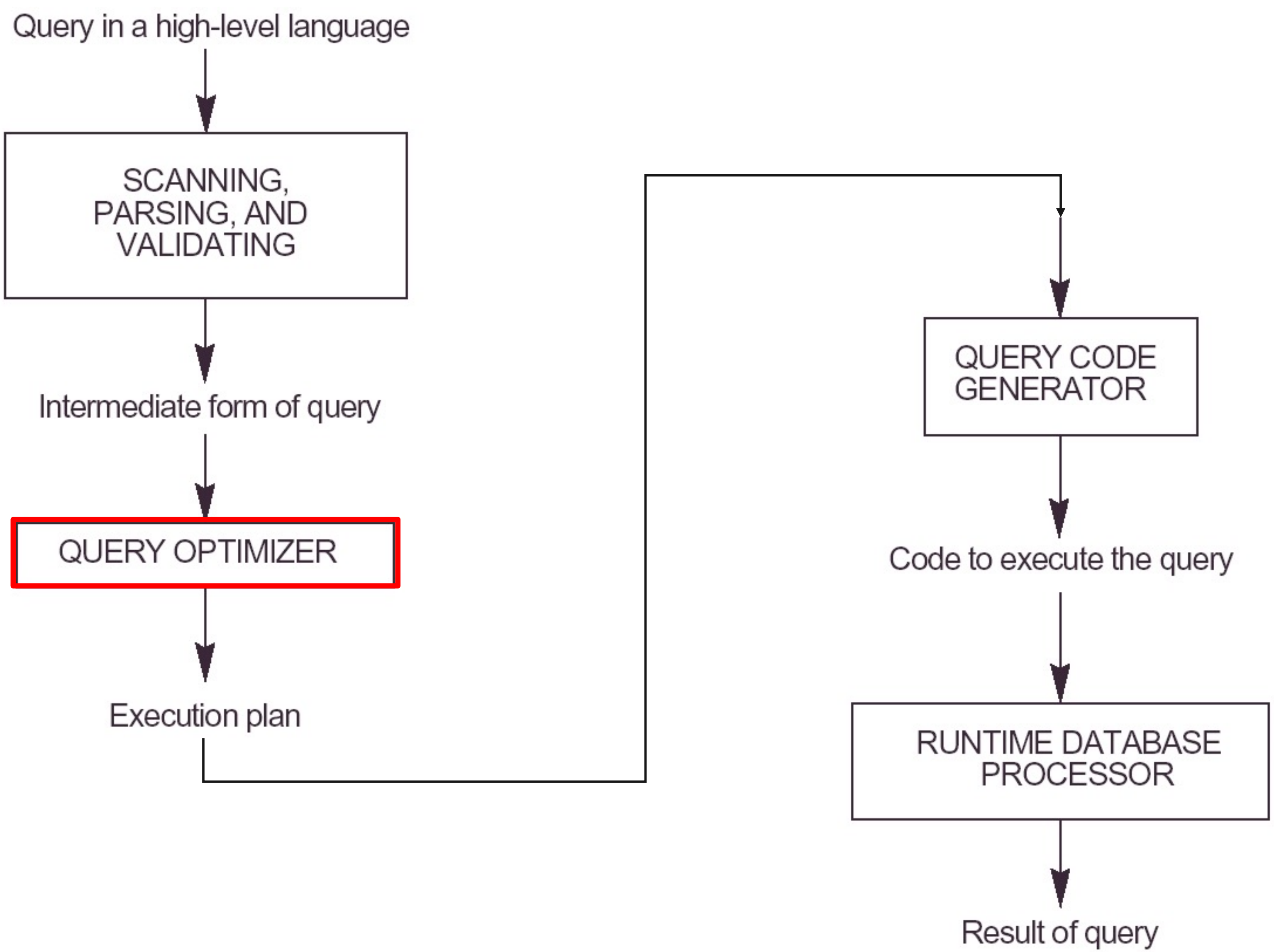
Query Optimization

政治大學
資訊科學系
沈錳坤

Outline

1. Query Optimization
2. Using Heuristics in Query Optimization
3. Using Selectivity & Cost Estimates in Query Optimization
4. Semantic Query Optimization

Query Optimization




There are many routes to achieve the revenue goal

Query Optimization (cont.)

- Query optimization
  - Process of planning a good execution strategy
  - Finding the optimal strategy is usually too time-consuming
  - Sometimes the chosen plan is not the optimal one, but a reasonably efficient strategy for executing the query

- For low level navigational DB languages (e.g. hierarchical DBMS):
  - Programmer is given the capability to choose optimal execution strategy

Query Optimization (cont.)

- Approaches for implementing query optimization
  1. Heuristic rules for ordering operations in a query execution strategy
  2. Systematically estimating the cost of different execution strategies and choosing the execution plan with the lowest estimate
  3. Semantic query optimization

Outline

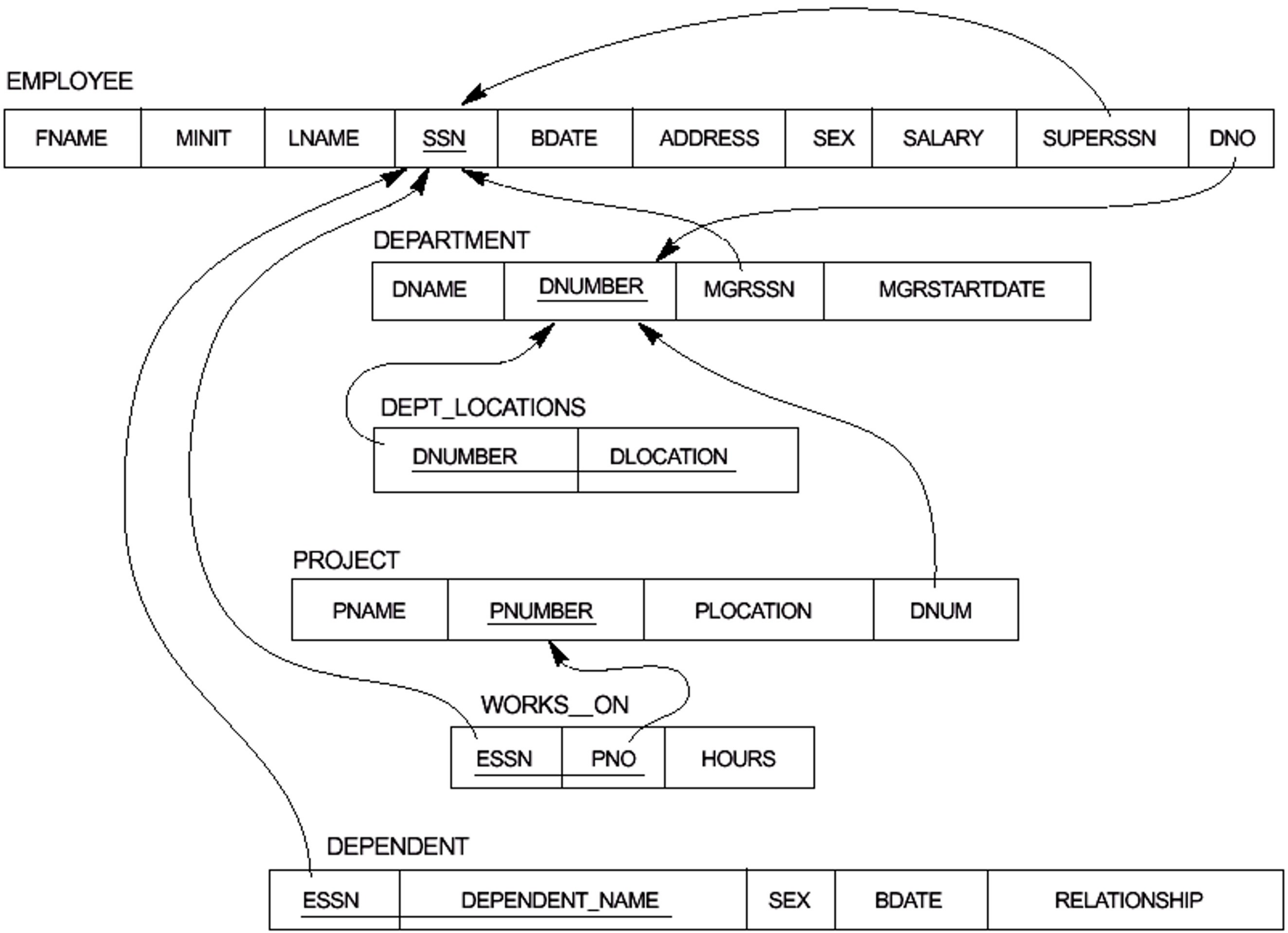
1. Query Optimization
2. Using Heuristics in Query Optimization
3. Using Selectivity & Cost Estimates in Query Optimization
4. Semantic Query Optimization

