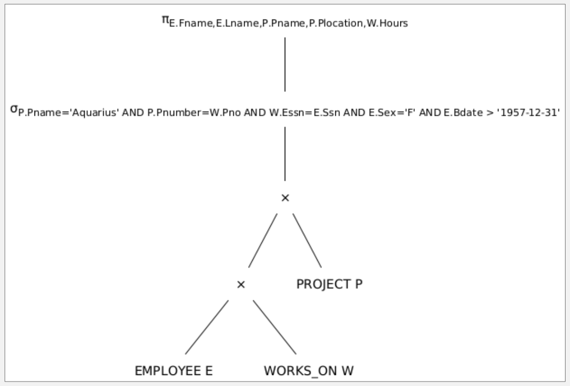
Heuristic Query Optimization Summary

1. Break up conjunctive SELECT conditions (ANDs) into a series of individual SELECTs.
2. Move each SELECT as far down as possible.
3. Combine CROSS PRODUCT and SELECT into JOIN.
4. Reorder consecutive SELECTs so the most selective are done first.
5. Reorder JOINs so the most restrictive operations are done first. (without introducing CROSS PRODUCTs)
6. Break up PROJECTs and move them as far down as possible.
7. Address opportunities for pipelining.

Initial Query Tree

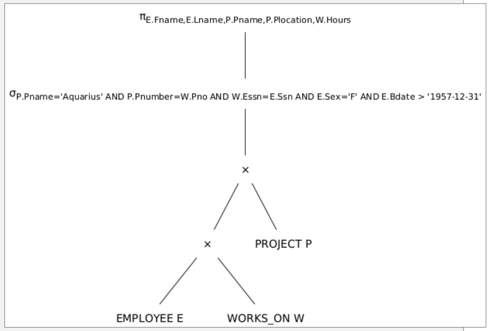
```
SELECT E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours
FROM EMPLOYEE E, WORKS_ON W, PROJECT P
WHERE P.Pname='Aquarius' AND P.Pnumber=W.Pno AND
      W.Essn=E.Ssn AND E.Bdate > '1957-12-31' AND E.Sex='F'
```

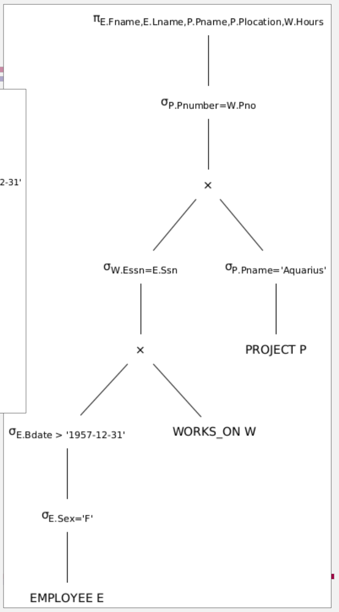

$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\sigma_{P.Pname='Aquarius' \text{ AND } P.Pnumber=W.Pno \text{ AND } W.Essn=E.Ssn \text{ AND } E.Sex='F' \text{ AND } E.Bdate > '1957-12-31'}$
×
×
PROJECT P
EMPLOYEE E
WORKS_ON W

Algorithm

1. Break up conjunctive SELECT conditions into a series of individual SELECT operations.
2. Move each SELECT as far down as possible.
   - SELECT is commutative (order of SELECTs doesn't matter)
   - SELECT can be moved below PROJECT
   - SELECT can be moved below JOIN and CROSS PRODUCT if the selection condition involves only attributes in one of the relations being joined
   - SELECT can be moved below UNION, INTERSECTION, and DIFFERENCE

Move SELECTs Down


$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\sigma_{P.Pname='Aquarius' \text{ AND } P.Pnumber=W.Pno \text{ AND } W.Essn=E.Ssn \text{ AND } E.Sex='F' \text{ AND } E.Bdate > '1957-12-31'}$
×
×
PROJECT P
EMPLOYEE E
WORKS_ON W

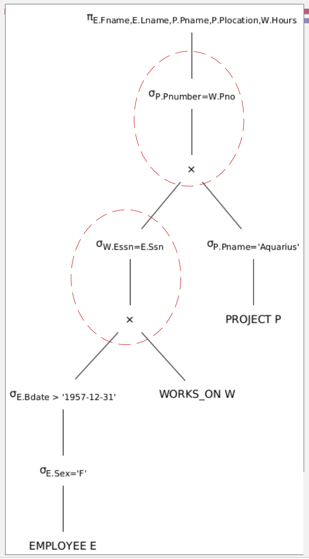

$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\sigma_{P.Pnumber=W.Pno}$
×
$\sigma_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
×
PROJECT P
$\sigma_{E.Bdate > '1957-12-31'}$
WORKS_ON W
$\sigma_{E.Sex='F'}$
EMPLOYEE E

