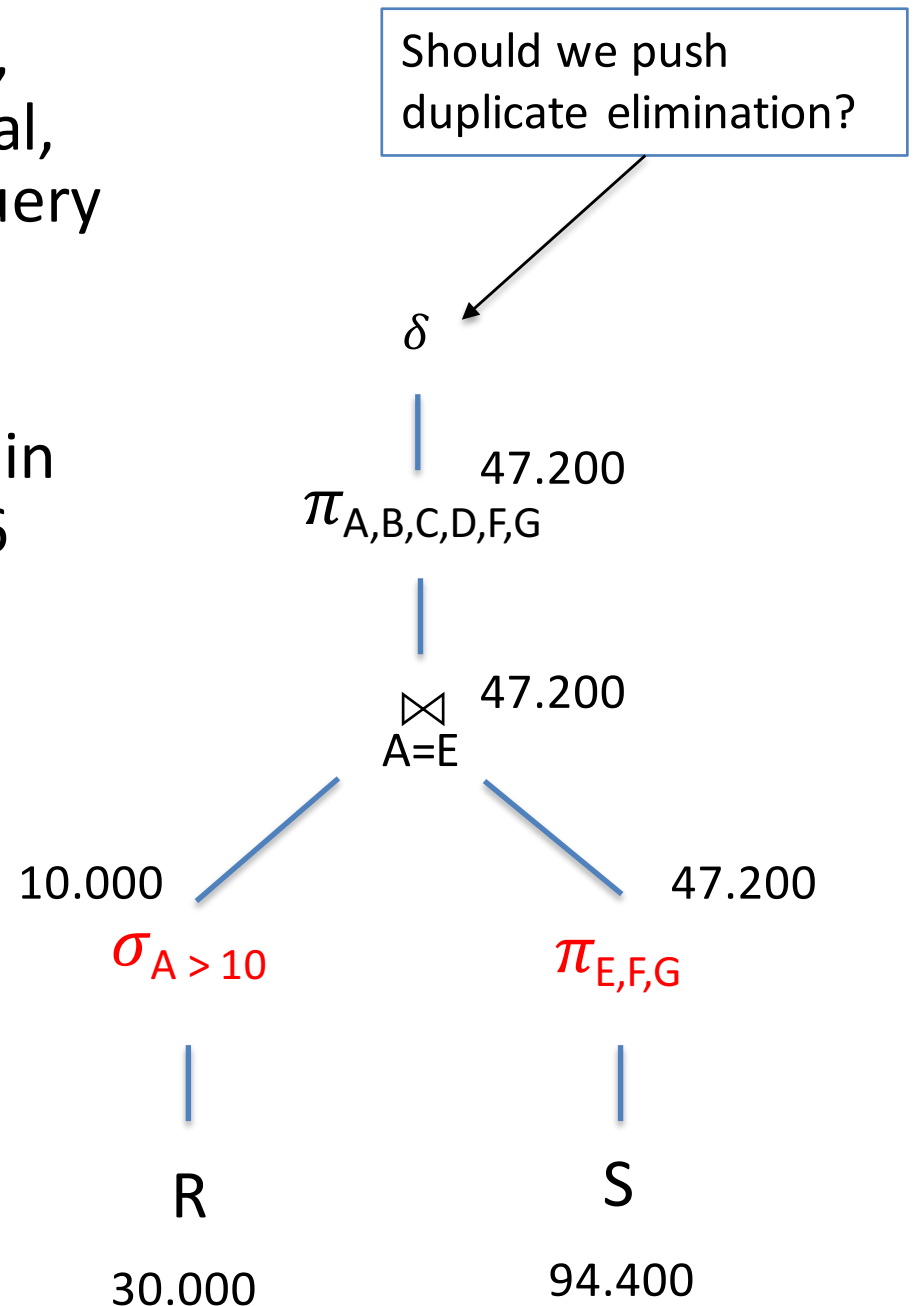
select distinct A,B,C,D,F,G
from R, S
where A > 10 and A=E

Solution exercise 10

Since pushing selection and, in this case, pushing projection is obviously beneficial, we immediately transform the logical query plan as follows:

