M146 Database Systems
Spring 2020

Georgia Koutrika

Query Optimization

SQL Query

PHASE 1: Logical optimization
Rewrite the query into relational algebra
Find logically equivalent- but more efficient- RA expression

** Relational Algebra allows us to translate SQL queries into precise and optimizable expressions!*

Query tree

PHASE 2: Physical optimization
Find an execution plan with the lowest cost

Execution plan

How is it done?

1. Heuristic rules (on query trees)

2. Cost-based estimation

Query Optimization:
Phase 1. Logical

Example

Let us assume that we have these tables:

PROJECT(Pnumber, Plocation, Dnum)

DEPARTMENT(Dnumber, Dname, Mgr_ssn)

EMPLOYEE(SSN, Fname, Lname, Address, Bdate)

Example

Let us assume that we have these tables:

PROJECT(<u>Pnumber</u>, Plocation, Dnum)

DEPARTMENT(<u>Dnumber</u>, Dname, Mgr_ssn)

EMPLOYEE(<u>SSN</u>, Fname, Lname, Address, Bdate)

Query:
for every project located in ‘Stafford’ that had ‘Smith’ as manager,
retrieve the project number, the controlling department number, and the department manager’s last name, address, and birth date

Example

Let us assume that we have these tables:

PROJECT(Pnumber, Plocation, Dnum, PStartDate)

DEPARTMENT(Dnumber, Dname, Mgr_ssn)

EMPLOYEE(SSN, Fname, Lname, Address, Bdate)

Query:
for every project located in 'Stafford' that had 'Smith' as manager,
retrieve the project number, the controlling department number, and the department manager's last name, address, and birth date

SQL:

```
SELECT P.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate
FROM   PROJECT P, DEPARTMENT D, EMPLOYEE E
WHERE P.Dnum = D.Dnumber AND D.Mgr_ssn = E.SSN
          AND P.Plocation = 'STAFFORD' AND E. Lname='Smith';
```

Example

SQL:

SELECT P.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate

FROM PROJECT P, DEPARTMENT D, EMPLOYEE E

WHERE P.Dnum = D.Dnumber AND D.Mgr_ssn = E.SSN
AND P.Plocation = ‘STAFFORD’ AND E. Lname=’Smith’;

Rewrite the query into **relational algebra**

Relational algebra (naïve conversion):

$\pi_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}}(\sigma_{\text{PLOCATION='STAFFORD' AND LNAME='SMITH' AND DNUM=DNUMBER AND MGRSSN=SSN}}(\text{PROJECT} \times \text{DEPARTMENT} \times \text{EMPLOYEE}))$

Example

Relational algebra (naïve conversion):

$$\pi_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}} (\sigma_{\text{PLOCATION='STAFFORD' AND LNAME='SMITH' AND DNUM=DNUMBER AND MGRSSN=SSN}} (\text{PROJECT} \times \text{DEPARTMENT} \times \text{EMPLOYEE}))$$

Map the relational algebra expression into a **query tree**

Query Tree

- A relational algebra expression can be represented by a query tree
  - Leaf nodes are <u>input relations</u> of the query
  - Internal nodes represent <u>relational algebra operations</u>

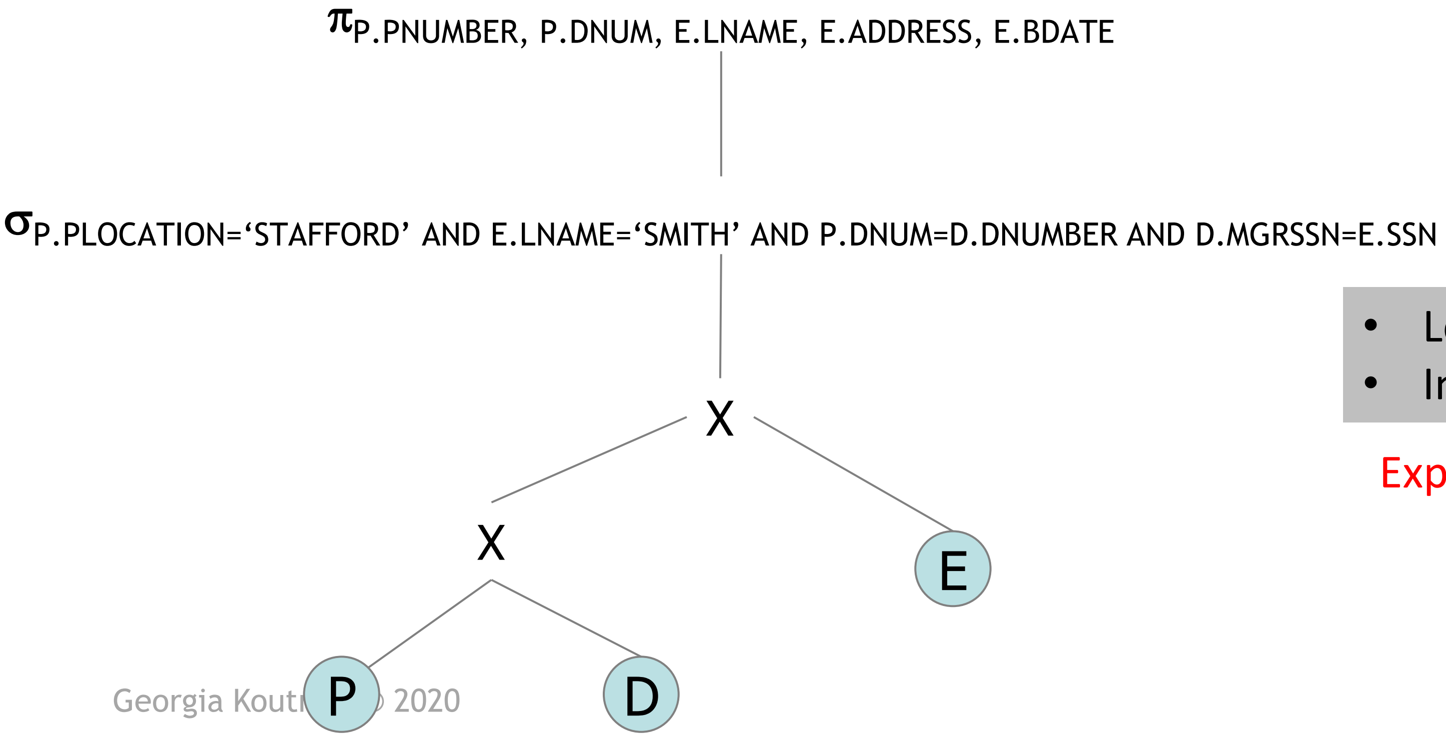
- A query tree can be "executed"
  - Execute an internal node operation whenever its operands are available and then replace the internal node by the relation that results from executing the operation

Example

Relational algebra (naïve conversion):

$$\pi_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}} \left(\sigma_{\text{PLOCATION='STAFFORD' AND DNUM=DNUMBER AND MGRSSN=SSN}} \left(\text{PROJECT} \times \text{DEPARTMENT} \times \text{EMPLOYEE} \right) \right)$$

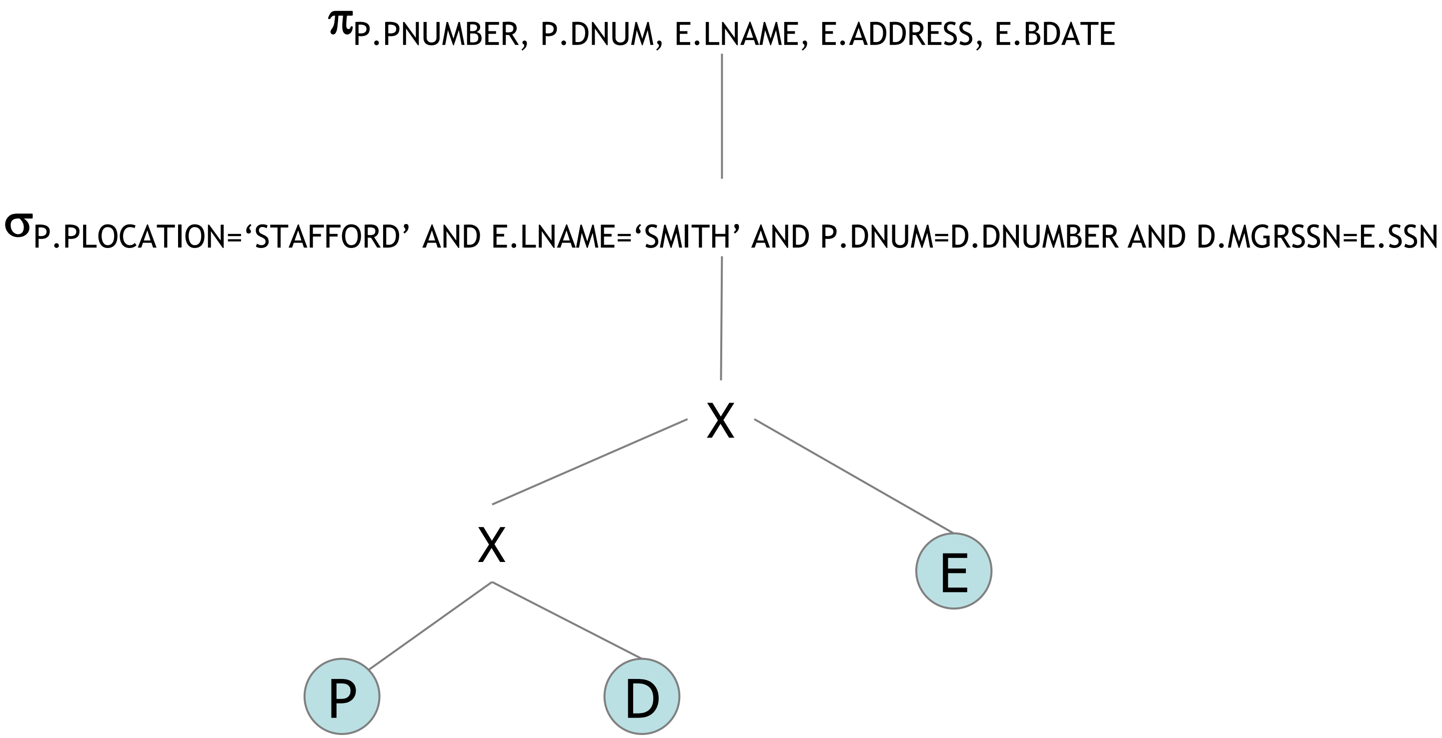
Map the relational algebra expression into a **query tree**



- Leaf nodes are relations
- Internal nodes are relational algebra operations

Expensive to process!