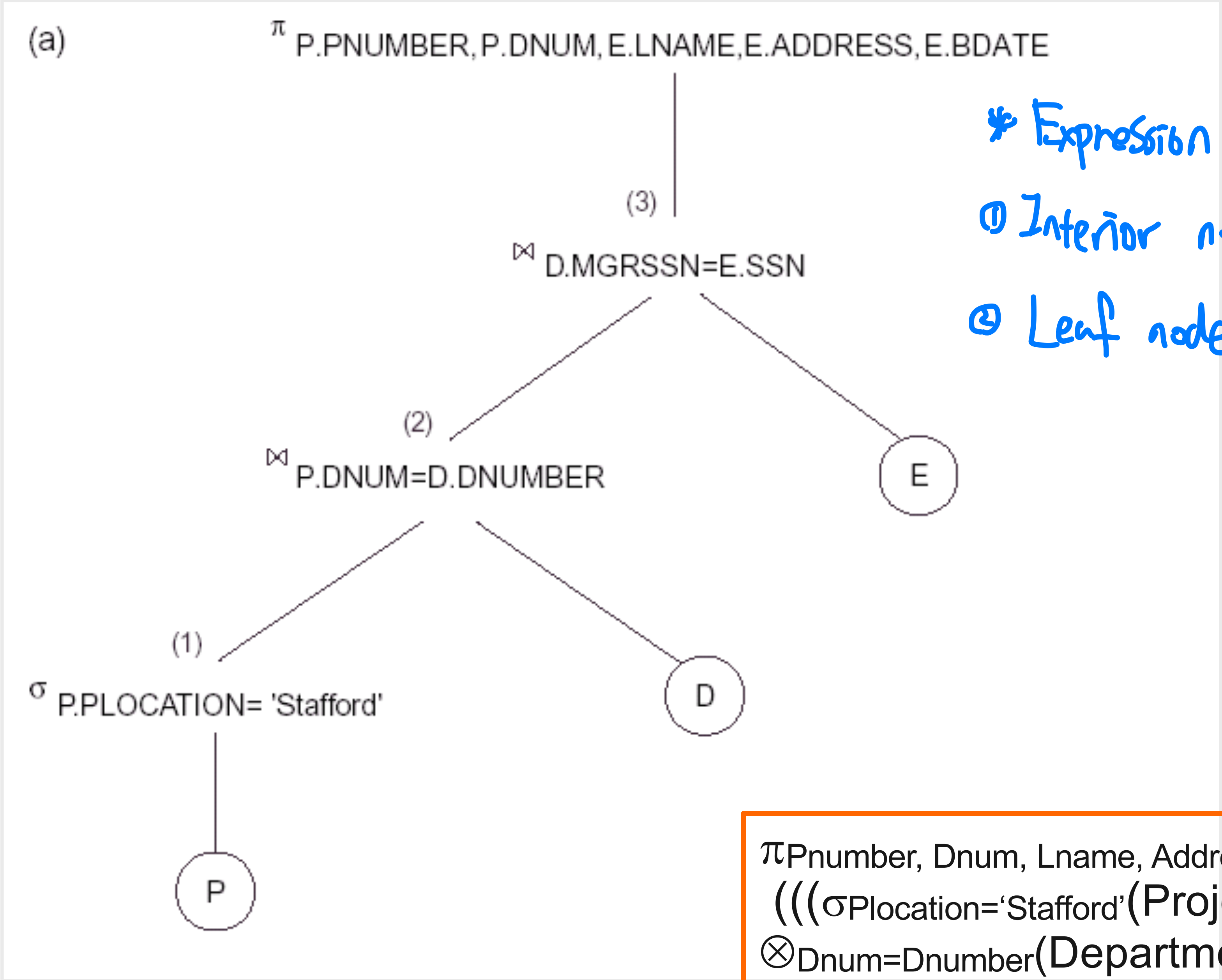
Query Tree

For every project located in 'Stafford', retrieve the project number, the controlling department number, and the department manager's last name, address, and birthdate.

$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\pi_{\text{Plocation='Stafford'}}(\text{project})) \otimes_{\text{Dnum=Dnumber}}(\text{Department})) \otimes_{\text{Mgrssn=SSN}}(\text{Employee}))$

Select P.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate
From Project as P, Department as D, Employee as E
Where P.Dnum=D.Dnumber and D.Mgrssn=E.SSN and P.Plocation='Stafford';


EMPLOYEE
FNAME
MINIT
LNAME
SSN
BDATE
ADDRESS
SEX
SALARY
SUPERSSN
DNO
DEPARTMENT
DNAME
DNUMBER
MGRSSN
MGRSTARTDATE
DEPT_LOCATIONS
DNUMBER
DLOCATION
PROJECT
PNAME
PNUMBER
PLOCATION
DNUM
WORKS__ON
ESSN
PNO
HOURS
DEPENDENT
ESSN
DEPENDENT_NAME
SEX
BDATE
RELATIONSHIP



