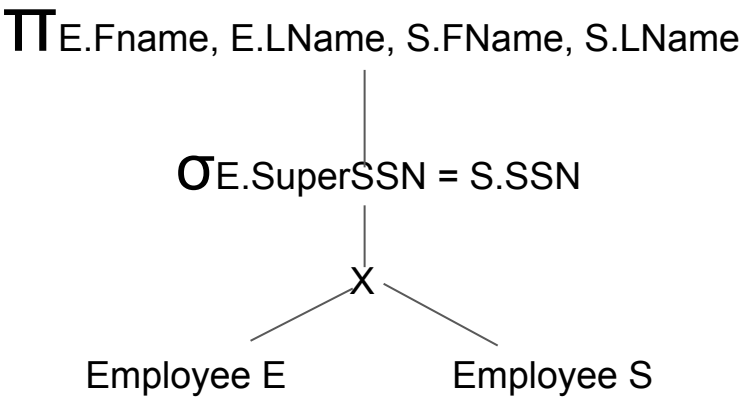
15.13 Consider SQL queries Q8, Q27 from Chapter 8.

Draw at least two query trees that can represented each of these queries. (Canonical and Optimized)

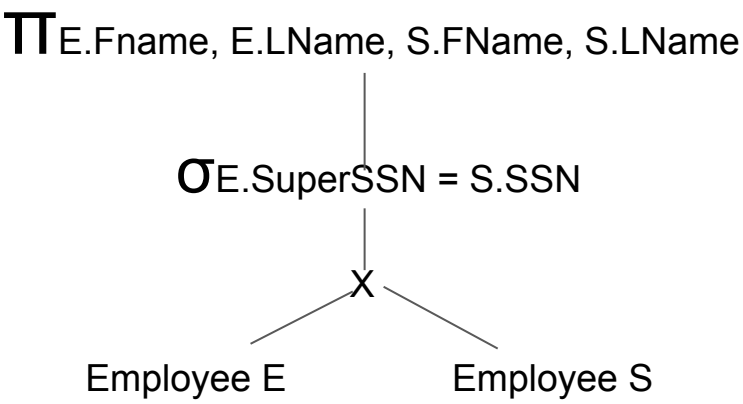
Step 1 : canonical Form

Q8) Select E.Fname, E.LName, S.FName, S.LName from Employee E, Employee S where E.SuperSSN = S.SSN