Example



The task of heuristic optimization to find **a final query tree that is efficient** to execute by applying a set of **heuristics rules**

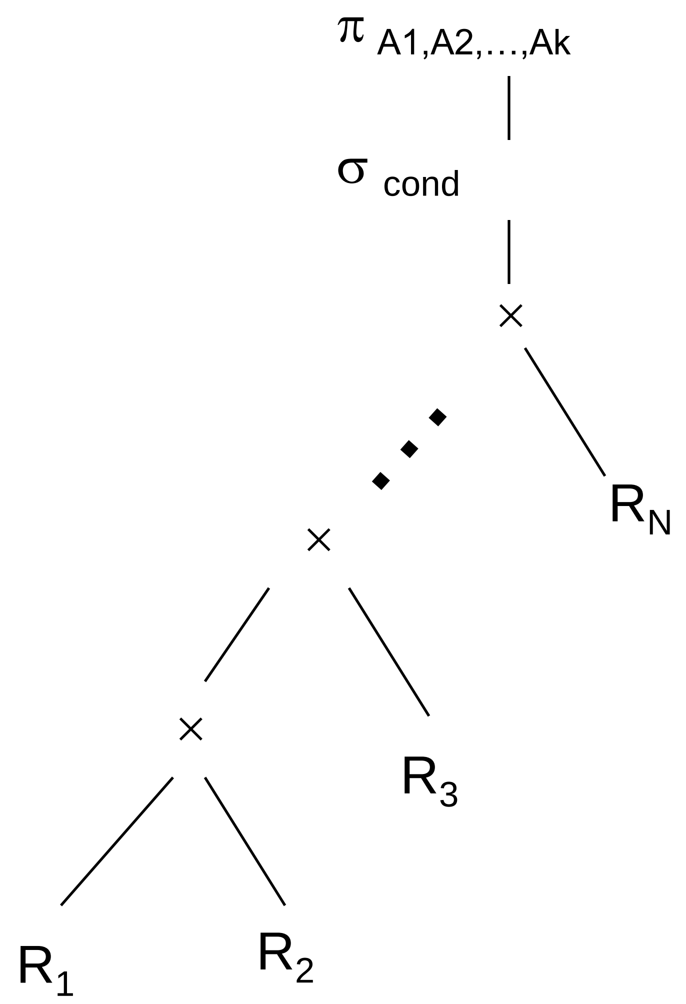
Heuristics Rules for Query Optimization

Query:

SELECT DISTINCT $A_1, A_2, \ldots, A_k$
FROM $R_1, R_2, \ldots, R_N$
WHERE Cond

STEP 0: Naïve conversion to relational algebra:

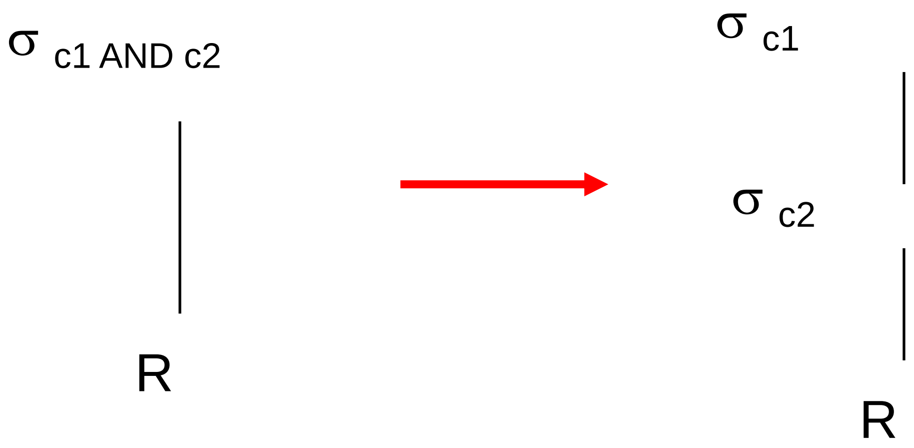
$$\pi_{A1,A2,\ldots,Ak} (\sigma_{Cond} (R_1 \times R_2 \times \ldots \times R_N))$$



Using Heuristics in Query Optimization

STEP 1: Break up any select operations with conjunctive conditions into a **cascade of select operations**

$$\sigma_{c1 \text{ AND } c2 \text{ AND } \ldots \text{ AND } cN}(R_1 \times R_2 \times \ldots \times R_N) = \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cN}(R_1 \times R_2 \times \ldots \times R_N)\,)\,)$$



Note: Disjuncts cannot be split like the conjuncts. We can separate disjuncts by union but it may NOT be useful.

$\sigma_{c1 \text{ OR } c2} = \sigma_{c1} \cup \sigma_{c2}$ and the two selections can be done in any order

Using Heuristics in Query Optimization

STEP 2: **Push SELECT operation** as far **down** the query tree as permitted

- This is possible due to the commutativity of select operator with other operations
- AND conditions in selection can be separated and reordered.
- Algebraic operations are independent of column positions in the table because they work on column names.

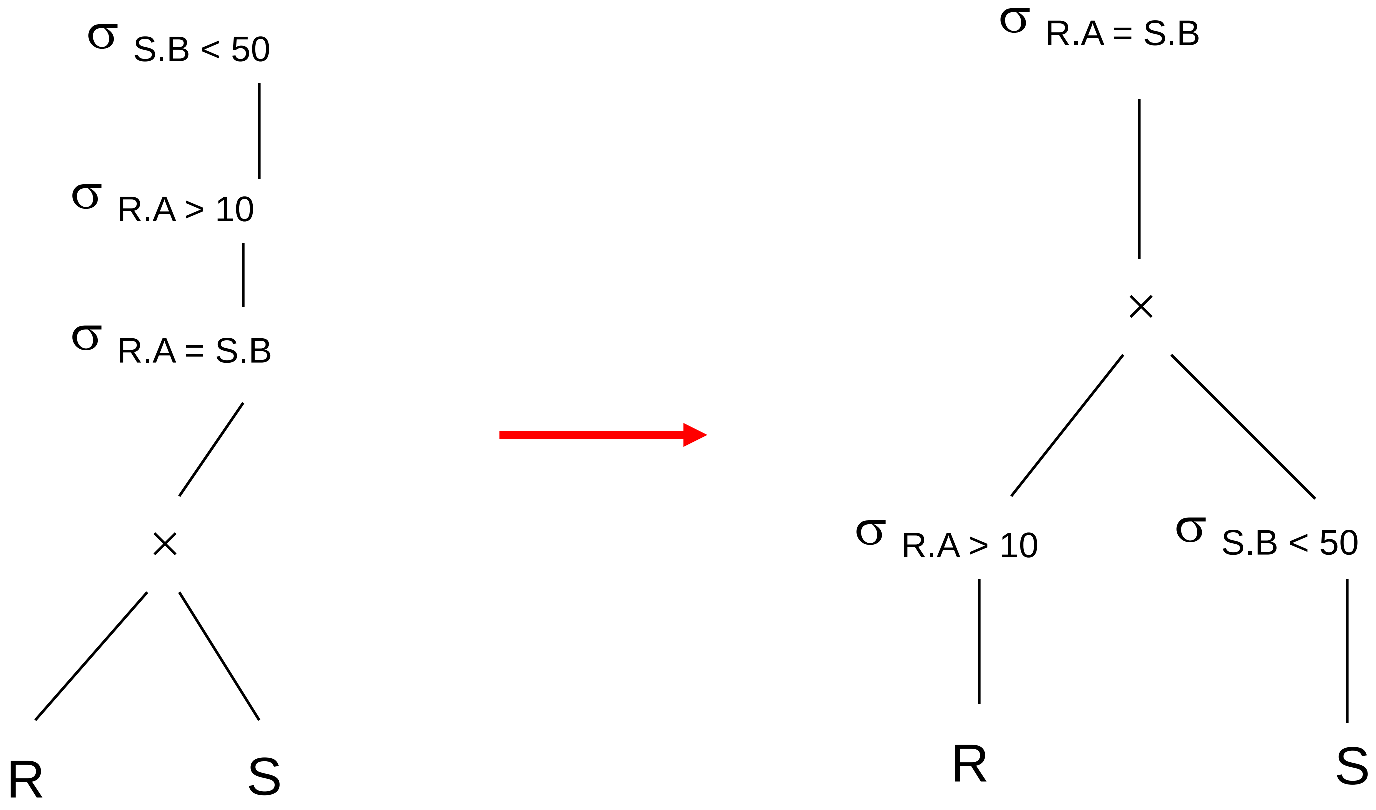
Using Heuristics in Query Optimization

STEP 2: **Push SELECT operation** as far **down** the query tree as permitted

1. $\sigma_{c1} (\sigma_{c2}(R)) = \sigma_{c2} (\sigma_{c1}(R))$
2. $\pi_{A1, A2, \ldots, An} (\sigma_{c} (R)) = \sigma_{c} (\pi_{A1, A2, \ldots, An} (R))$ (assuming c is in Ai)
3. $\sigma_{c1 \text{ AND } c2} (R \bowtie S) = (\sigma_{c1} (R)) \bowtie (\sigma_{c2} (S))$
4. $\sigma_{c1 \text{ AND } c2} (R \times S) = (\sigma_{c1} (R)) \times (\sigma_{c2} (S))$
5. $\sigma_{c} (R \cup S) = (\sigma_{c} (R)) \cup (\sigma_{c} (S))$
6. $\sigma_{c} (R \cap S) = (\sigma_{c} (R)) \cap (\sigma_{c} (S))$
7. $\sigma_{c} (R - S) = (\sigma_{c} (R)) - (\sigma_{c} (S))$

Using Heuristics in Query Optimization

STEP 2: **Push SELECT operation** as far **down** the query tree as permitted



Using Heuristics in Query Optimization

STEP 3: **Rearrange binary operations**

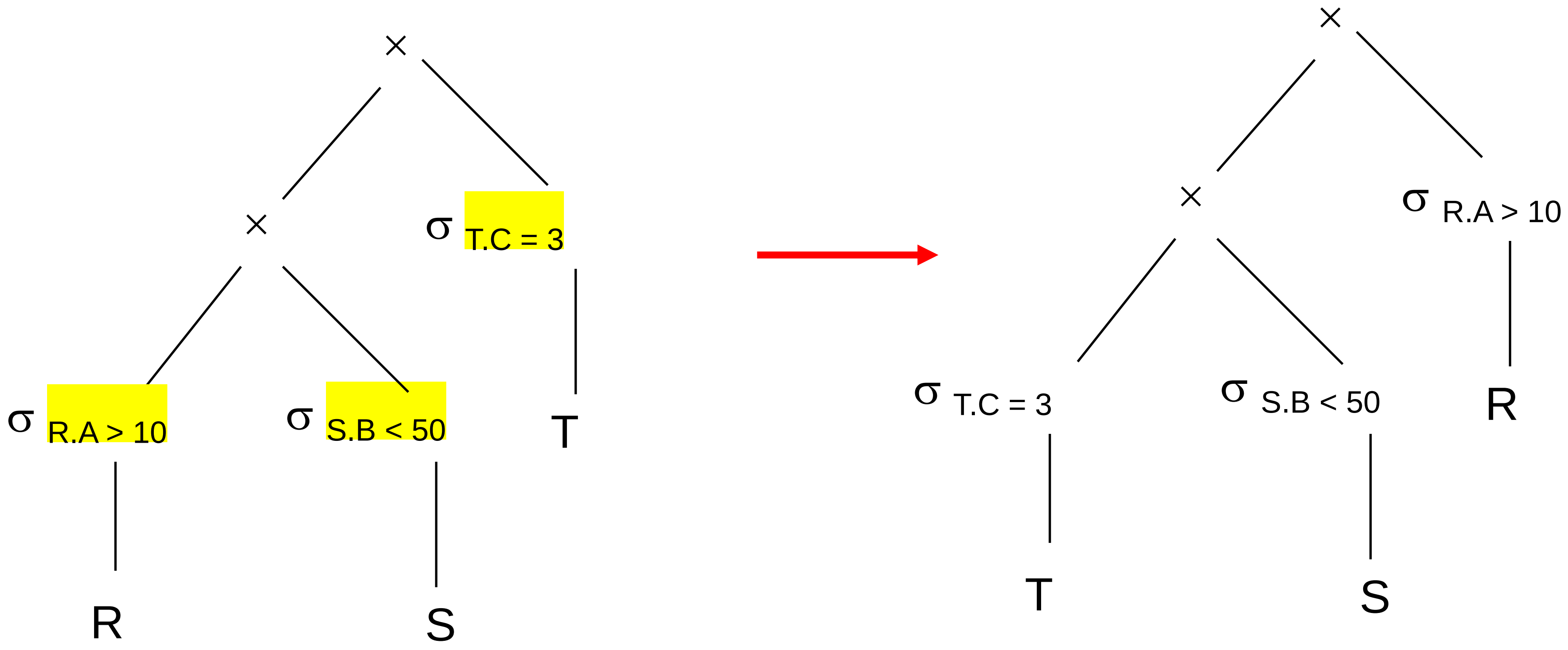
- Position the leaf node relations with **most restrictive SELECT operations to the left** of the query tree
  - Most restrictive: produce relation with fewest tuples or smallest selectivity
    - Selectivity is the ratio of the number of records that satisfy the select (σ) condition