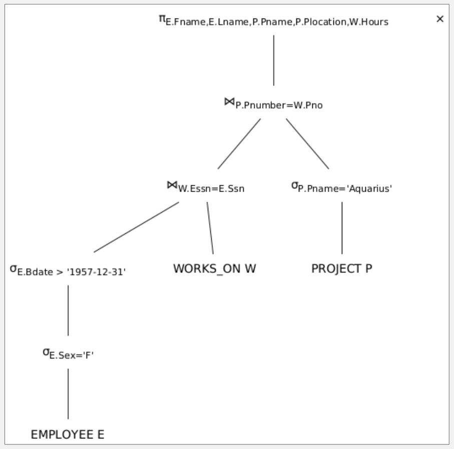
```
EMPLOYEE(Fname,Minit,Lname,Ssn,Bdate,Address,
         Sex,Salary,Super_ssn,Dno)
PROJECT(Pname,Pnumber,Plocation,Dnum)
WORKS_ON(Essn,Pno,Hours)
```

Algorithm

3. Combine CROSS PRODUCT and SELECT into JOIN.
   - requires SELECT condition to be a conjunction of equality comparisons

Transform CROSS PRODUCT Into JOIN


$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\sigma_{P.Pnumber=W.Pno}$
$\times$
$\sigma_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\times$
PROJECT P
$\sigma_{E.Bdate > '1957-12-31'}$
WORKS_ON W
$\sigma_{E.Sex='F'}$
EMPLOYEE E


$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{P.Pnumber=W.Pno}$
$\bowtie_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\sigma_{E.Bdate > '1957-12-31'}$
WORKS_ON W
PROJECT P
$\sigma_{E.Sex='F'}$
EMPLOYEE E

```
EMPLOYEE(Fname,Minit,Lname,Ssn,Bdate,Address,Sex,
         Salary,Super_ssn,Dno)
PROJECT(Pname,Pnumber,Plocation,Dnum)
WORKS_ON(Essn,Pno,Hours)
```

Algorithm

4. Reorder consecutive SELECTs so that the most selective are done first.
   - SELECT is commutative (order of SELECTs doesn't matter)
   - most selective can be based on the number of result tuples or the selectivity or rule of thumb (equality more selective than range)

Selectivity is the ratio of rows resulting from the SELECT to initial rows: $|\sigma_c(R)|/|R|$

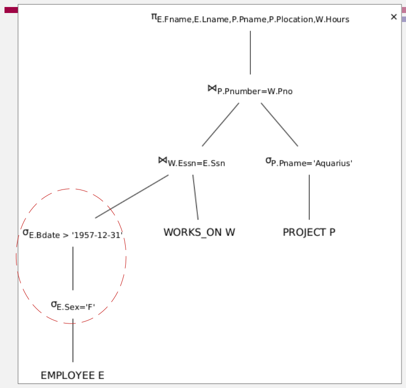
- can be estimated from DB catalog info
  – use number of distinct values to estimate for equality conditions
  – use range of values to estimate for range conditions

Database Catalog Info

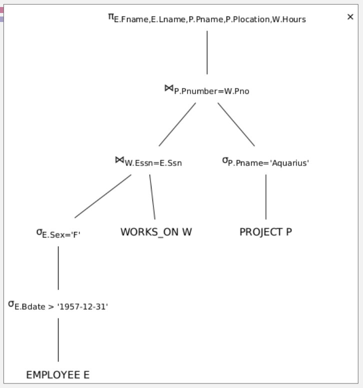
table	column	# distinct values (d)	low value	high value
PROJECT	Pnumber	11	1	92
	Pname	11		
	Plocation	9		
	Dnum	5	1	8
WORKS_ON	Pno	11	1	92
	Essn	38		
EMPLOYEE	Ssn	40		
	Bdate	40	1927-11-10	1980-05-21
	Lname	37		
	Sex	2		

table	# records (r)
PROJECT	11
WORKS_ON	48
EMPLOYEE	40

Reorder SELECTs



$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{P.Pnumber=W.Pno}$
$\bowtie_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\sigma_{E.Bdate > '1957-12-31'}$
WORKS_ON W
PROJECT P
$\sigma_{E.Sex='F'}$
EMPLOYEE E