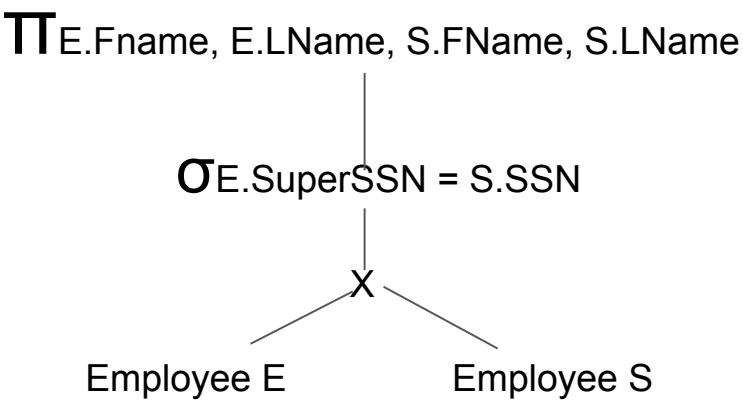
Step 2 : Move Selection Down

Q8) There is nothing we can do at this step



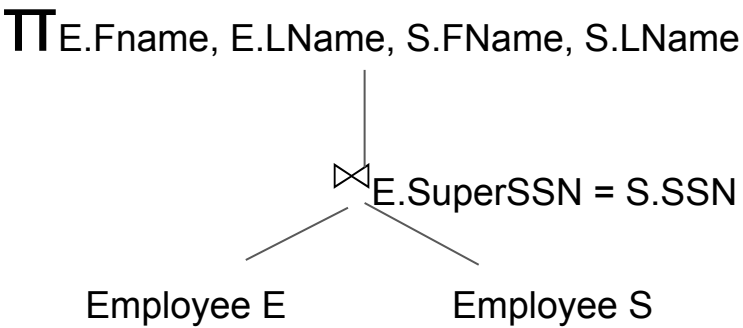
Step 3 : Apply Most restrictive select

Q8) There is nothing we can do at this step



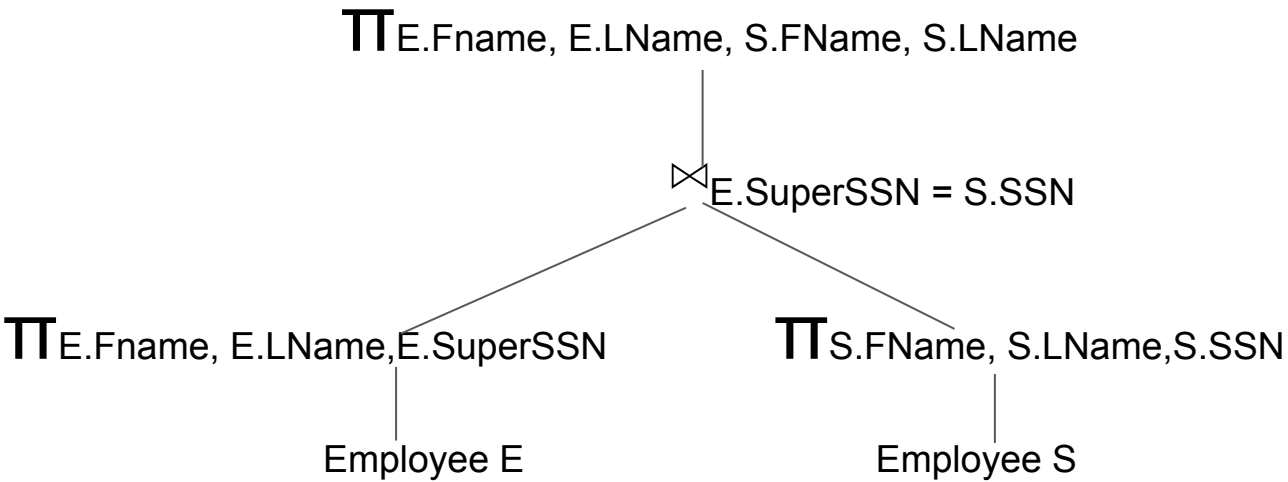
Step 4 : Replace (Cartesian product and Select) with join

Q8)



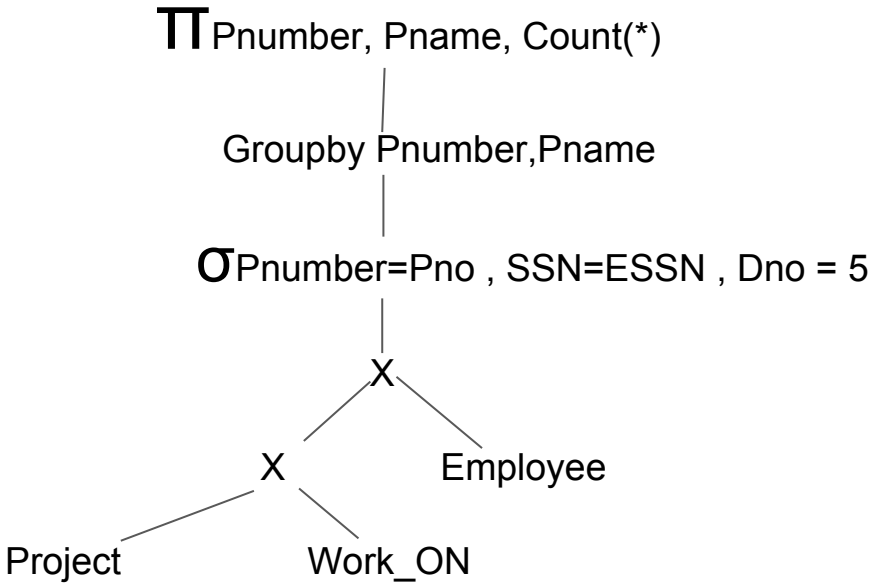
Step 5 : Moving projection operations down

Q8)