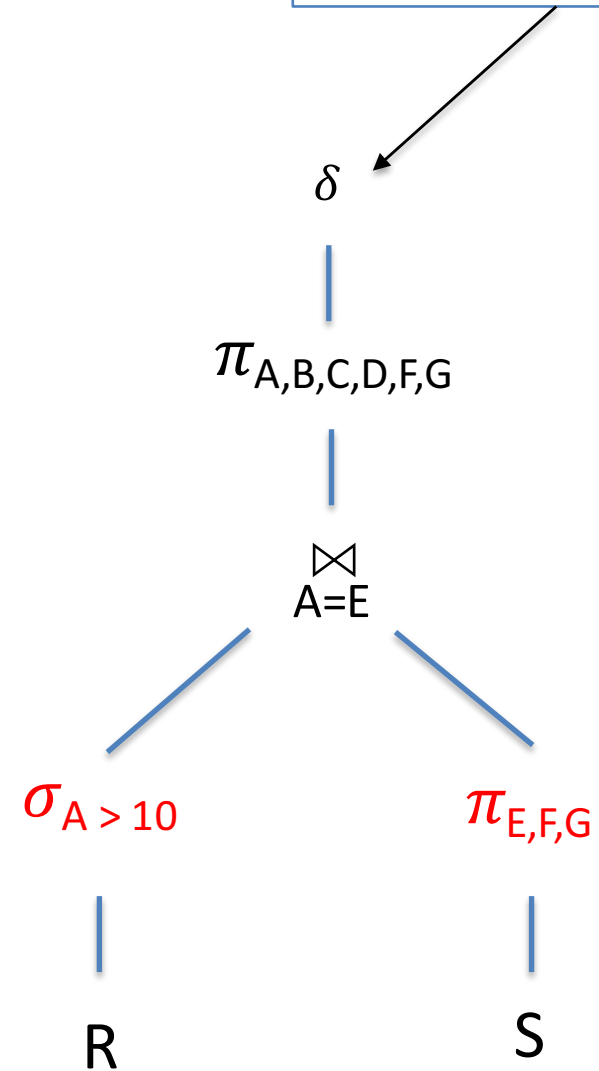
Note that, since 20 tuples of 4 values fit in one page, we assume that 13 tuples of 6 values fit in one page. Therefore, we assume that S has 94.400 tuples.

select distinct A,B,C,D,F,G
from R, S
where A > 10 and A=E



Solution exercise 10

Should we push
duplicate elimination?