$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{P.Pnumber=W.Pno}$
$\bowtie_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\sigma_{E.Sex='F'}$
WORKS_ON W
PROJECT P
$\sigma_{E.Bdate > '1957-12-31'}$
EMPLOYEE E

$\sigma_{Bdate>'1957-12-31'}$(EMPLOYEE)
- date span is approx. 53 years, with 22 years covered by the range → assuming uniform distribution, selectivity is approx. 22/53 = 0.41
- number of result tuples = 0.41*40 = 16

$\sigma_{Sex='F'}$(EMPLOYEE)
- 2 distinct values → assuming uniform distribution, selectivity is 1/2 = 0.5
- number of result tuples = 0.5*40 = 20

Algorithm

5. Reorder JOINs so that the most restrictive operations are first (low and left), as long as CROSS PRODUCT operations aren't introduced.
   - most restrictive can be based on the number of result tuples or the join selectivity
     - heuristic: fewer input tuples leads to fewer result tuples
   - JOIN, CROSS PRODUCT are commutative
   - JOIN, CROSS PRODUCT are associative but there are complications (CROSS PRODUCTs introduced, shifting JOIN conditions) if the join conditions are not limited to consecutive relations

Join selectivity is the fraction of rows actually resulting from a JOIN: $|R \bowtie_c S|/(|R|\ |S|)$

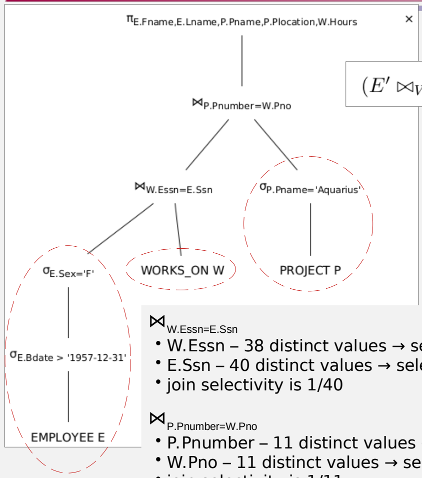
- can be estimated from DB catalog info
  - for join condition R.a = S.b, *js* = min($1/d_a$,$1/d_b$) where d_a and d_b are the number of distinct values for R.a and S.b, respectively
    - *d* = *r* (number of records) for keys

Database Catalog Info

table	column	# distinct values (d)	low value	high value
PROJECT	Pnumber	11	1	92
	Pname	11		
	Plocation	9		
	Dnum	5	1	8
WORKS_ON	Pno	11	1	92
	Essn	38		
EMPLOYEE	Ssn	40		
	Bdate	40	1927-11-10	1980-05-21
	Lname	37		
	Sex	2		

table	# records (r)
PROJECT	11
WORKS_ON	48
EMPLOYEE	40

Reorder JOINs



$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{P.Pnumber=W.Pno}$
$\bowtie_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\sigma_{E.Sex='F'}$
WORKS_ON W
PROJECT P
$\sigma_{E.Bdate > '1957-12-31'}$
EMPLOYEE E

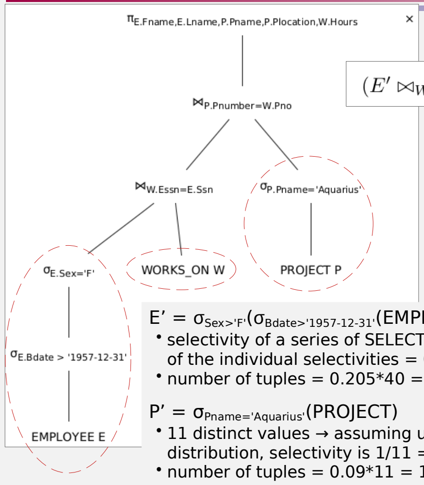
$$(E' \bowtie_{W.Essn=E.Ssn} W) \bowtie_{P.Pnumber=W.Pno} P'$$

$\bowtie_{W.Essn=E.Ssn}$
- W.Essn – 38 distinct values → selectivity is 1/38
- E.Ssn – 40 distinct values → selectivity is 1/40
- join selectivity is 1/40