- Based on commutativity and associativity of binary operations
  - R $\bowtie_c$ S = S $\bowtie_c$ R;
  - R x S = S x R
  - (R *op* S) *op* T = R *op* (S *op* T)
    - where *op* is either $\bowtie$, $\times$, $\cup$, or $\cap$

Using Heuristics in Query Optimization

STEP 3: Rearrange binary operations

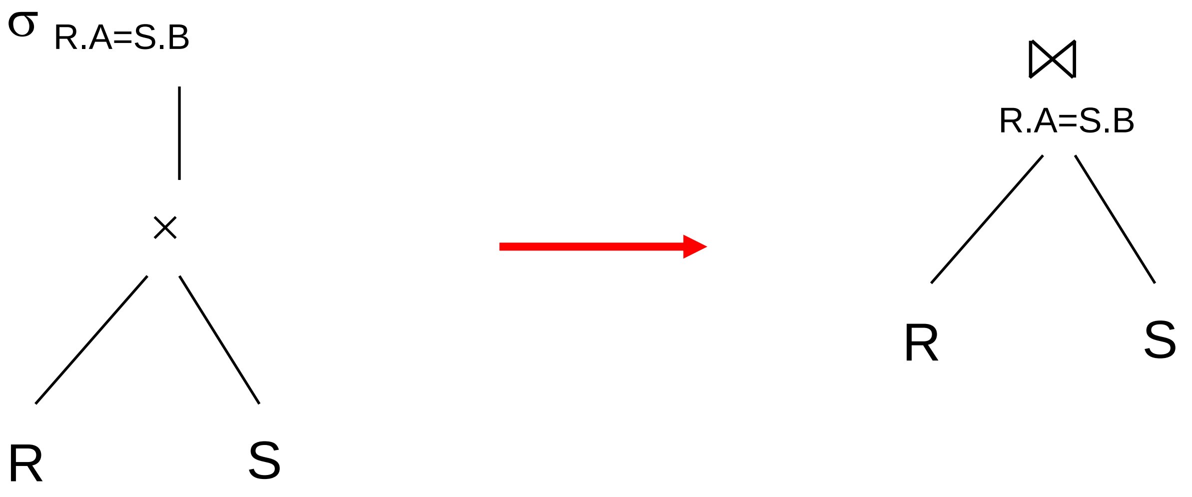
which one is the most restrictive ?



Using Heuristics in Query Optimization

STEP 4: **Combine CARTESIAN PRODUCT with SELECT** operation into a JOIN operation

- $(\sigma_C (R \times S)) = (R \bowtie_C S)$



Using Heuristics in Query Optimization

STEP 5:

Move PROJECT operation down the tree as far as possible by creating new PROJECT operations as needed

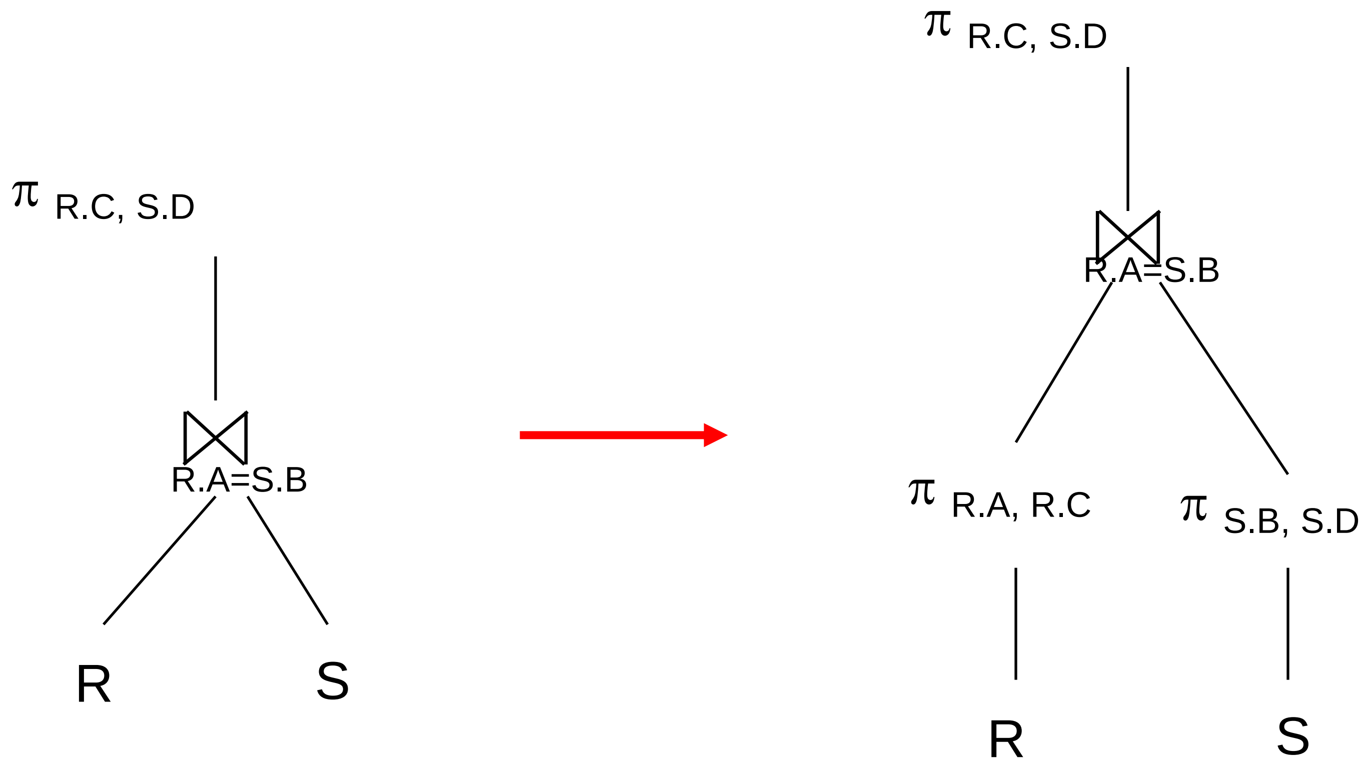
Using cascading and commutativity of PROJECT operations

Using Heuristics in Query Optimization

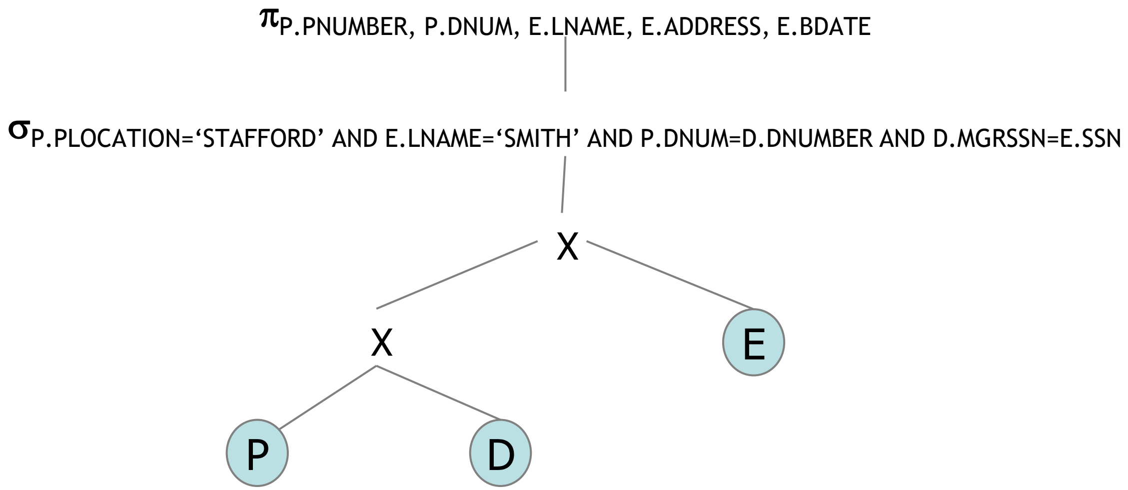
STEP 5:

Move PROJECT operation down the tree as far as possible by creating new PROJECT operations as needed