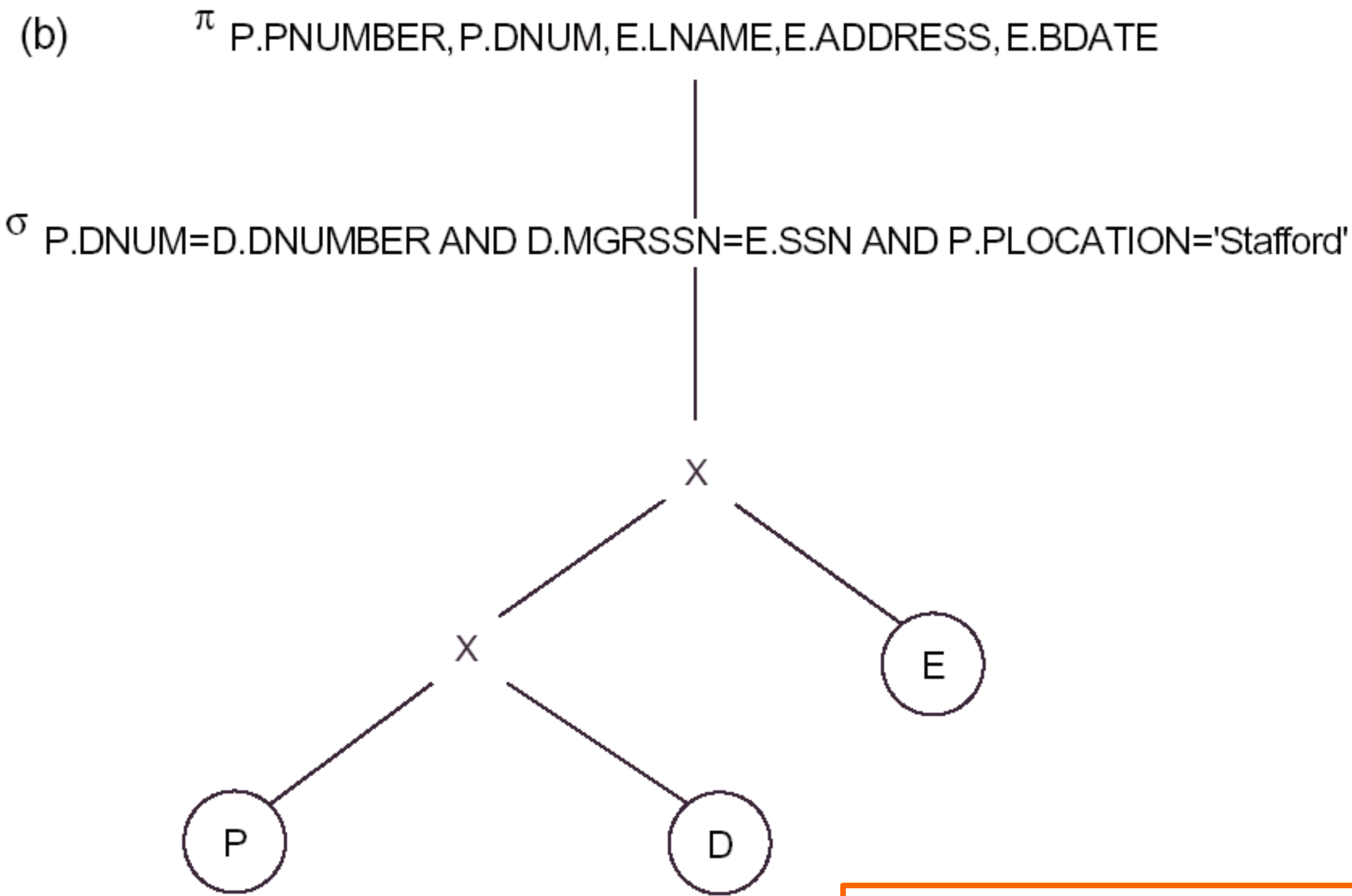
* Expression Tree:

① Interior nodes: operators

② Leaf nodes: operands

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}\ (((\sigma_{\text{Plocation='Stafford'}}(\text{Project})) \otimes_{\text{Dnum=Dnumber}}(\text{Department})) \otimes_{\text{Mgrssn=SSN}}(\text{Employee}))$$

(b)



$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}\ (((\sigma_{\text{Plocation='Stafford'}}(\text{Project})) \otimes_{\text{Dnum=Dnumber}}(\text{Department})) \otimes_{\text{Mgrssn=SSN}}(\text{Employee}))$$

Heuristic Rules

- Heuristic rule for query optimization
  - σ π
  - Apply selection & projection operations before JOIN or other binary operations

Heuristic Optimization of Query Trees

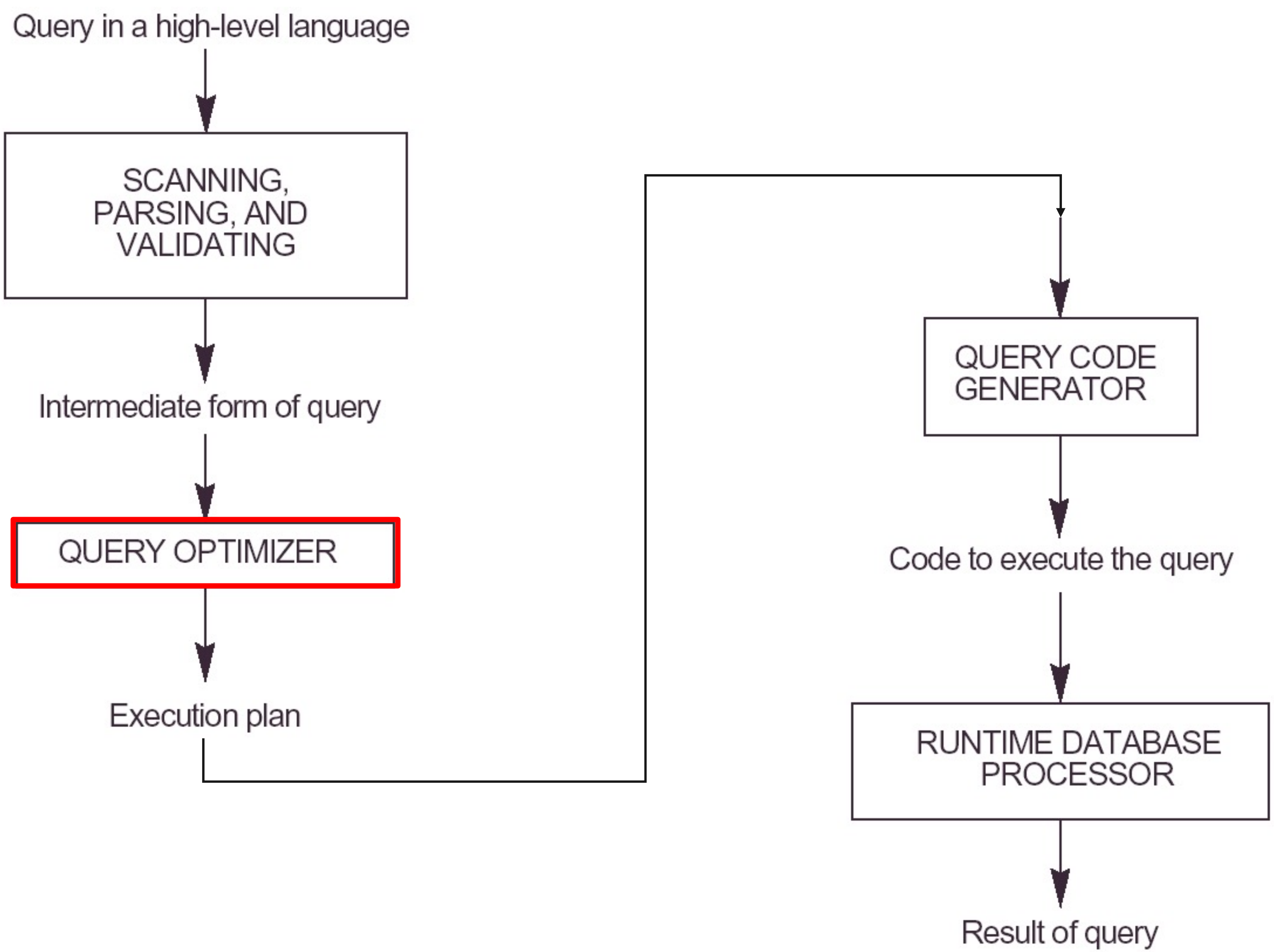
- Query parser generates standard initial query tree to correspond to an SQL query, without doing any optimization
- Different relational algebra expressions (different query trees) correspond to the same query
- Heuristic query optimizer
  - transform the initial query tree into a final query tree that is efficient to execute
  - Include rules for equivalence among relational algebra expressions that can be applied to the initial tree
  - Utilize these equivalence expressions to transform the initial tree into the final, optimized query tree