$\bowtie_{P.Pnumber=W.Pno}$
- P.Pnumber – 11 distinct values → selectivity is 1/11
- W.Pno – 11 distinct values → selectivity is 1/11
- join selectivity is 1/11

based on join selectivity, no reordering

Reorder JOINs



$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{P.Pnumber=W.Pno}$
$\bowtie_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\sigma_{E.Sex='F'}$
WORKS_ON W
PROJECT P
$\sigma_{E.Bdate > '1957-12-31'}$
EMPLOYEE E

$$(E' \bowtie_{W.Essn=E.Ssn} W) \bowtie_{P.Pnumber=W.Pno} P'$$

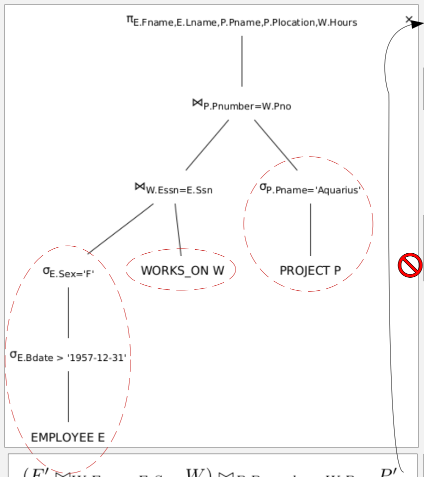
E' = $\sigma_{Sex='F'}(\sigma_{Bdate>'1957-12-31'}(\text{EMPLOYEE}))$
- selectivity of a series of SELECTs is the product of the individual selectivities = 0.41*0.5 = 0.205
- number of tuples = 0.205*40 = 9

P' = $\sigma_{Pname='Aquarius'}(\text{PROJECT})$
- 11 distinct values → assuming uniform distribution, selectivity is 1/11 = 0.09
- number of tuples = 0.09*11 = 1

W = WORKS_ON
- 48 tuples

based on input tuples as a proxy for result tuples, P' is smallest and $\bowtie_{P.Pnumber=W.Pno}$ should be first

Reorder JOINs



$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{P.Pnumber=W.Pno}$
$\bowtie_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\sigma_{E.Sex='F'}$
WORKS_ON W
PROJECT P
$\sigma_{E.Bdate > '1957-12-31'}$
EMPLOYEE E

$$(E' \bowtie_{W.Essn=E.Ssn} W) \bowtie_{P.Pnumber=W.Pno} P'$$

P' is smallest, so put it into the first (lowest) join

$$E' \bowtie_{W.Essn=E.Ssn} (W \bowtie_{P.Pnumber=W.Pno} P')$$

E' is next smallest, but no join condition directly involves P' and E' – this would introduce a cross product

E' $\bowtie_{W.Essn=E.Ssn}$ (P' $\bowtie_{P.Pnumber=W.Pno}$ W)

🚫 (E' × P') $\bowtie_{P.Pnumber=W.Pno \text{ AND } W.Essn=E.Ssn}$ W

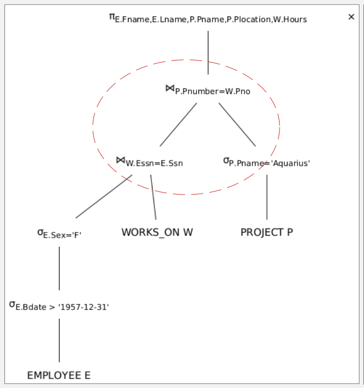
instead just swap join operands so smallest things are on the left

P' $\bowtie_{P.Pnumber=W.Pno}$ W
- js is 1/11
- number of result tuples = js |P'| |W| = (1/11)*1*48 = 4.36

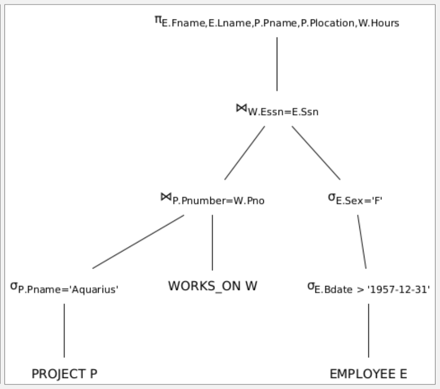
E' $\bowtie_{W.Essn=E.Ssn}$ (P' $\bowtie_{P.Pnumber=W.Pno}$ W)

(P' $\bowtie_{P.Pnumber=W.Pno}$ W) $\bowtie_{W.Essn=E.Ssn}$ E'