Using cascading and commutativity of PROJECT operations

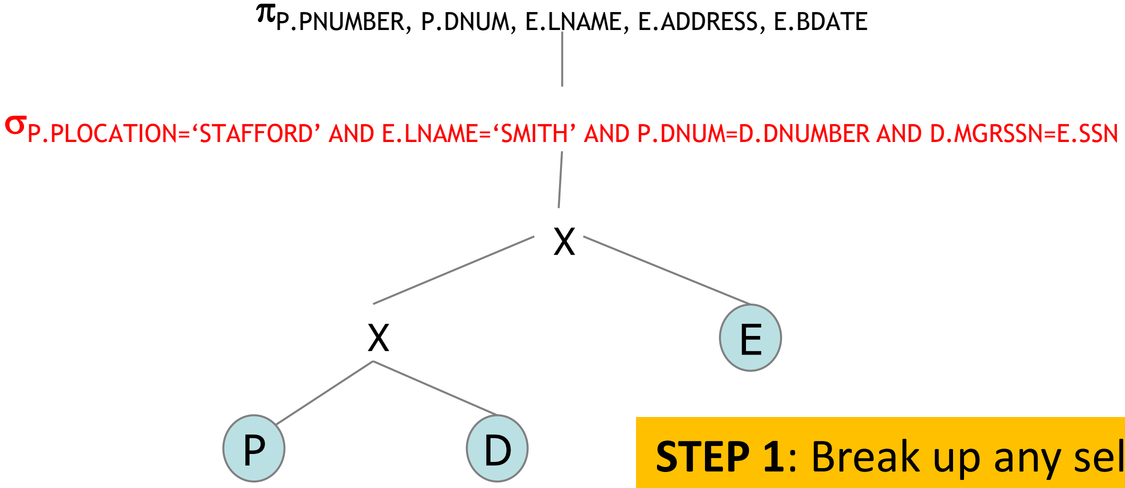


Going back to our Example

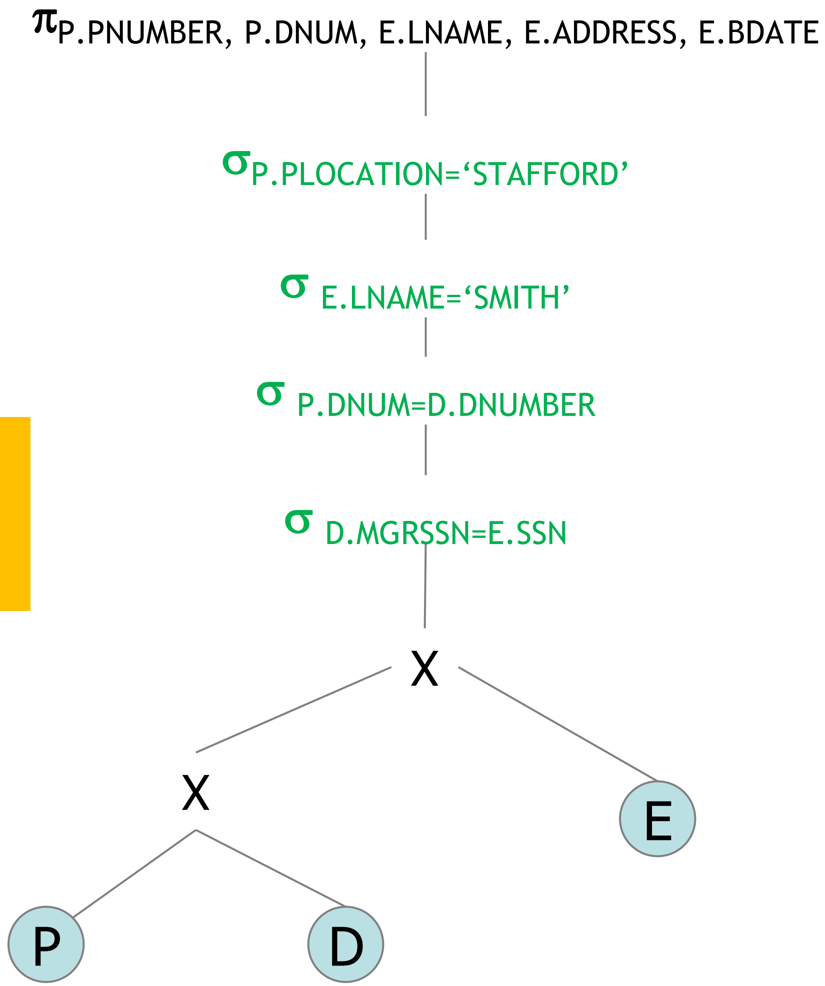


The task of heuristic optimization to find **a final query tree that is efficient** to execute by applying a set of **heuristics rules**

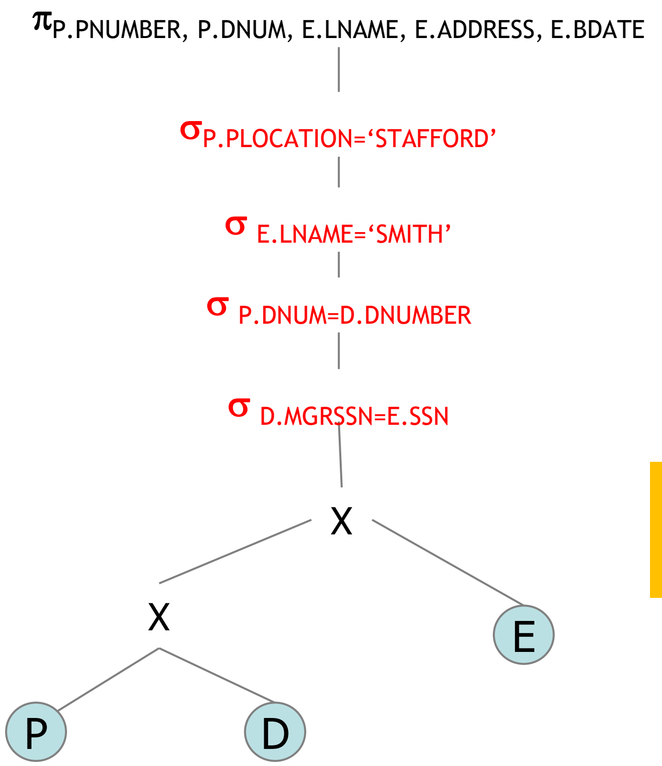
Example



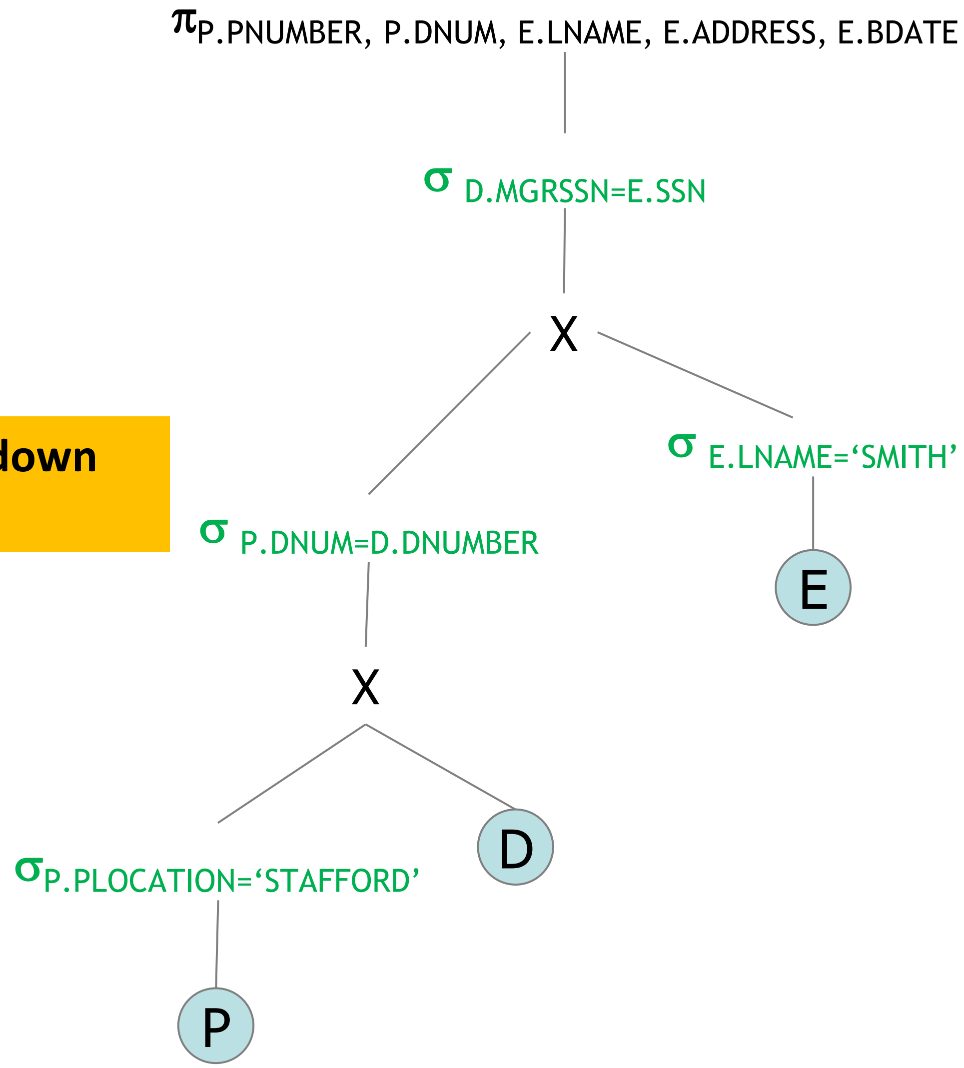
STEP 1: Break up any select operations with conjunctive conditions into a **cascade of select operations**



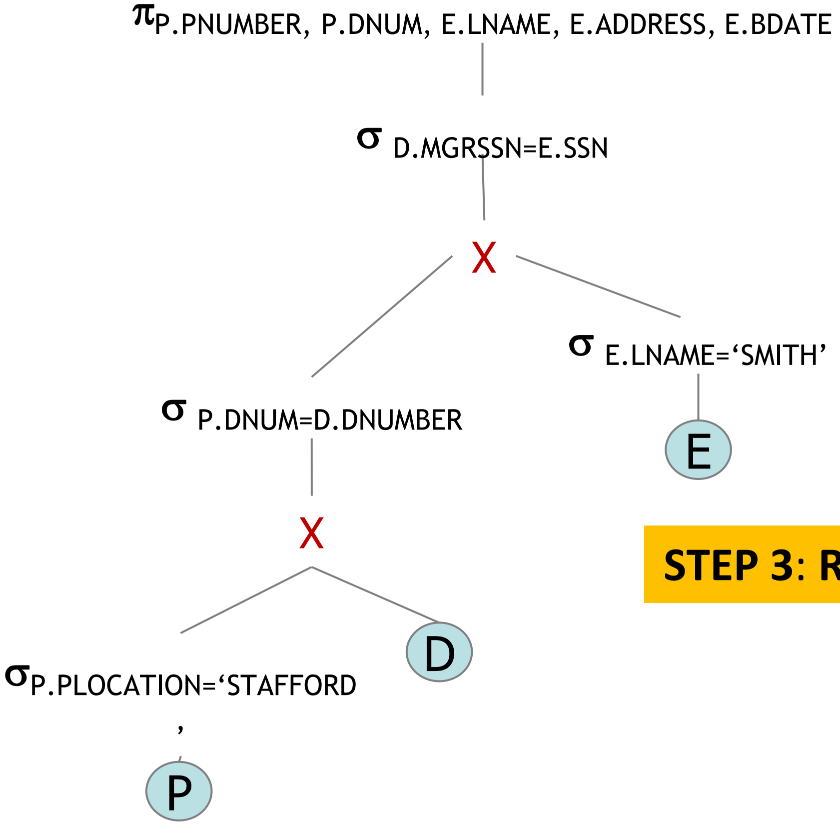
Example



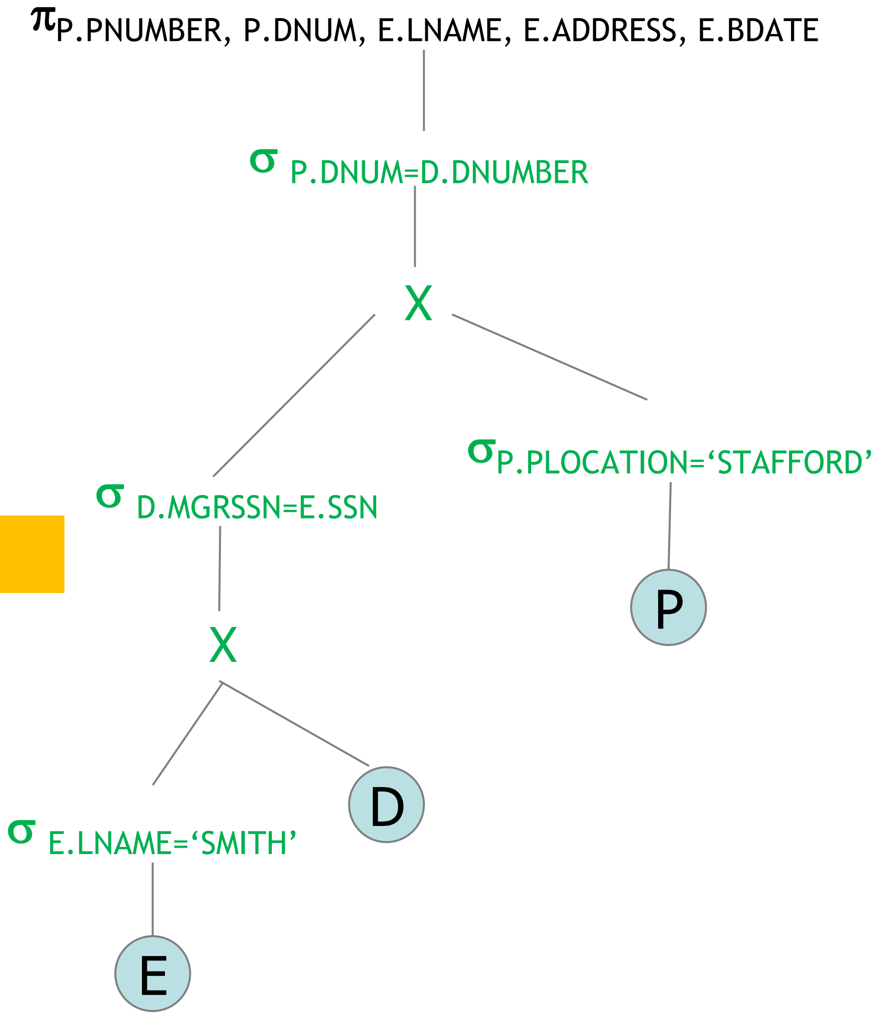
STEP 2: Push SELECT operation as far **down** the query tree as permitted