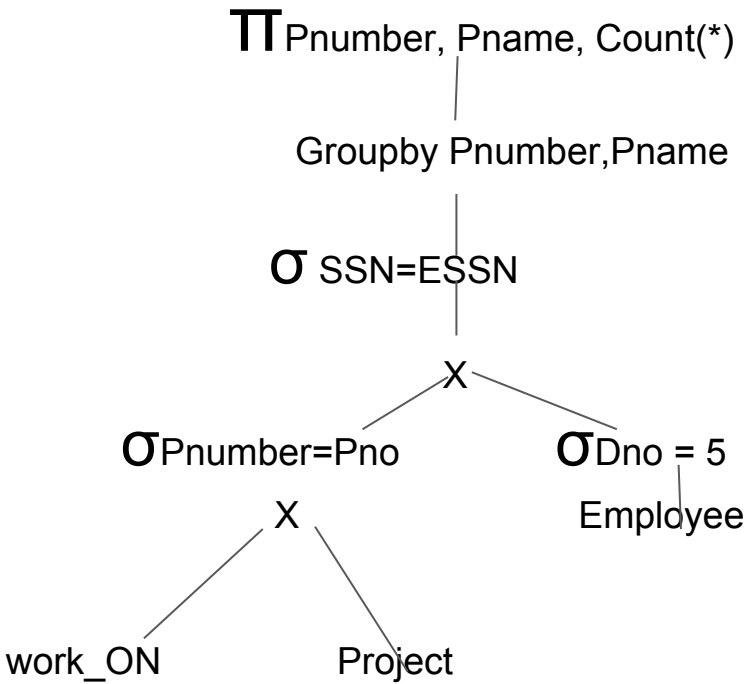


Step 1 : canonical Form

Q8) Select Pnumber, Pname, Count(*) From Project, WORK_ON, Employess where Pnumber=Pno and SSN=ESSN and pno = 5 Groupby Pnumber and Pname