Example of Transforming a Query

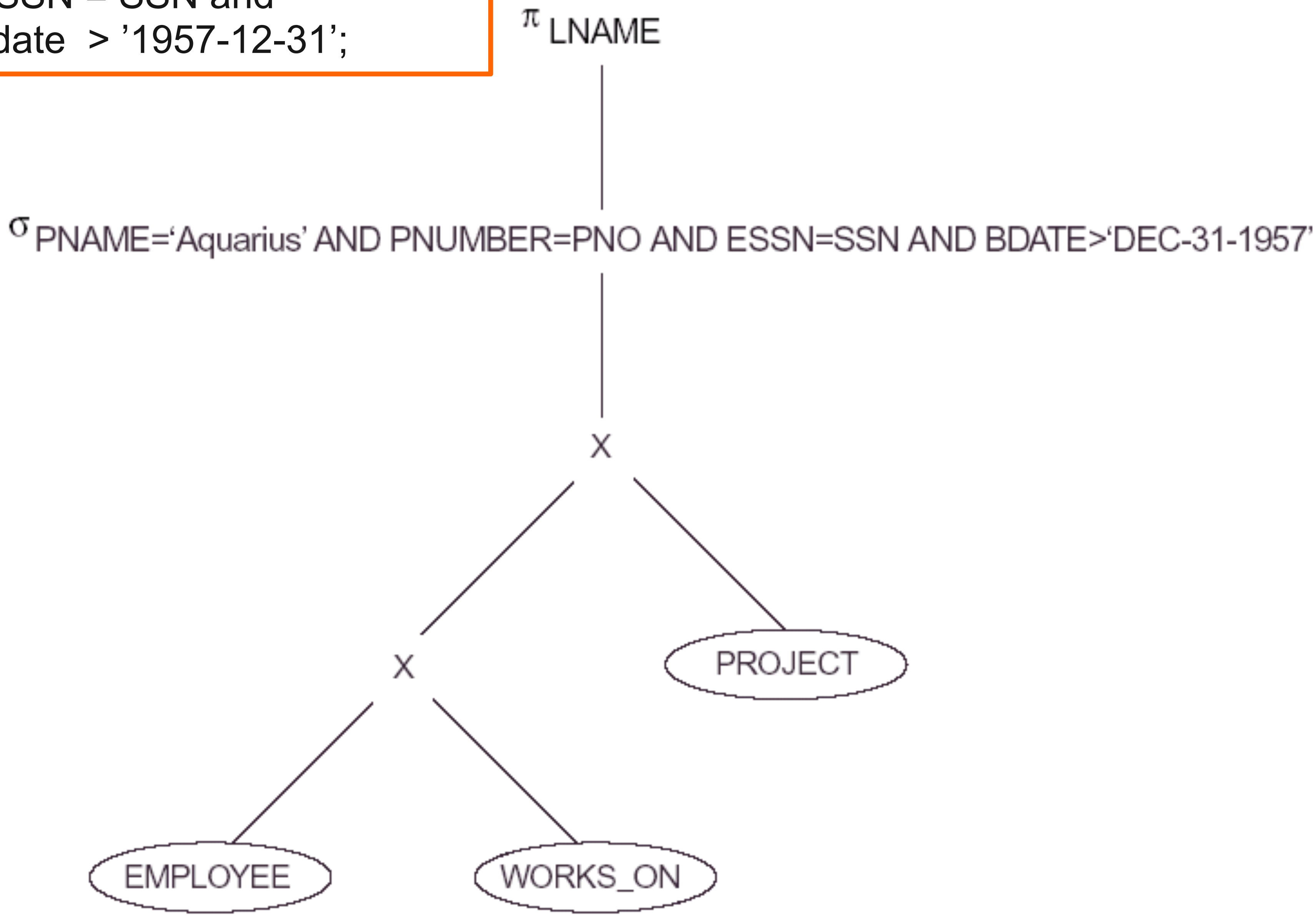
- Find the last names of employees born after 1957 who work on a project named ‘Aquarius’

```
SELECT Lname
FROM Employee, Works_on, Project
WHERE Pname = ‘Aquarius’ and
          Pnumber = Pno and
          ESSN = SSN and
          Bdate > ‘1957-12-31’;
```