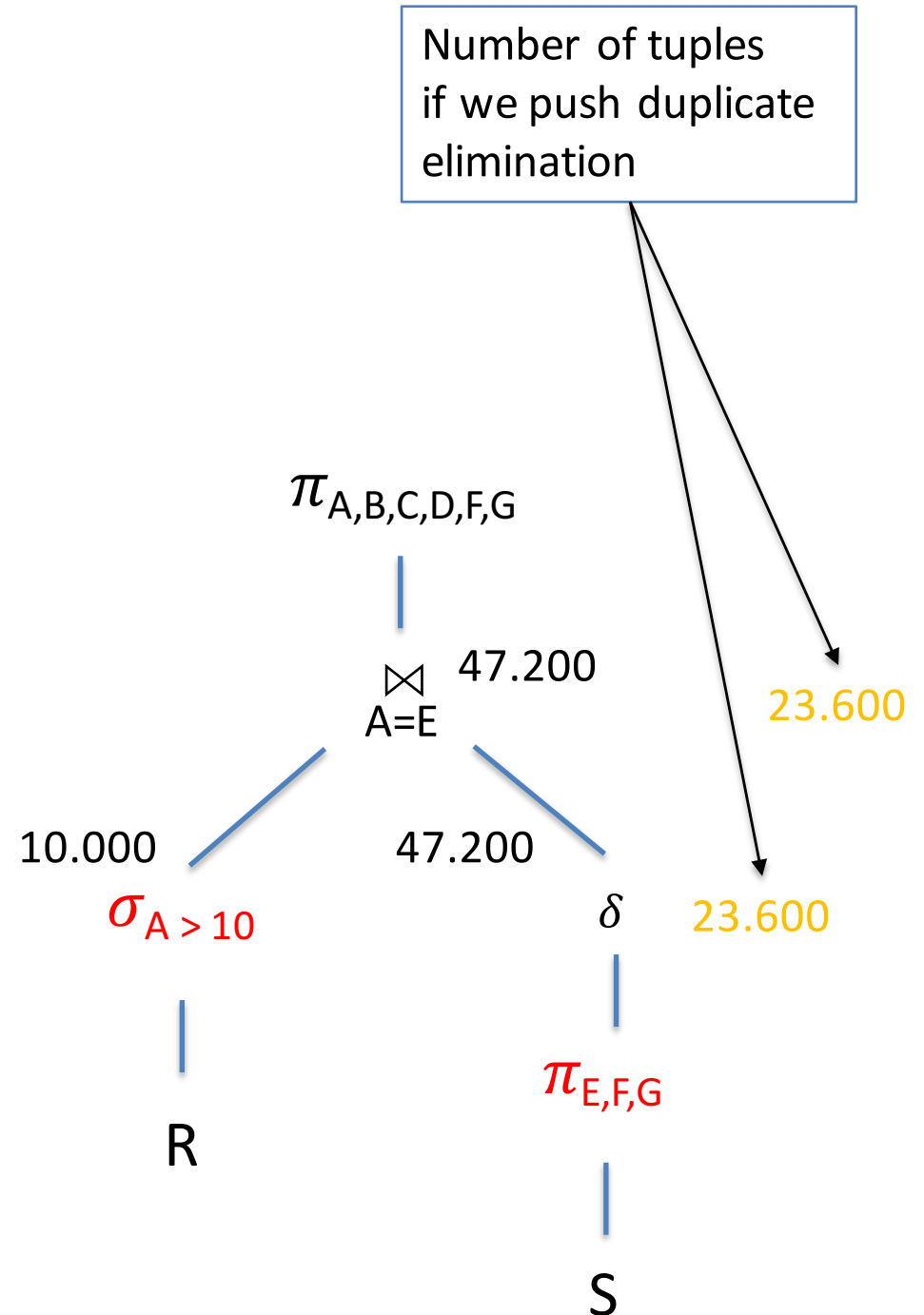


select distinct A,B,C,D,F,G
from R, S
where A > 10 and A=E

Solution exercise 10

If we remain at the logical level, it seems that pushing duplicate elimination is beneficial (note that pushing duplicate elimination on R is useless).

select distinct A,B,C,D,F,G
from R, S
where A > 10 and A=E



Solution exercise 10

But let focus on physical query plan.

First of all, the selection operator will be done through the index, and therefore, let us compute the cost of accessing the index.

R occupies 1.500 pages, with 20 tuples per page, and the associated index is a dense clustering tree-based index on A using alternative 2. Note that R has 30.000 tuples.

Since 20 tuples of 4 values fit in one page, 40 data entries (of 2 values) fit in one page. Since we have to store 30.000 values, the leaves of the tree require $30.000/40=750$ pages, and taking into account the 67% occupancy rule, we have 1.125 leaves.

Also, we can assume 30 as fan out, and therefore, the cost of accessing the index with a specific value for the search key is $\log_{30} 1.125 = 3$.