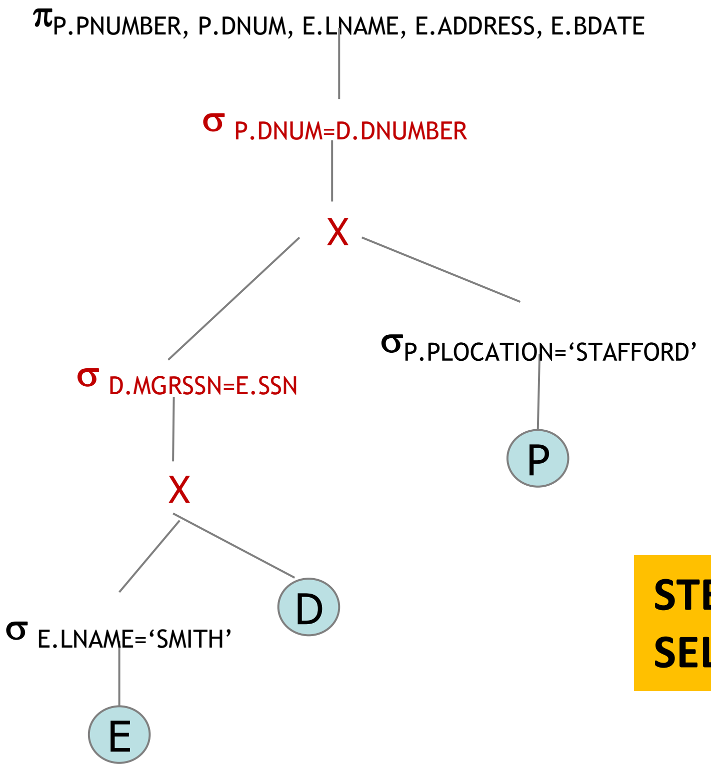
Example



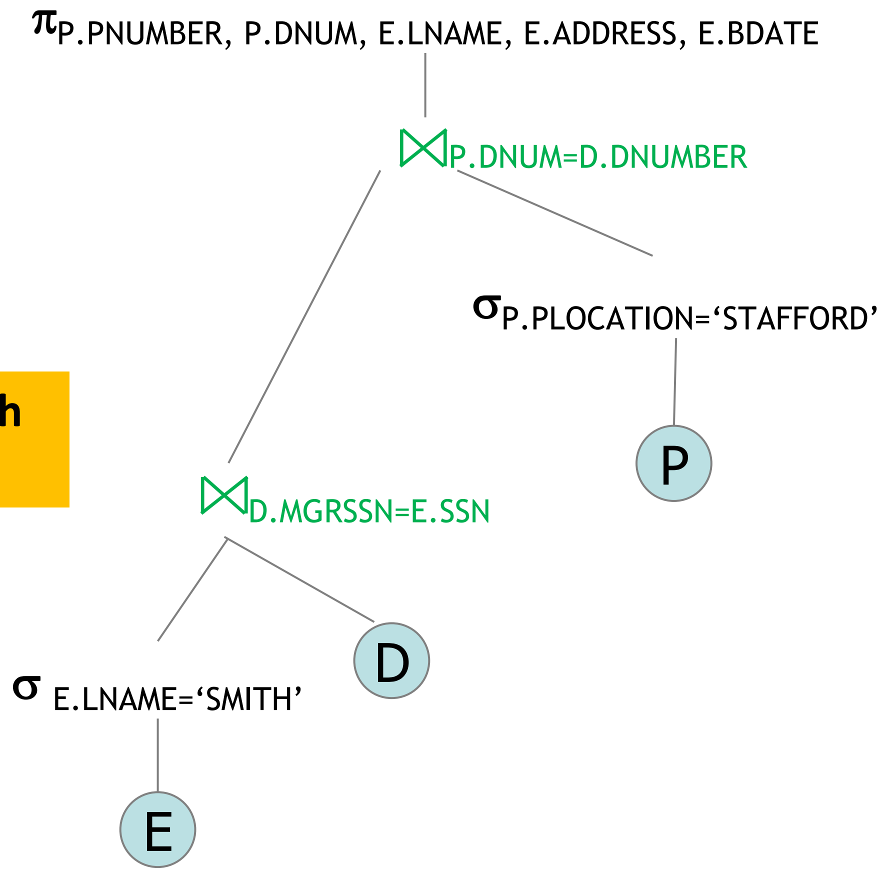
STEP 3: Rearrange binary operations



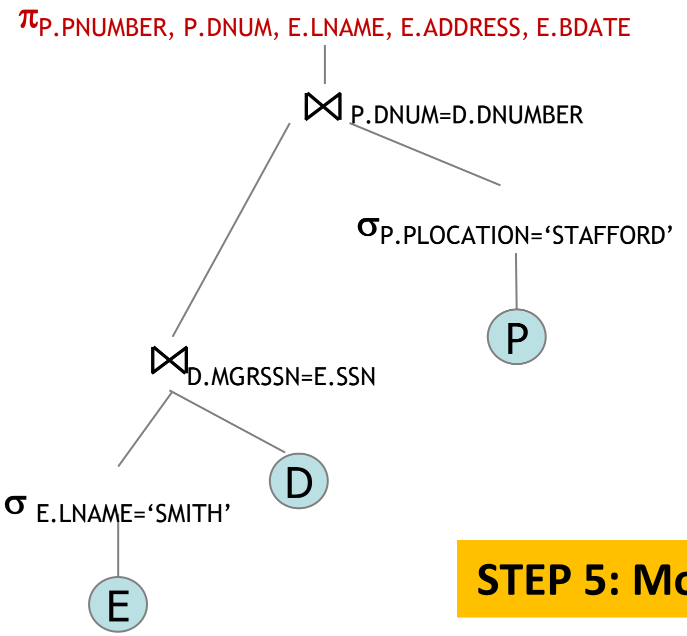
Example



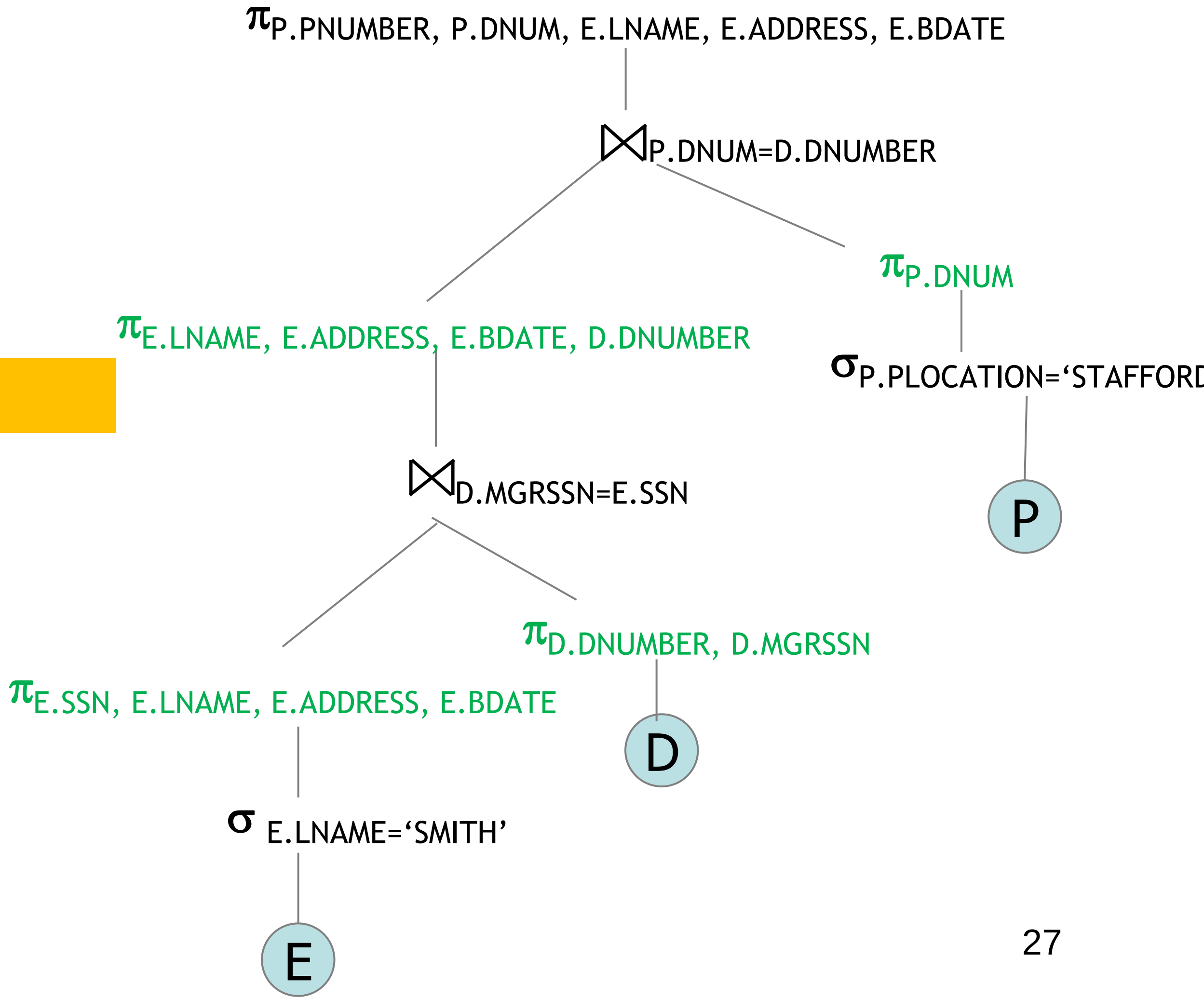
STEP 4: Combine CARTESIAN PRODUCT with SELECT operation into a JOIN operation



Example



STEP 5: Move PROJECT operation down



Let us assume that we have these tables:
PROJECT(Pnumber, Plocation, Dnum, PStartDate)
DEPARTMENT(Dnumber, Dname, Mgr_ssn)
EMPLOYEE(SSN, Fname, Lname, Address, Bdate)