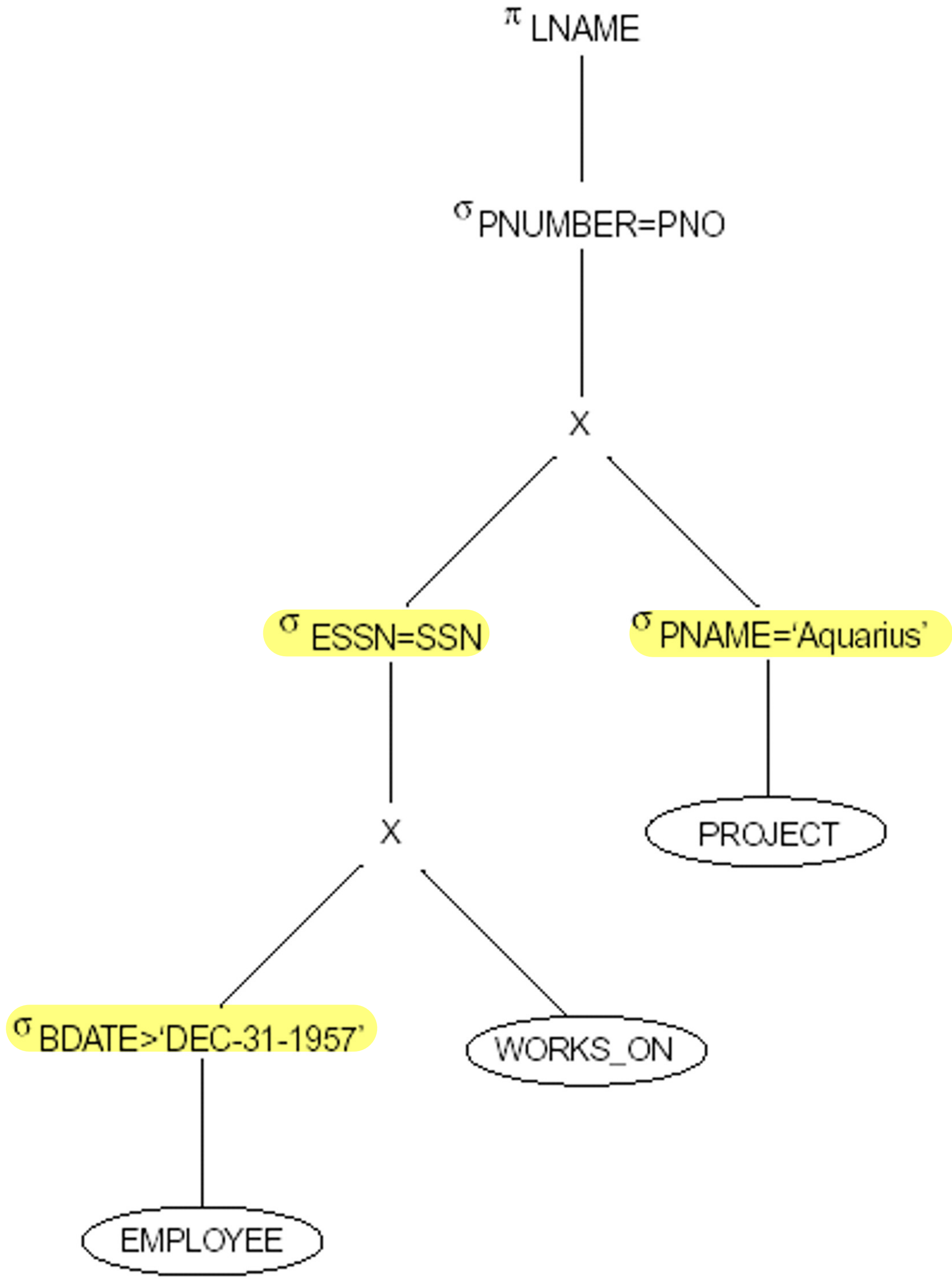
Query Optimization



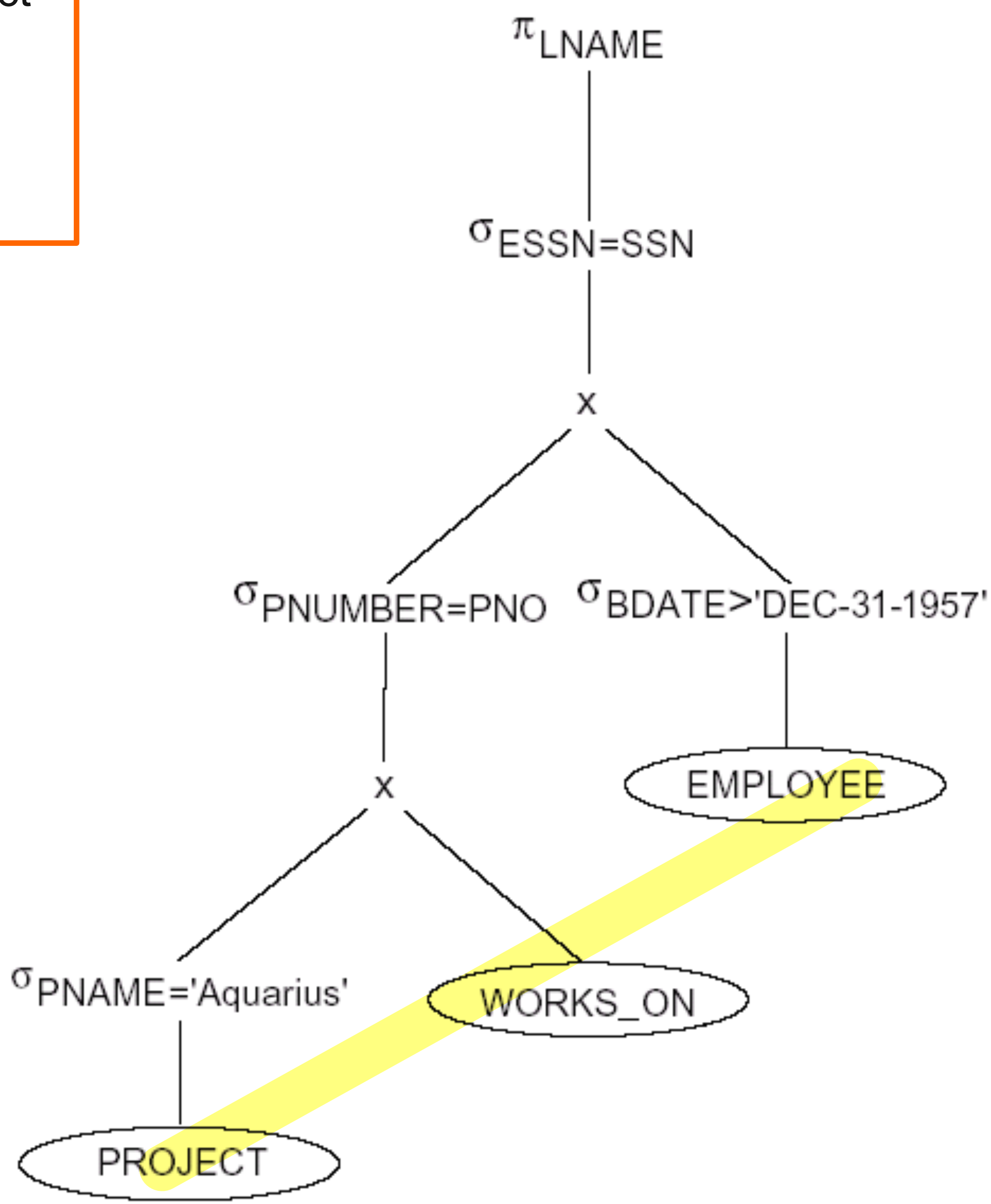
```
SELECT Lname
FROM Employee, Works_on, Project
WHERE Pname = 'Aquarius' and
          Pnumber = Pno and
          ESSN = SSN and
          Bdate  > '1957-12-31';
```



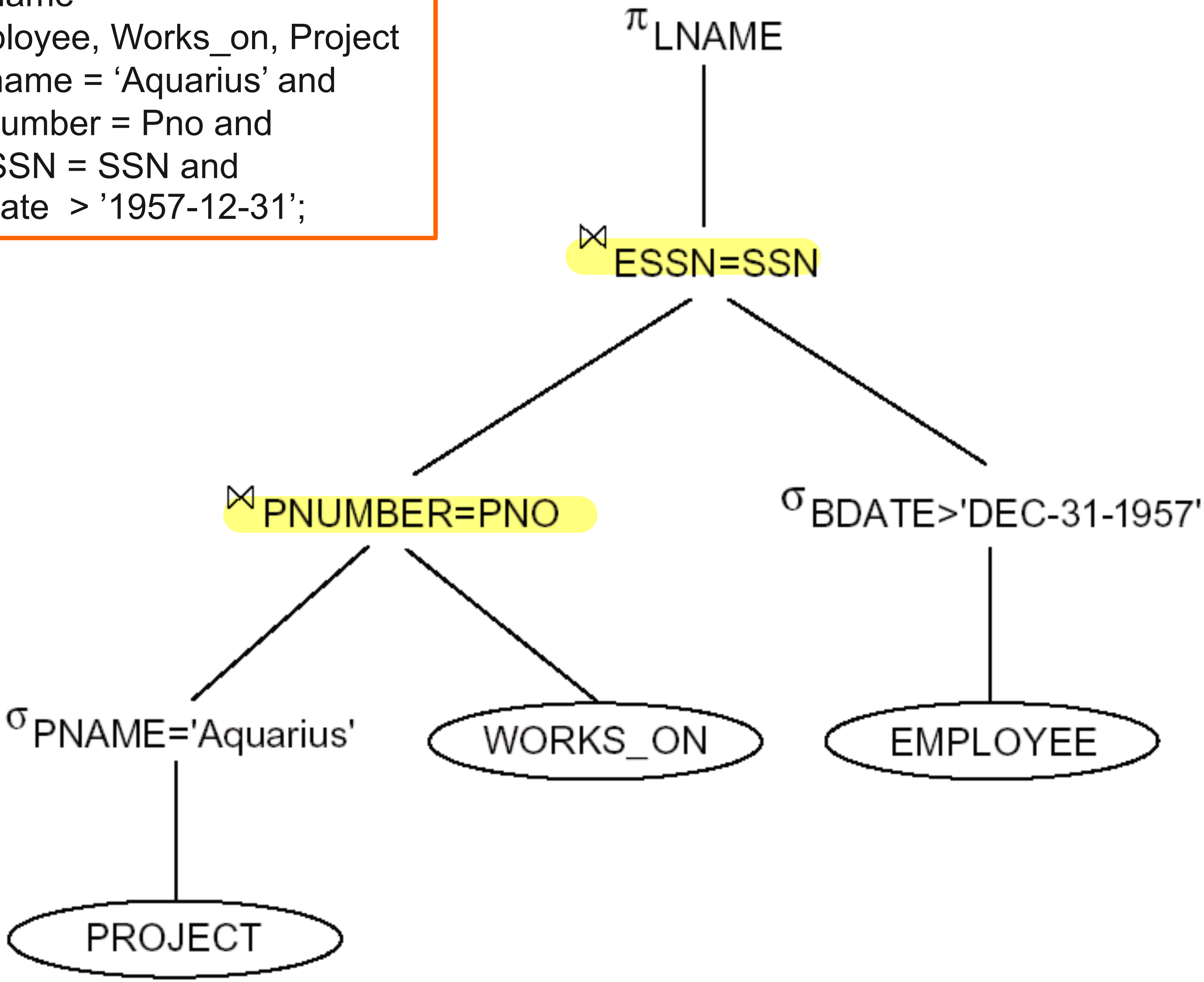
```
SELECT Lname
FROM Employee, Works_on, Project
WHERE Pname = 'Aquarius' and
          Pnumber = Pno and
          ESSN = SSN and
          Bdate  > '1957-12-31';
```



```
SELECT Lname
FROM Employee, Works_on, Project
WHERE Pname = 'Aquarius' and
          Pnumber = Pno and
          ESSN = SSN and
          Bdate  > '1957-12-31';
```