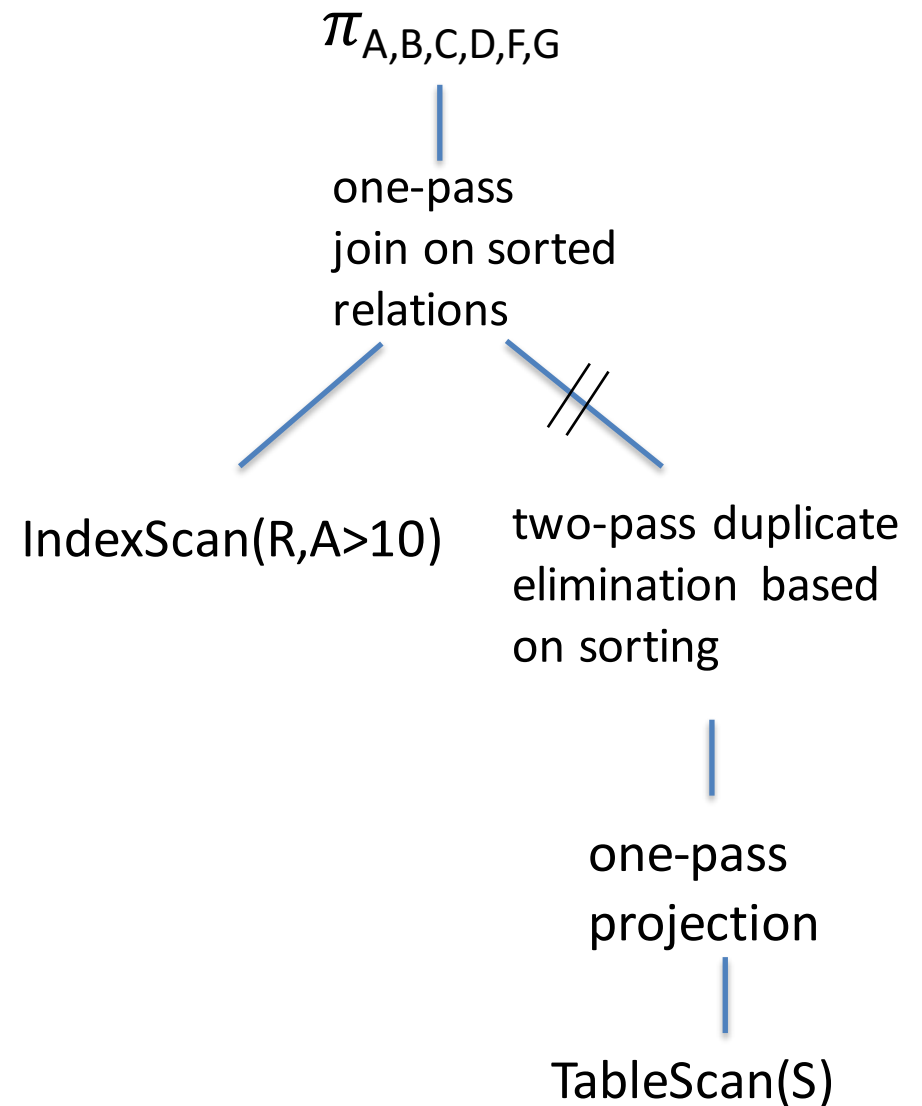
Solution exercise 10

However, if we consider the physical plan, we realize that, if we push duplicate elimination, the corresponding cost in terms of page accesses is

$$4.800 + 2 \times 2.400 + 1.200 + 1.503 + 1.200 = 13.503$$

where:

- 4.800 is the cost of table scan of S
- 2×2.400 is the cost of duplicate elimination, i.e., writing of sublists and merge of sublists for the $4.800/2$ pages of the projection of S
- 1.200 pages is the cost of writing the result of duplicate elimination (which we consider half of the input)
- $1.503 + 1.200$ is the cost of the one-pass join: 1.503 is the cost of reading the first operand (where 3 is the access to the index), and 1.200 is the number of pages of the second operand



Solution exercise 10

On the other hand, if we do not push duplicate elimination, the physical plan is the one shown here:

The corresponding cost in terms of page accesses is

$$4.800 + 2 \times 2.400 + 1.503 + 1.200 = 12.303$$

where

- 4.800 is the cost of table scan of S
- 2×2.400 is the first pass of the two-pass sort-merge join (reading and writing $2.400/50 = 48$) for the $4.800/2$ pages of the projection of S
- $1.503 + 1.200$ is the cost of the second pass (merge phase) of the two-pass sort-merge join, taking into account that we have the relation coming out from the index scan already sorted so, 49 sorted relations and merged using 50 free frames)
- one-pass join: 1.503 is the cost of reading the first operand (where 3 is the access to the index), and 1.200 is the number of pages of the second operand

We conclude that this is the best physical query plan.