Reorder JOINs



$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{P.Pnumber=W.Pno}$
$\bowtie_{W.Essn=E.Ssn}$
$\sigma_{P.Pname='Aquarius'}$
$\sigma_{E.Sex='F'}$
WORKS_ON W
PROJECT P
$\sigma_{E.Bdate > '1957-12-31'}$
EMPLOYEE E

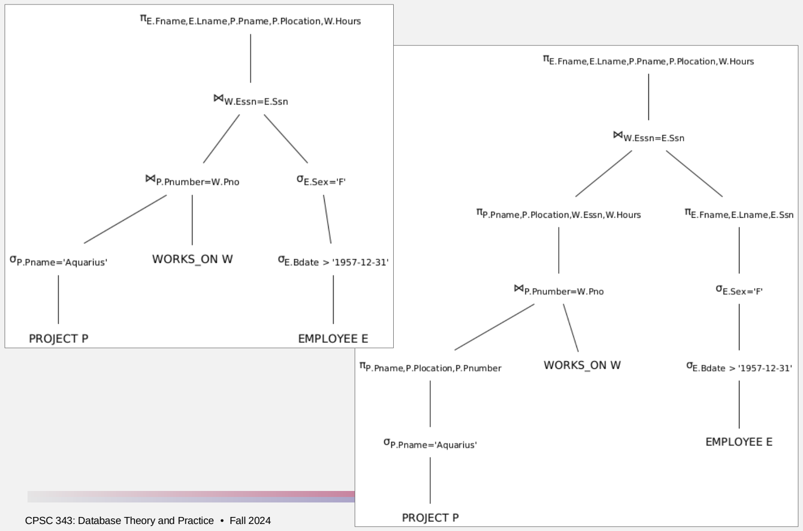


$\pi_{E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours}$
$\bowtie_{W.Essn=E.Ssn}$
$\bowtie_{P.Pnumber=W.Pno}$
$\sigma_{E.Sex='F'}$
$\sigma_{P.Pname='Aquarius'}$
WORKS_ON W
$\sigma_{E.Bdate > '1957-12-31'}$
PROJECT P
EMPLOYEE E

Algorithm

6. Break up PROJECT operations and move them down the tree as far as possible.
   - only include PROJECT immediately above a base table if the PROJECT can be satisfied completely by only reading an index
   - all but the last PROJECT in a series of PROJECTs can be ignored
   - PROJECT can be moved below SELECT if the SELECT condition involves only the attributes in the PROJECT
   - PROJECT can be moved below JOIN if the JOIN condition involves only the attributes in the PROJECT
     - otherwise, PROJECT with just the needed attributes can be added below JOIN
   - PROJECT can be moved below CROSS PRODUCT

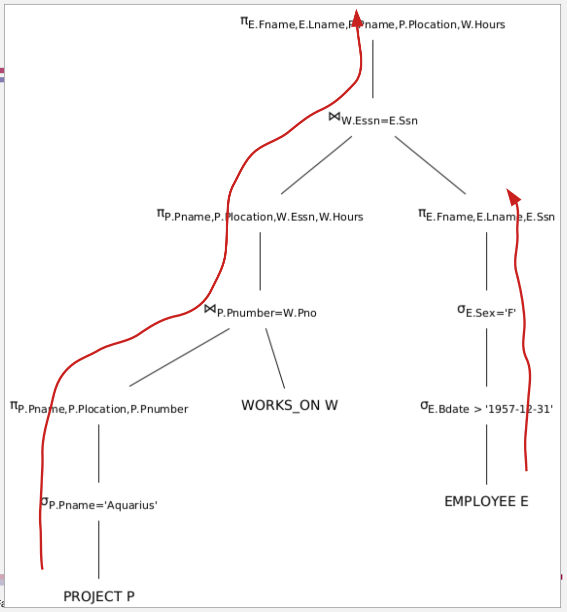
Move PROJECTs Down



Algorithm

7. Address pipelining opportunities.
   - pipelining avoids writing intermediate results to disk – output from one operation is used directly for the next
   - typically (only) possible for unary operations and the left operand of binary operations

Pipelining


π E.Fname,E.Lname,P.Pname,P.Plocation,W.Hours
⋈ W.Essn=E.Ssn
π P.Pname,P.Plocation,W.Essn,W.Hours
π E.Fname,E.Lname,E.Ssn
⋈ P.Pnumber=W.Pno
σ E.Sex='F'
π P.Pname,P.Plocation,P.Pnumber
WORKS_ON W
σ E.Bdate > '1957-12-31'
σ P.Pname='Aquarius'
EMPLOYEE E
PROJECT P