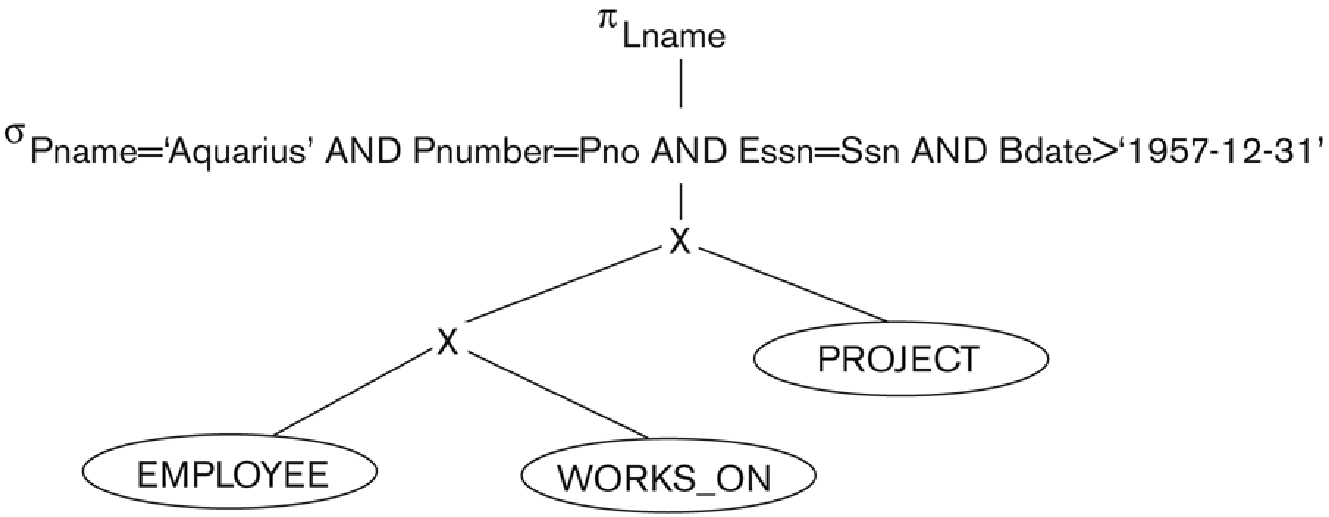
Now your turn

Example

EMPLOYEE(SSN, BDATE, LNAME)
WORKS_ON(ESSN,PNO)
PROJECT (PNUMBER, PNAME)

Query: Find the last names of employees born after 1957 who work on a project named 'Aquarius'

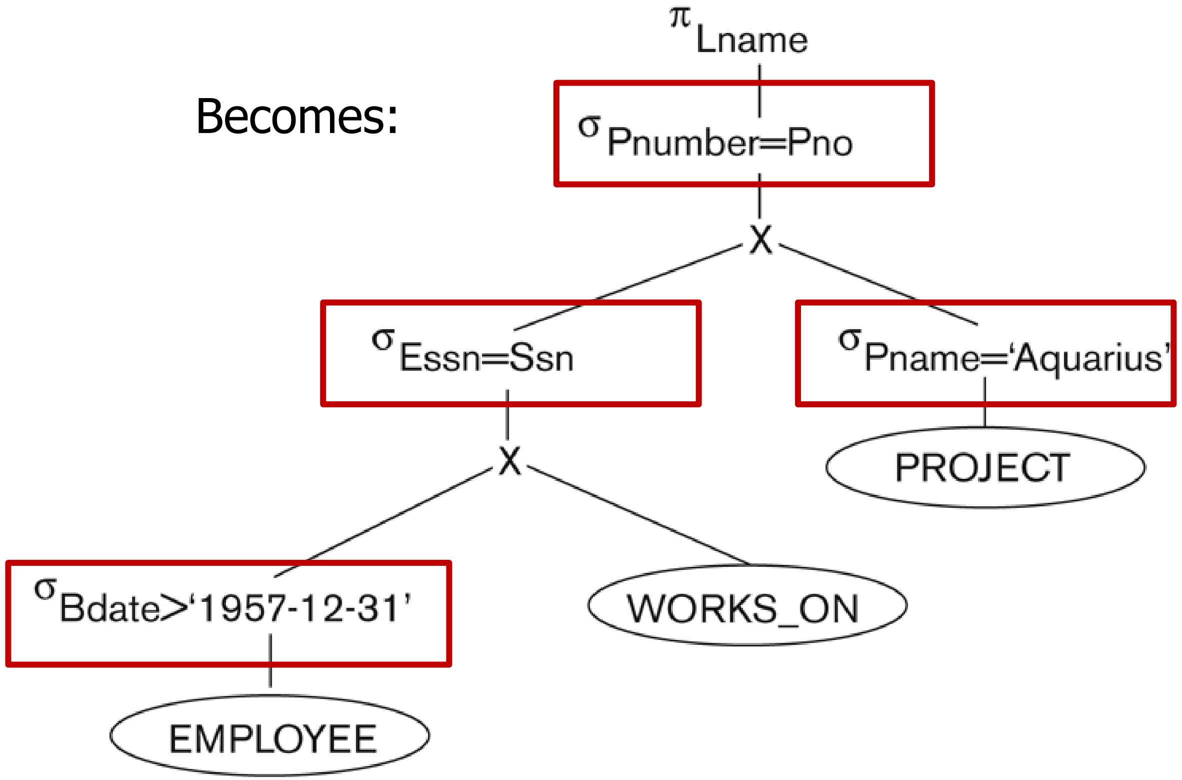
```
SELECT    LNAME
FROM      EMPLOYEE, WORKS_ON, PROJECT
WHERE     PNAME = 'AQUARIUS' AND PNUMBER=PNO
          AND ESSN=SSN AND BDATE > '1957-12-31';
```



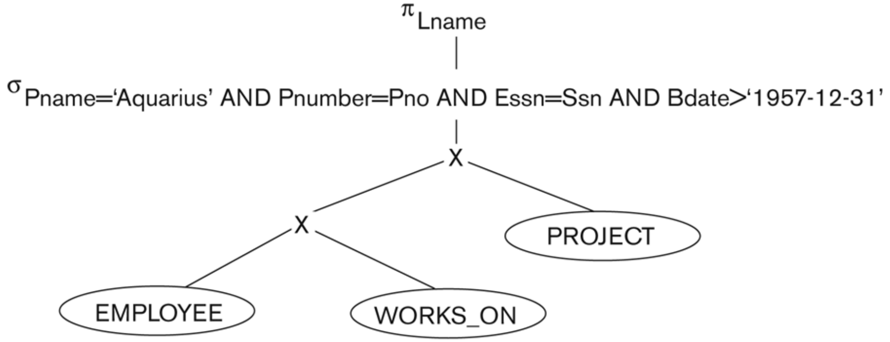
Example

STEP 1: Break up any select operations with conjunctive conditions into a **cascade of select operations**
STEP 2: Move SELECT operations down the query tree

Becomes:



Was:

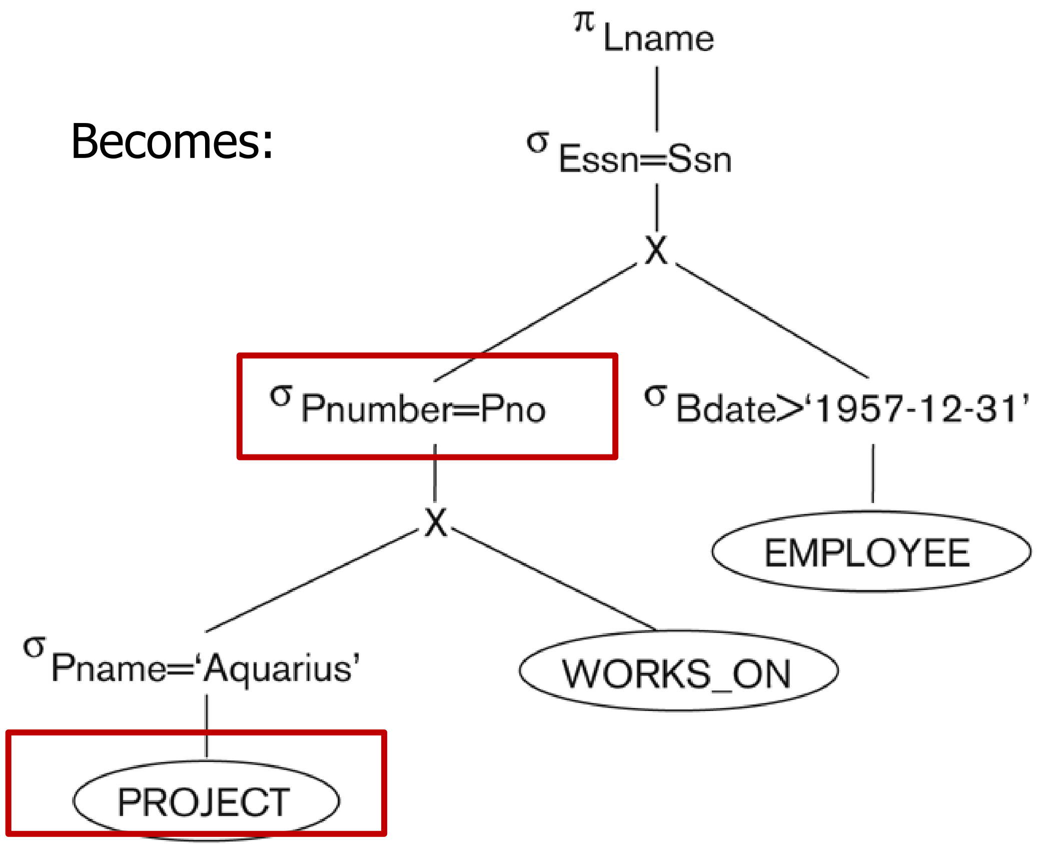


EMPLOYEE(SSN, BDATE, LNAME)
WORKS_ON(ESSN,PNO)
PROJECT (PNUMBER, PNAME)

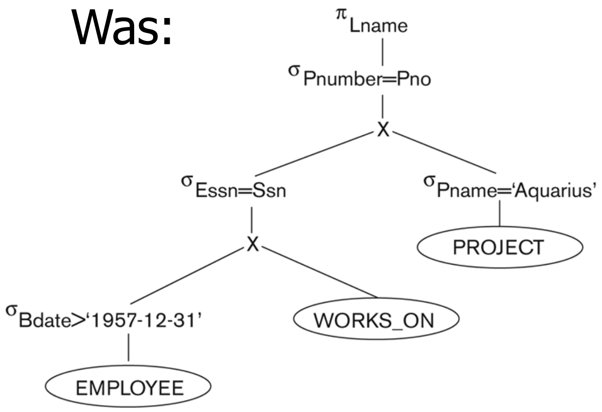
Example

STEP 3: Apply most restrictive SELECT operation first
Rearrange the binary operations

Becomes:



Was:

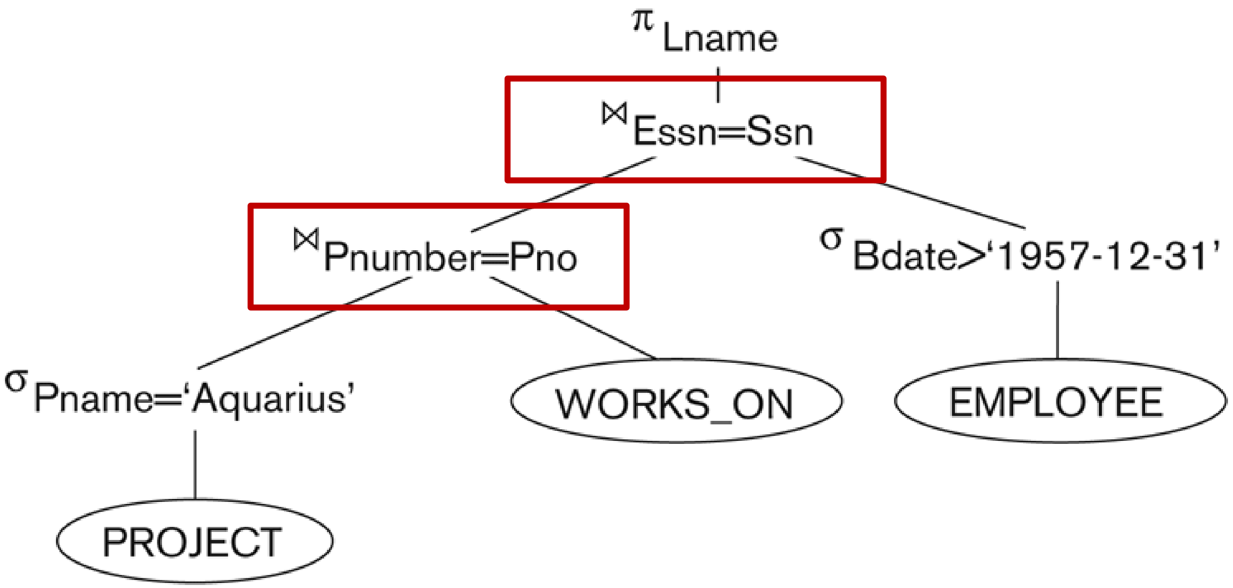


EMPLOYEE(SSN, BDATE, LNAME)
WORKS_ON(ESSN,PNO)
PROJECT (PNUMBER, PNAME)