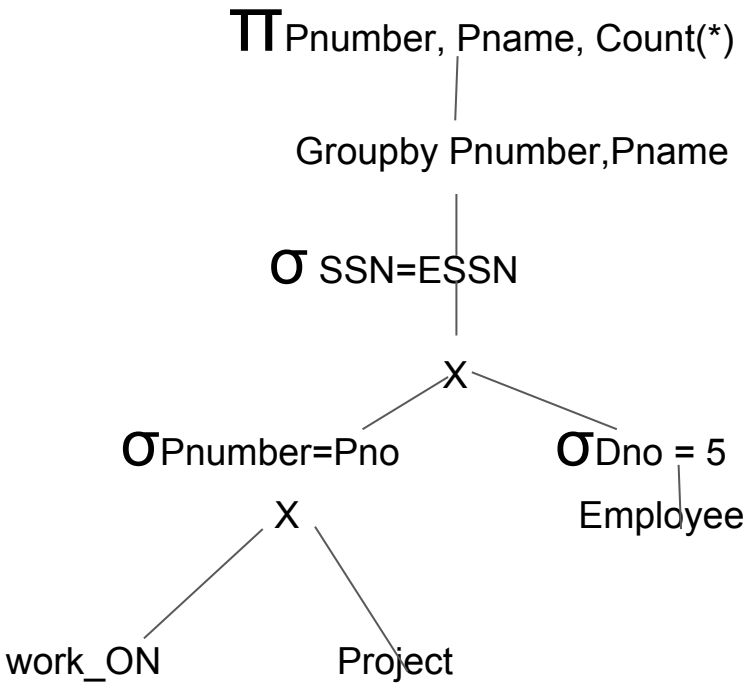


Step 2 : Moving selection down

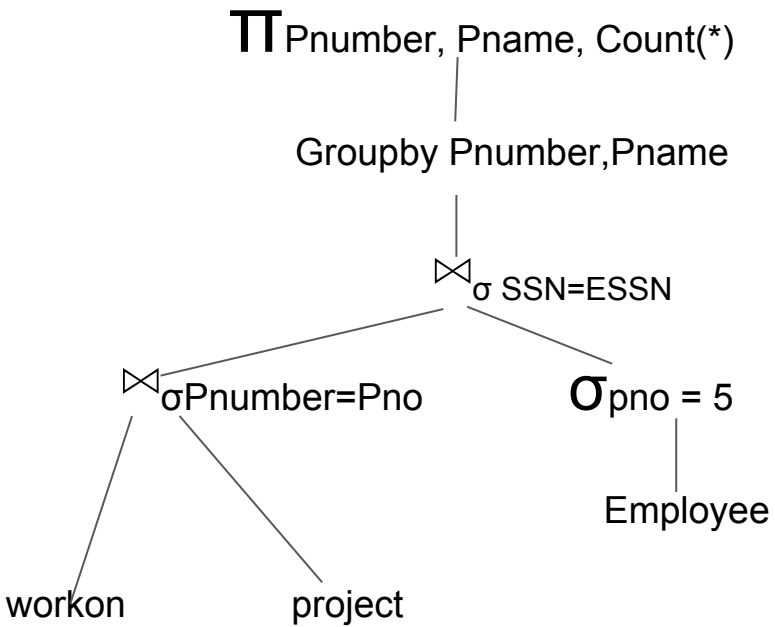


Step 3 : Apply More restrictive selection

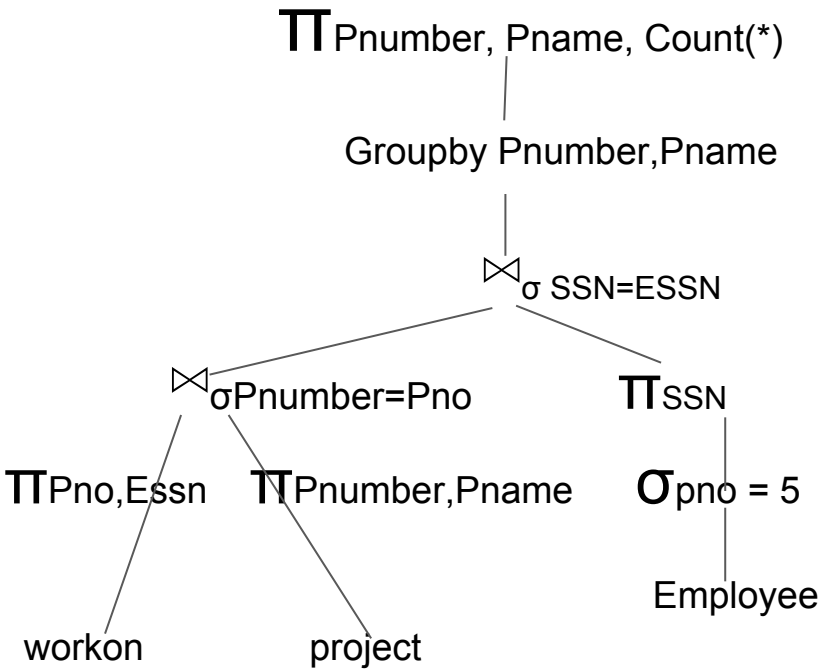
Assume no. of projects less than number of employees in a single Dep. or reverse the join



Step 4 : Replace (Cartesian product and Select) with join



Step 5 : Moving projection operations down



15.14 A file of 4096 blocks is to be sorted with an available buffer space of 64 blocks. How many passes will be needed in the merge phase of the external sort-merge algorithm?

15.14 A file of 4096 blocks is to be sorted with an available buffer space of 64 blocks. How many passes will be needed in the merge phase of the external sort-merge algorithm?

Sort-Phase: no. of partitions = Ceiling(4096/64) = 64
Merge Phase: pass1 = Ceiling(no. of partitions / buffer space -1) = 64/63 = 2
Pass2 = Ceiling (2/63) = 1

Two Passes in merge. (can you think of another formula)

15.17 Can a nondense (sparse) index be used in the implementation of an aggregate operator? Why or why not?

15.17 Can a nondense (sparse) index be used in the implementation of an aggregate operator? Why or why not?

**If the keys in the index correspond to the smallest key value in the data block then the sparse index could be used to compute the MIN function.
However, the MAX function could not be determined from the index.

**since all values do not appear in the index, AVG, SUM and COUNT could not be determined from just the index.

15.21 Extend the sort-merge join algorithm to implement the left outer join.

15.21 Extend the sort-merge join algorithm to implement the left outer join.

If the records of R and S are physically sorted (ordered) by value of the join attributes A and B, respectively, we can implement the join in the most efficient way possible.