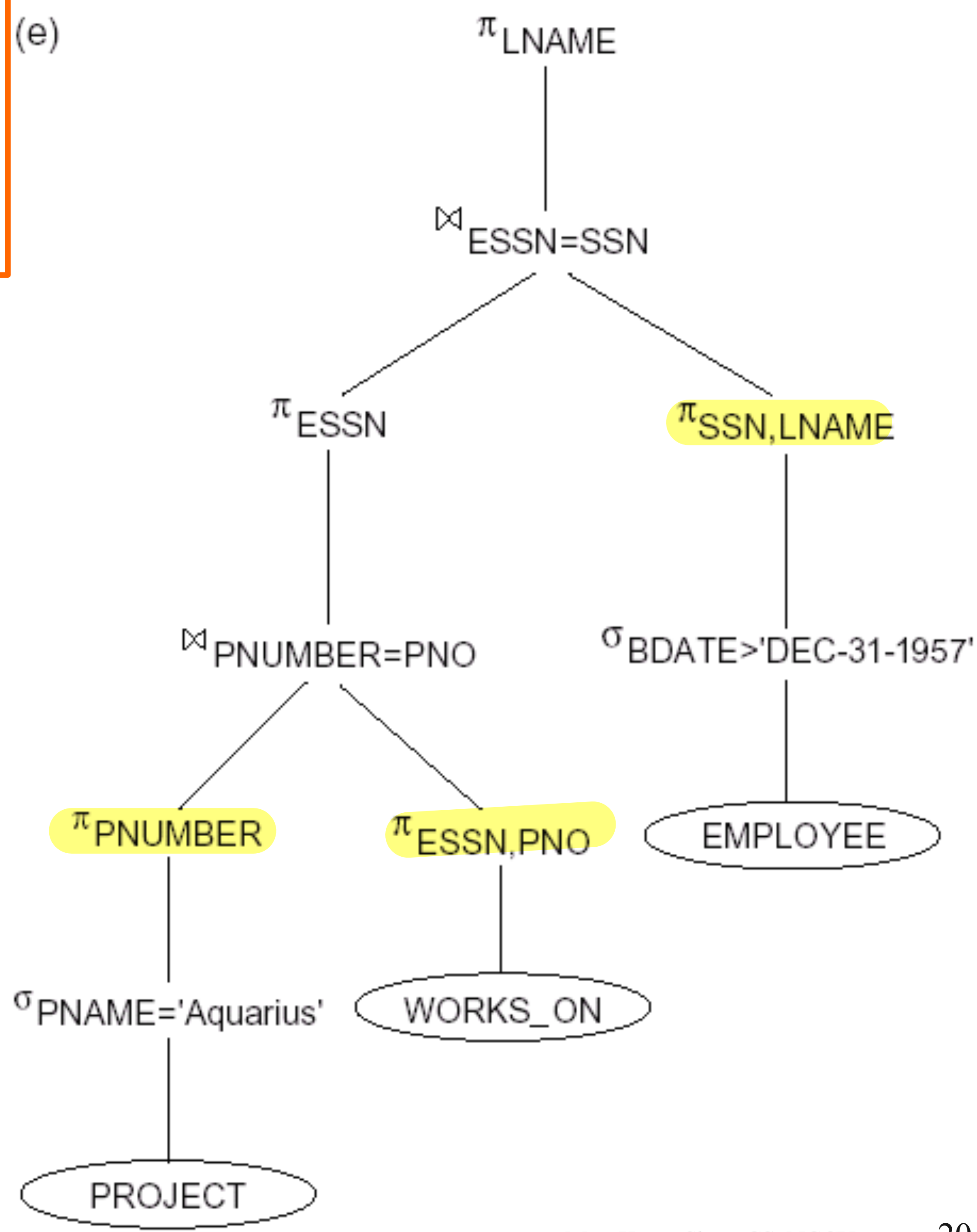


```sql
SELECT Lname
FROM Employee, Works_on, Project
WHERE Pname = 'Aquarius' and
            Pnumber = Pno and
            ESSN = SSN and
            Bdate  > '1957-12-31';
```



```
SELECT Lname
FROM Employee, Works_on, Project
WHERE Pname = 'Aquarius' and
            Pnumber = Pno and
            ESSN = SSN and
            Bdate  > '1957-12-31';
```

(e)



General Transformation Rules

1. Cascade of σ

   $\sigma_{\langle C1\rangle \text{ and } \langle C2\rangle \cdots \text{and } \langle Cn\rangle} (R) = \sigma_{\langle C1\rangle}(\sigma_{\langle C2\rangle}(\ldots(\sigma_{\langle Cn\rangle} (R))\ldots))$

2. Commutativity of σ

   $\sigma_{\langle C1\rangle}(\sigma_{\langle C2\rangle}(R)) = \sigma_{\langle C2\rangle}(\sigma_{\langle C1\rangle}(R))$

3. Cascade of π

   $\pi_{\langle L1\rangle}(\pi_{\langle L2\rangle}(\ldots(\pi_{\langle Ln\rangle} (R))\ldots)) = \pi_{\langle L1\rangle}(R)$

   $L_n \subseteq \cdots \subseteq L_2 \subseteq L_1$

* 4. Commuting σ with π

   $\pi_{\langle A1\rangle \text{ and } \langle A2\rangle \cdots \text{and } \langle An\rangle} (\sigma_{\langle C\rangle} (R)) =$

   $\sigma_{\langle C\rangle} (\pi_{\langle A1\rangle \text{ and } \langle A2\rangle \cdots \text{and } \langle An\rangle} (R))$

* σ: select, π: project

General Transformation Rules

5. Commutativity of ⊗ and X

   $R \otimes S = S \otimes R$ $R \times S = S \times R$

* 6. Commuting σ with ⊗ (or X)

   $\sigma_c (R \otimes S) = (\sigma_c (R)) \otimes S$

   $\sigma_c (R \otimes S) = (\sigma_{c1} (R)) \otimes (\sigma_{c1} (S))$ if c=c1 and c2

* 7. Commuting π with ⊗ (or X)

   $\pi_L (R \otimes S) = (\pi_{A1, A2, \ldots, An} (R)) \otimes (\pi_{B1, B2, \ldots, Bm} (S))$ ($L = A \cup B$)

8. Commutativity of set operations

   ∩, ∪ are commutative, - is not

* ⊗: join, X: Cartesian product

General Transformation Rules (cont.)