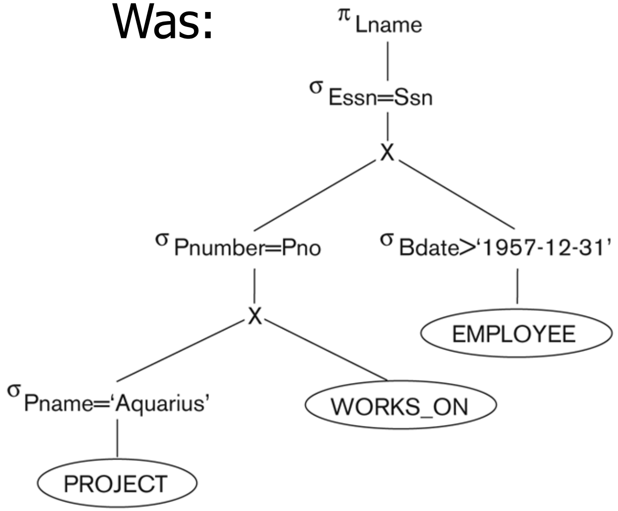
Example

STEP 4: Replace Cartesian Product and Select with Join operations

Becomes:



Was:

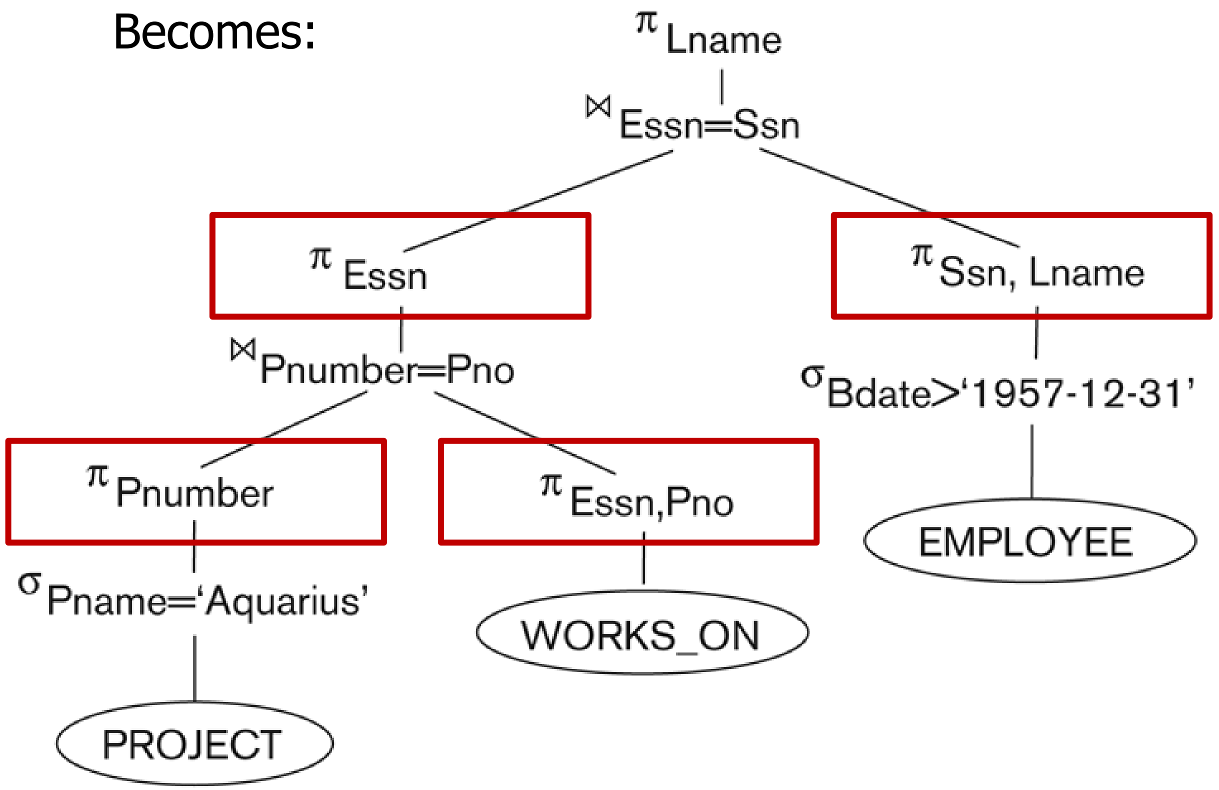


EMPLOYEE(SSN, BDATE, LNAME)
WORKS_ON(ESSN,PNO)
PROJECT (PNUMBER, PNAME)

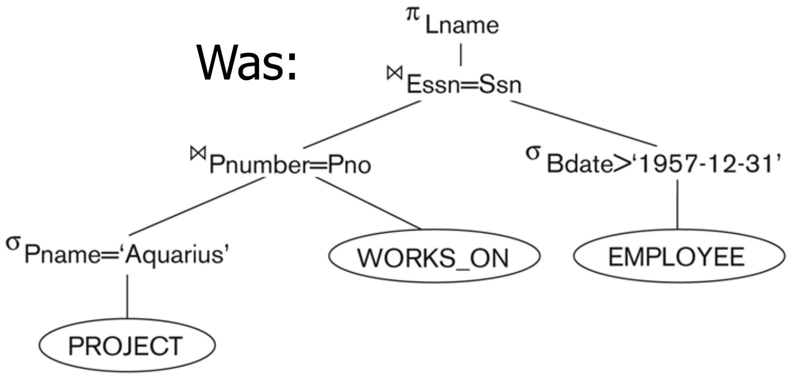
Example

STEP 5: Moving PROJECT operations down the query tree

Becomes:



Was:



EMPLOYEE(SSN, BDATE, LNAME)
WORKS_ON(ESSN,PNO)
PROJECT (PNUMBER, PNAME)

A note on pipelining …

Query Processing

A query is mapped into a sequence of operations

SELECT E.Fname, E.Lname, W.Pno
FROM Employee E, Works_on W
WHERE E.Salary > 50000 AND W.Hours > 40 AND E.SSN = W.ESSN

$$\pi_{E.Fname, E.Lname, W.Pno}\left(\sigma_{E.Salary>50000}\left(Employee\right) \bowtie_{E.SSN=W.ESSN} \sigma_{W.Hours>40}\left(Works\_on\right)\right)$$

Query Processing

- Using temporary tables
  1. Temp1 $\leftarrow \sigma_{E.Salary>50000}$ (Employee)
  2. Temp2 $\leftarrow \sigma_{W.Hours>40}$ (Works_on)
  3. Temp3 $\leftarrow$ Temp1 $\bowtie_{E.SSN=W.ESSN}$ Temp2
  4. Result $\leftarrow \pi_{E.Fname,E.Lname,W.Pno}$ (Temp3)

Each execution of an operation produces a **temporary result**
Generating and saving temporary files on disk is **time consuming and expensive**

Combining Operations using Pipelining

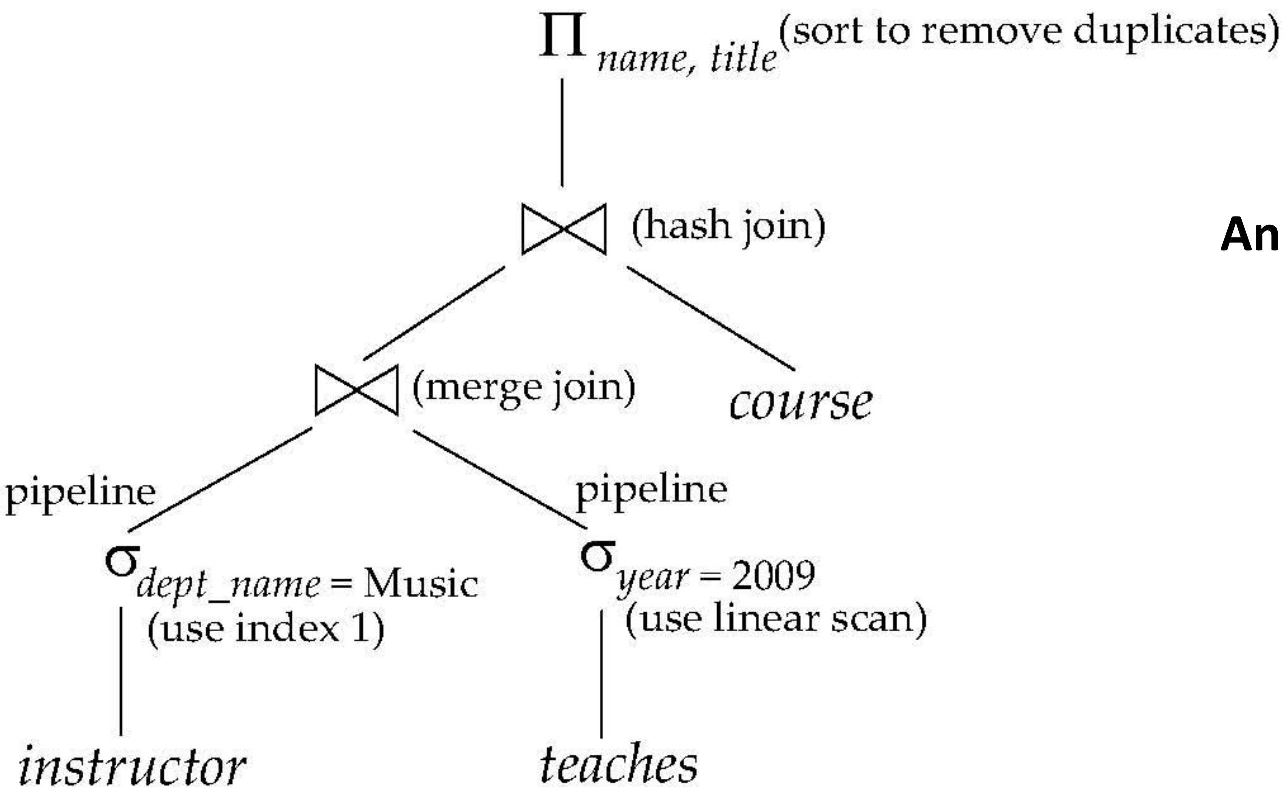
- Without pipelining:
  1. Temp1 $\leftarrow \sigma_{E.Salary>50000}$ (Employee)
  2. Temp2 $\leftarrow \sigma_{W.Hours>40}$ (Works_on)
  3. Temp3 $\leftarrow$ Temp1 $\bowtie_{E.SSN=W.ESSN}$ Temp2
  4. Result $\leftarrow \pi_{E.Fname,E.Lname,W.Pno}$ (Temp3)

- Pipelining:
  - Avoid constructing temporary tables as much as possible.
  - Pass the result of a previous operator to the next without waiting to complete the previous operation.