Both files are scanned in order of the join attributes, matching the records that have the same values for A and B OR the Values in A that doesn’t exist in B

In this method, the records of each file are scanned only once each for matching with the other file—unless both A and B are non-key attributes, in which case the method needs to be modified slightly.

15.15 Develop (approximate) cost functions for the PROJECT, UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT algorithms

15.15 Develop (approximate) cost functions for the PROJECT, UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT algorithms

Assume that we have bR blocks for table R and bS blocks for table S
Projection:

Read file : bR
Write only projection list: bR2 (should be bRestemp depending on the no. of the field)

If the project list doesn't contain a unique key then we need to sort
bR2 log bR2 and remove duplicates
And the write the final Result bRes
Final Result = bR +bR2 + bR2 log bR2 + bRes

15.15 Develop (approximate) cost functions for the PROJECT, UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT algorithms

Assume that we have bR blocks for table R and bS blocks for table S
Union :

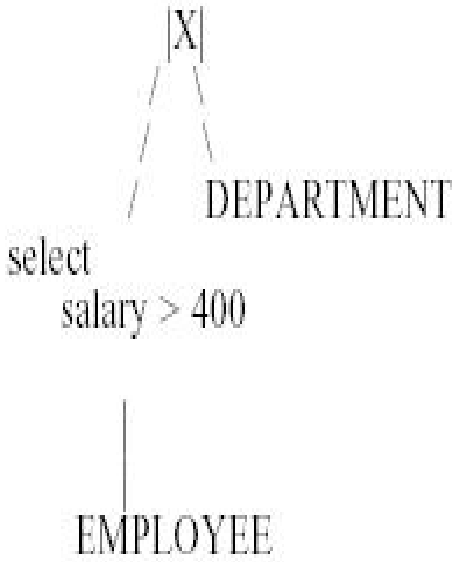
Sort two files : bR log bR + bS log bS
Read and Merge(write) sorted files: bS+bR + bRes

Or you can merge then remove duplicates

Plan 1

15.22 Compare the cost of two different query plans for the following query: for salary > 400 select (EMPLOYEE |X| DNO=DNUMBER DEPARTMENT)

No. of unique salaries = 500
No. of Employees = 10000
No. of Employees blocks = 2000
No. of unique departments = 50
No of dep. Blocks = 5
No. of levels BTree Index = 1 + leaf
No. of leaves in Salary index = 50



Column information. (b) Table information. (c) Index information.

(a)

Table_name	Column_name	Num_distinct	Low_value	High_value
PROJECT	Plocation	200	1	200
PROJECT	Pnumber	2000	1	2000
PROJECT	Dnum	50	1	50
DEPARTMENT	Dnumber	50	1	50
DEPARTMENT	Mgr_ssn	50	1	50
EMPLOYEE	Ssn	10000	1	10000
EMPLOYEE	Dno	50	1	50
EMPLOYEE	Salary	500	1	500

(b)

Table_name	Num_rows	Blocks
PROJECT	2000	100
DEPARTMENT	50	5
EMPLOYEE	10000	2000

(c)

Index_name	Uniqueness	Blevel*	Leaf_blocks	Distinct_keys
PROJ_PLOC	NONUNIQUE	1	4	200
EMP_SSN	UNIQUE	1	50	10000
EMP_SAL	NONUNIQUE	1	50	500

*Blevel is the number of levels without the leaf level.

Plan 1

15.22 Compare the cost of two different query plans for the following query: for salary > 400 select (EMPLOYEE |X| DNO=DNUMBER DEPARTMENT)

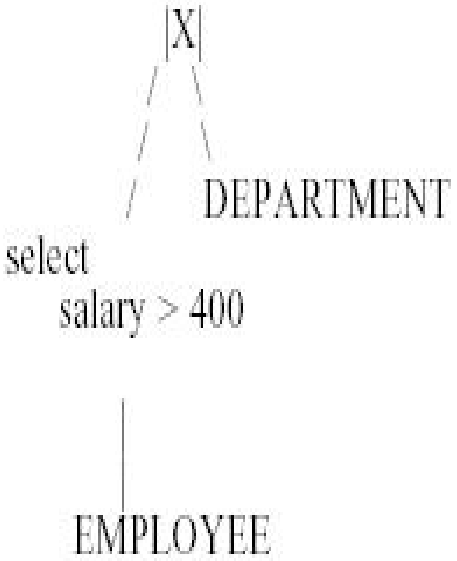
Percent of salaries chosen = (500 - 400) / 500 = ⅕