9. Associativity of $\otimes$, X, $\cap$, $\cup$

   $(R \Theta S) \Theta T = R \Theta (S \Theta T)$

10. Commuting σ with set operations

   $\sigma_c(R \Theta S) = (\sigma_c(R)) \Theta (\sigma_c(S))$

11. π commutes with $\cup$

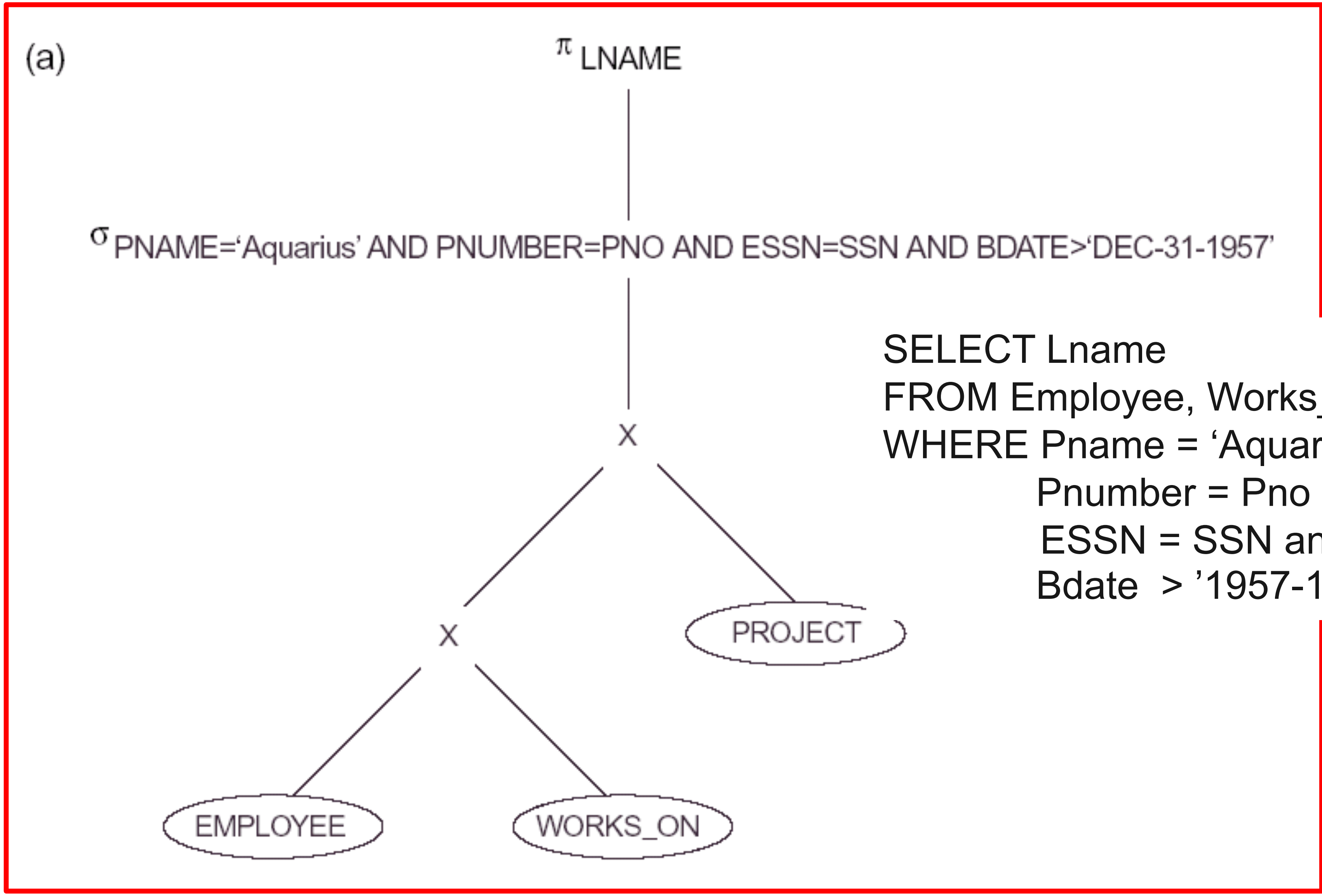
   $\pi_L(R \cup S) = (\pi_L(R)) \cup (\pi_L(S))$

12. $\sigma_c (R \times S) = R \otimes_c S$

Heuristic Algebraic Optimization Algorithm

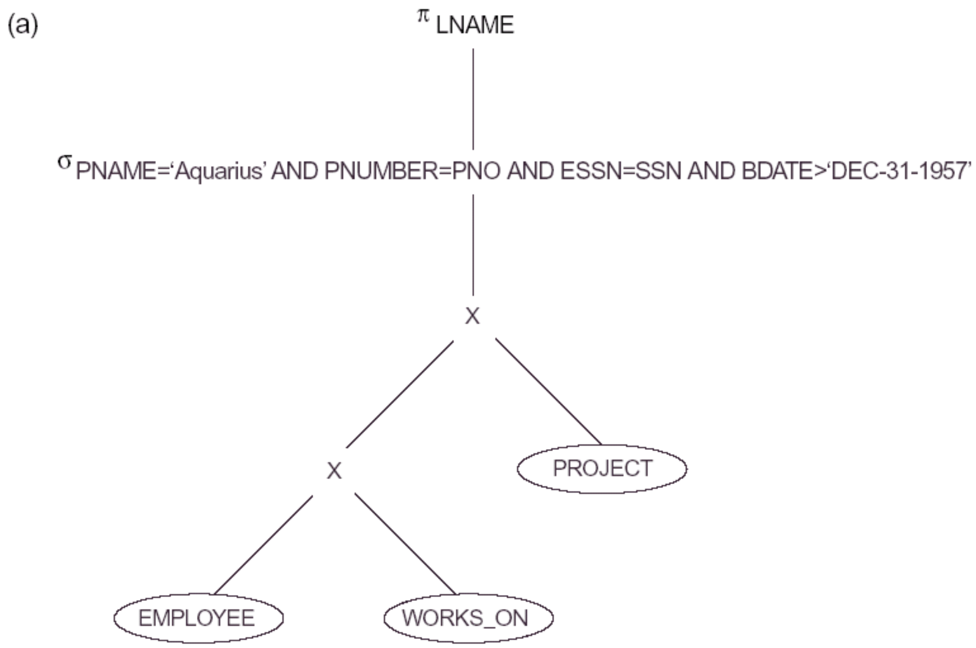
1. Using rule 1, break up SELECT operations with conjunctive conditions into a cascade of SELECT operations
2. Using rules 2, 4, 6, & 10, move each SELECT operation as far down the query tree
3. Using rules 5 & 9, rearrange the leaf nodes of the tree
   - Position the leaf node relations with the most restrictive SELECT operations
   - Make sure ordering of leaf nodes does not cause Cartesian product operations
4. Using rule 12, combine Cartesian product with SELECT into a JOIN operation
5. Using rules 3, 4, 7, 11, break down & move lists of projection attributes down the tree as far as possible by creating new Project operations as needed
6. Identify subtrees that represent groups of operations that can be executed by a single algorithm

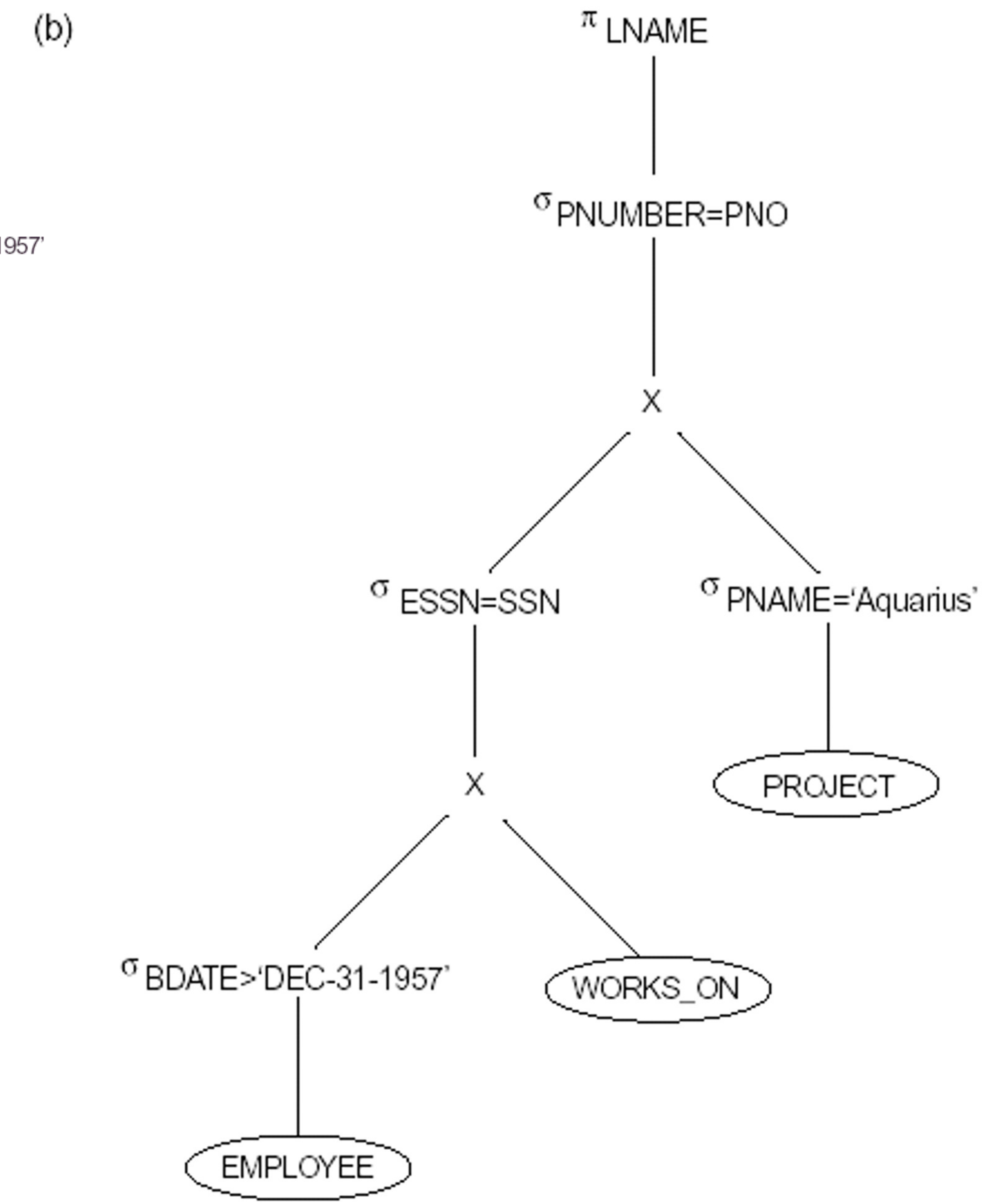
1. Using rule 1, break up SELECT operations with conjunctive conditions into a cascade of SELECT operations




```
SELECT Lname
FROM Employee, Works_on, Project
WHERE Pname = 'Aquarius' and
          Pnumber = Pno and
          ESSN = SSN and
          Bdate  > '1957-12-31';
```