- Pipelining: interleave the operations in steps 3 and 4

Query Optimization: Phase 2. Physical

Physical Optimization



An execution plan consists of a combination of

- The relational algebra query tree and
- Information about how to compute the relational operators in the tree based on the access paths and algorithms available

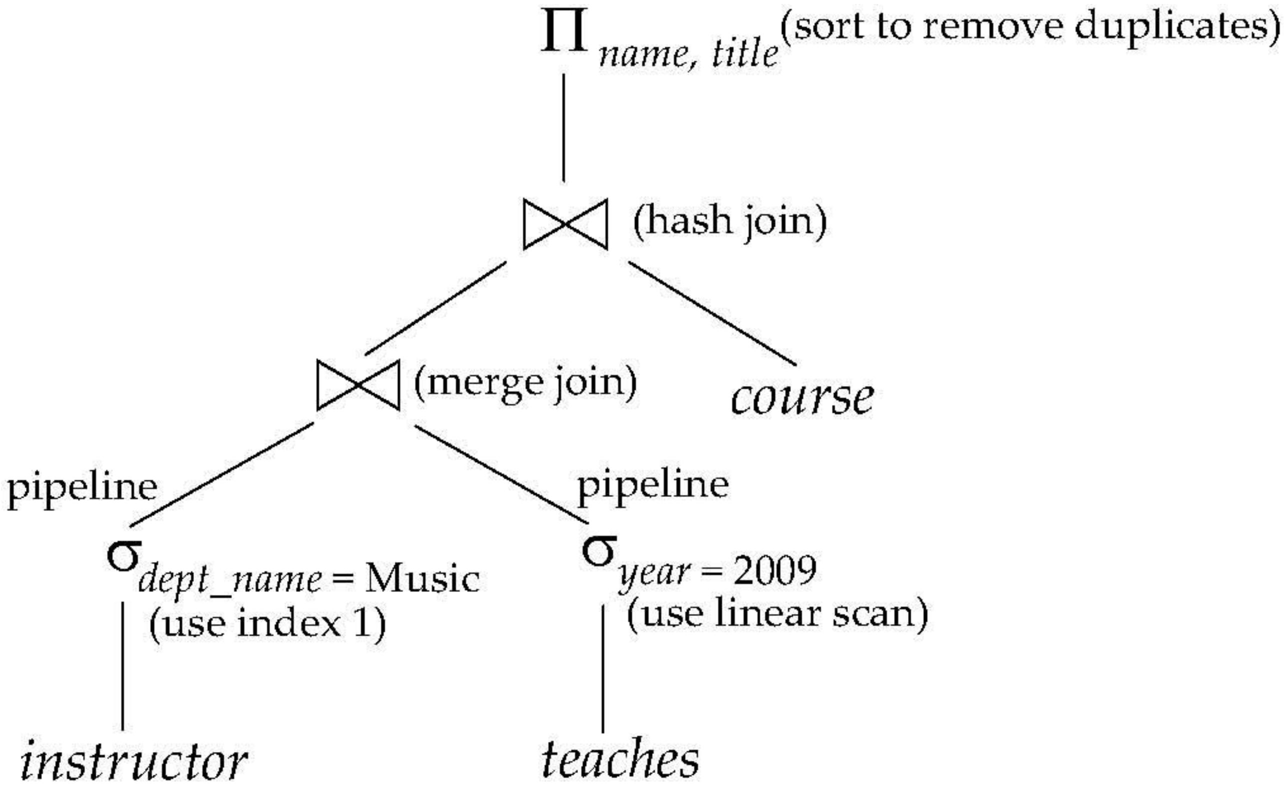
Query Optimizer

- What it needs:
  1. Information about how to compute the relational operators in the tree
     - Based on the access paths and algorithms available
  2. Information about the data stored
     - System Catalog Information
  3. Formulas to compute costs and cardinalities
  4. Strategy to generate plans and select the one to be executed

Query Optimizer

- What it needs:
  1. **Information about how to compute the relational operators in the tree**
     - **Based on the access paths and algorithms available**
  2. Information about the data stored
     - System Catalog Information
  3. Formulas to compute costs and cardinalities
  4. Strategy to generate plans and select the one to be executed

1. How to compute the relational operators



Access paths: how to access a relation

- Indexes
- Full scan

Join methods: how to execute a join

- Nested loop join
- Index loop join
- Merge Join
- …

What about the other operators?

Algorithms for PROJECT

PROJECT operations : $\Pi_{\text{<attribute list>}}(R)$

- If <attribute list> **has a key of relation R**,
  extract all tuples from R with only the values for the attributes in <attribute list>.

- If <attribute list> **does NOT include a key of relation R**,
  duplicated tuples must be removed from the results.

  Methods to remove duplicate tuples

  1. Sorting
  2. Hashing

Algorithms for SET Operations

CARTESIAN PRODUCT of relations R and S include all possible combinations of records from R and S. The attributes of the result include all attributes of R and S.

Cost analysis of CARTESIAN PRODUCT

If R has n records and j attributes and S has m records and k attributes, the result relation will have n*m records and j+k attributes.

The CARTESIAN PRODUCT operation is **very expensive** and should be avoided if possible.

Algorithms for SET Operations

UNION

- Sort the two relations on the same attributes.
- Scan and merge both sorted files concurrently
- Whenever the same tuple exists in both relations, only one is kept in the merged results.

INTERSECTION

- Sort the two relations on the same attributes.
- Scan and merge both sorted files concurrently
- Keep in the merged results only those tuples that appear in both relations.

SET DIFFERENCE R-S

- Keep in the merged results only those tuples that appear in relation R but not in relation S.

Algorithms for Aggregate Operations

Aggregate operators:

MIN, MAX, SUM, COUNT and **AVG**

Options to implement aggregate operators:

Table Scan

Index

Which one to use when?

Algorithms for Aggregate Operations

Example

```
SELECT    MAX (SALARY)
FROM      EMPLOYEE;
```

If an (ascending) index on SALARY exists for the employee relation,

the optimizer could decide on traversing the index for the largest value,

which would entail following the right most pointer in each index node from the root to a leaf.

Algorithms for Aggregate Operations

Dense Index

Sparse Index

Algorithms for Aggregate Operations

SUM, COUNT and AVG

For a **dense index** (each record has one index entry):
Apply the associated computation to the values in the index.

For a **non-dense index**:
Actual number of records associated with each index entry must be accounted for

Algorithms for Aggregate Operations

With **GROUP BY**:

The aggregate operator must be applied separately to each group of tuples.

- Use sorting or hashing on the group attributes to partition the file into the appropriate groups;
- Compute the aggregate function for the tuples in each group.

Query Optimizer

- What it needs:
  1. Information about how to compute the relational operators in the tree
     - Based on the access paths and algorithms available
  2. **Information about the data stored**
     - **System Catalog Information**
  3. Formulas to compute costs and cardinalities
  4. Strategy to generate plans and select the one to be executed

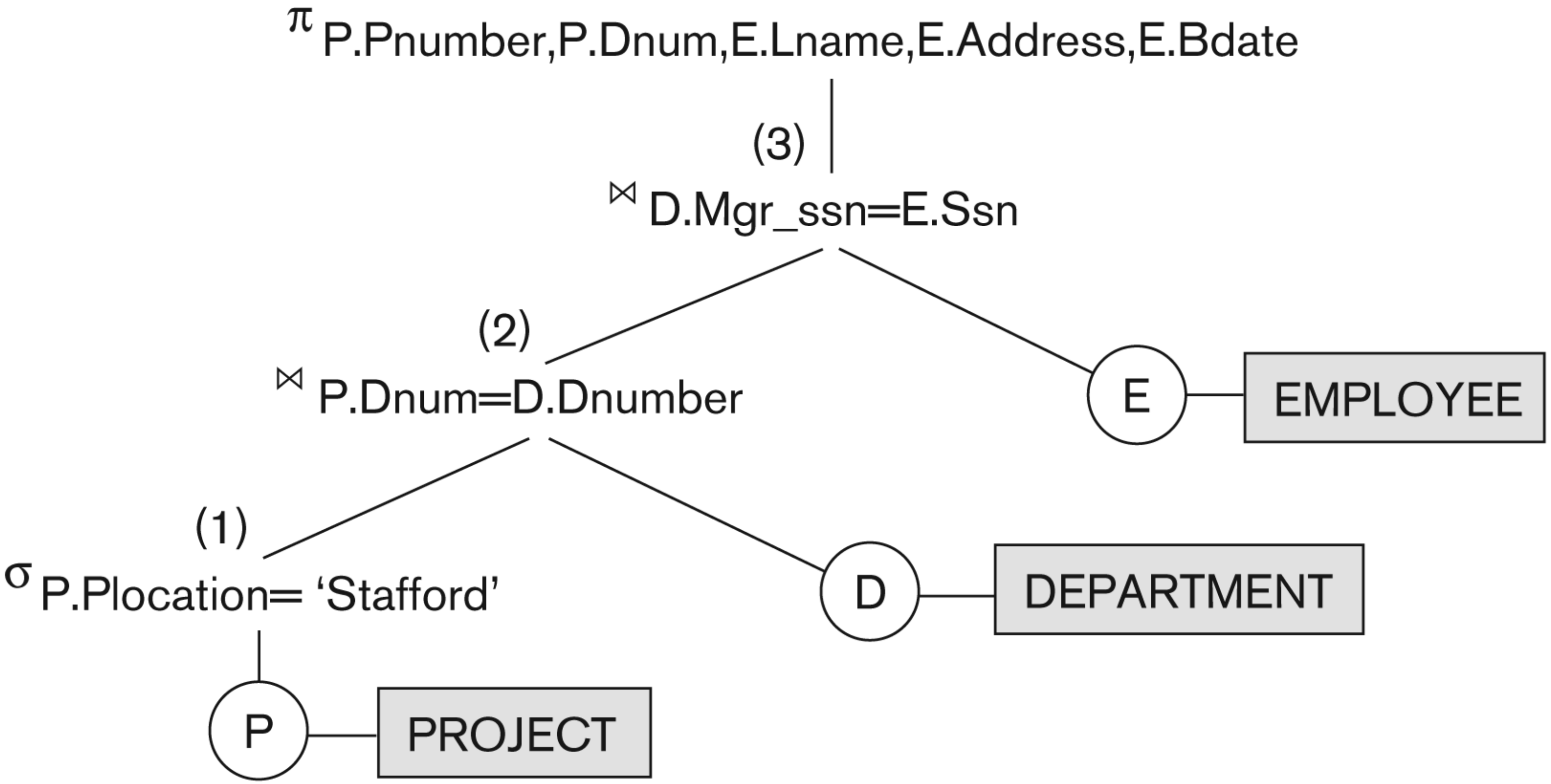
Cost-based Query Optimization

Estimate and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate.

- Example query:

```
SELECT   Pnumber, Dnum, Lname, Address, Bdate
FROM     PROJECT, DEPARTMENT, EMPLOYEE
WHERE    Dnum = Dnumber AND Mgr_ssn = Ssn
                 AND Plocation = 'Stafford'
```

Example



Need to estimate the cost for performing each relational algebra operation using different access paths and query processing methods

System Catalog Information

- Information about the size of a file
  - n_R*:* number of tuples in a relation *R.*
  - b_R: number of blocks containing tuples of *R.*
  - l_R: record size of *R.*
  - bf_R*:* blocking factor of *R* — i.e., the number of tuples of *R* that fit into one block.

- Information about indexes and indexing attributes of a file
  - Number of levels (x) of each multilevel index
  - Number of first-level index blocks (b_{I1})
  - Number of distinct values (d) of an attribute
  - Selectivity (sl) of an attribute

Example (System Catalog)

(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.