No. of salaries index block access =
blevel + ⅕ x no.Of leaves = 1 + ⅕*50 = 11

No. of records that represent salary>400 =
⅕ x total no of records = ⅕ x 10000 = 2000

Given uniform dist. Of records, no of block access
Needed to read records after index=2000

To store this result we need = 2000*2000/10000 =400



Plan 1

15.22 Compare the cost of two different query plans for the following query: for salary > 400 select (EMPLOYEE |X| DNO=DNUMBER DEPARTMENT)

Total cost of selection = 11 + 2000 + 400

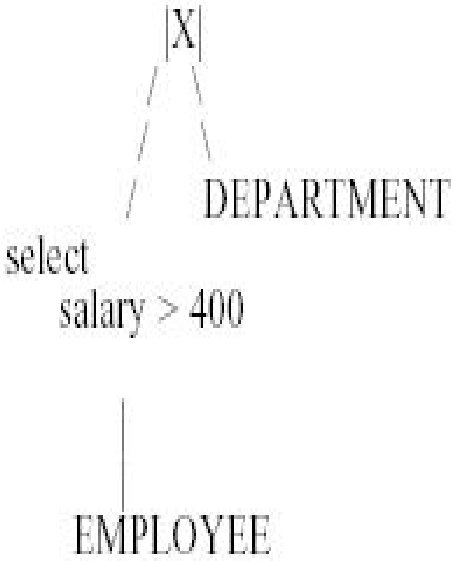
To join with dep.

Read dep = 5

Perform nest loop join = 5*400

//cost of result ignored bec. It will be the same in both P

Total cost is 11+2000+400+5+5*400 = 4461 block acces



Plan 1

15.22 Compare the cost of two different query plans for the following query: for salary > 400 select (EMPLOYEE |X| DNO=DNUMBER DEPARTMENT)

If we have more than 5 memory buffer this cost could be

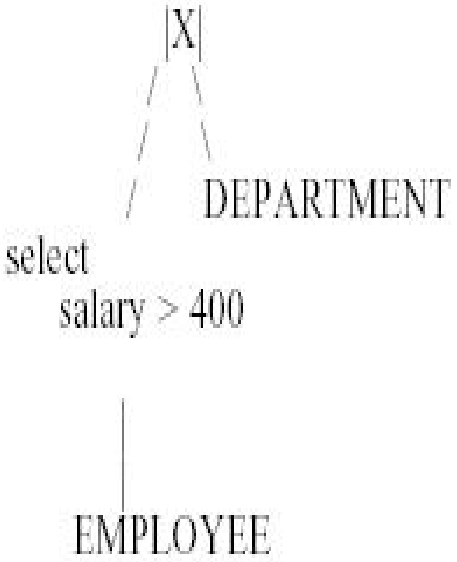
To join with dep.

Read dep = 5

Perform single loop join = 400

Total cost is 11+2000+400+5+400 = 2861 block access

If we used pipelining, cost will reduce to 11+2000+5



Plan 2

15.22 Compare the cost of two different query plans for the following query: for salary > 400 select (EMPLOYEE |X| DNO=DNUMBER DEPARTMENT)

To join with dep.

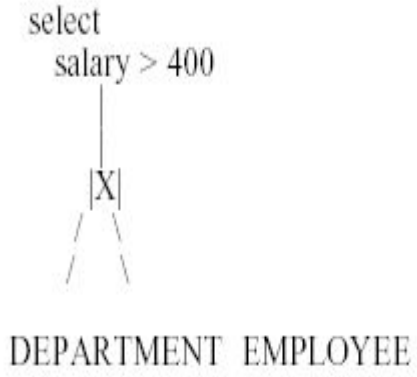
Read dep = 5

Perform nested loop join = 5*2000=10000

Or Given more than 5 memory buffers

Perform join using single loop = 2000

We can use pipelining to avoid writing overhead



Plan 2

15.22 Compare the cost of two different query plans for the following query: for salary > 400 select (EMPLOYEE |X| DNO=DNUMBER DEPARTMENT)

Selection =

writing of final result (which is the same in both plans)