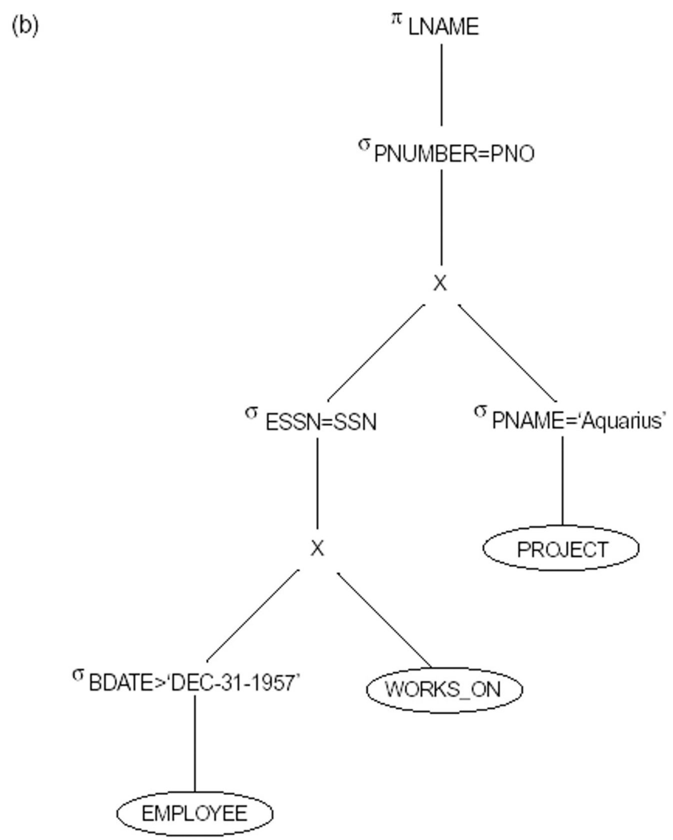
2. Using rules 2, 4, 6, & 10, move each SELECT operation as far down the query tree

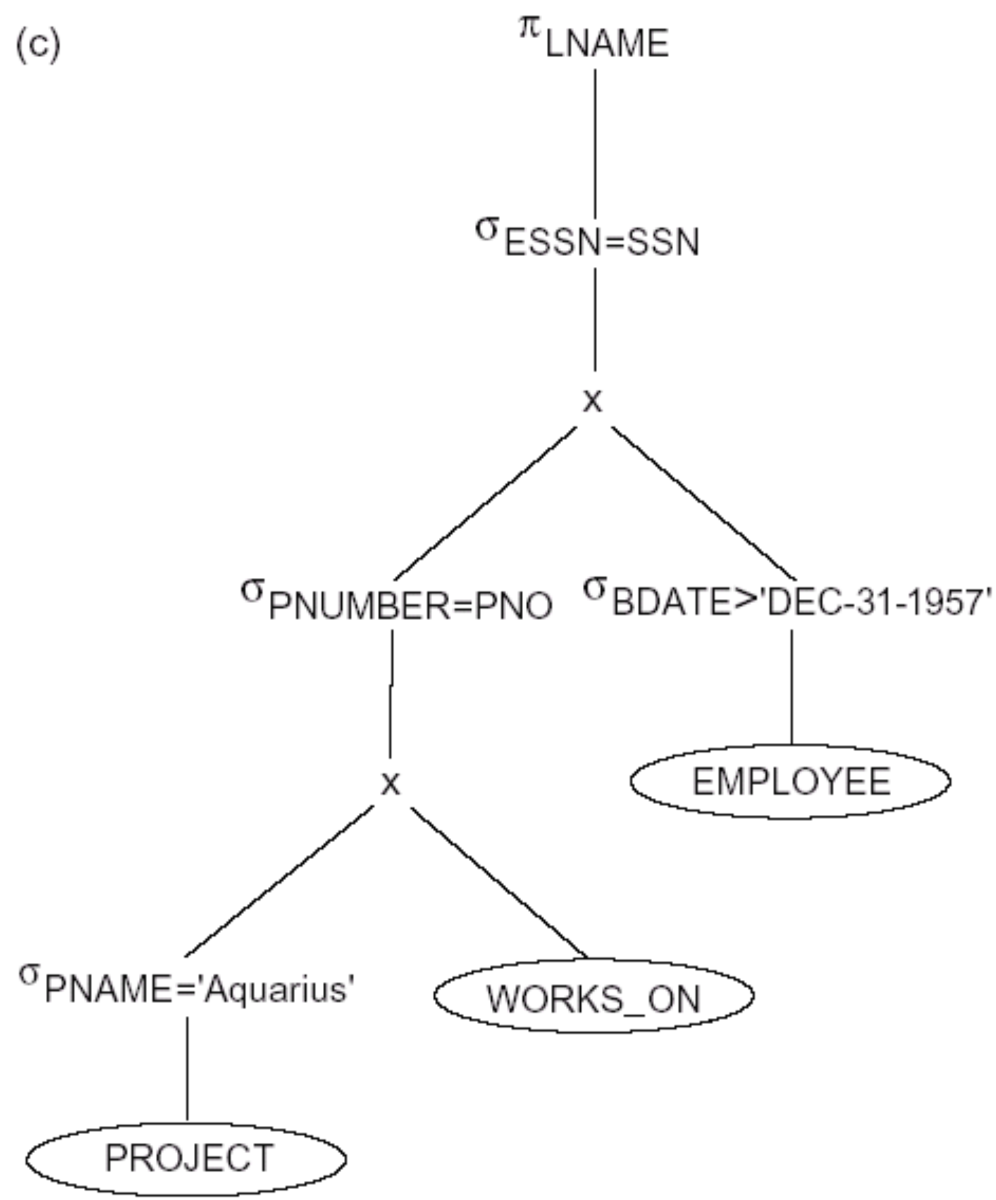




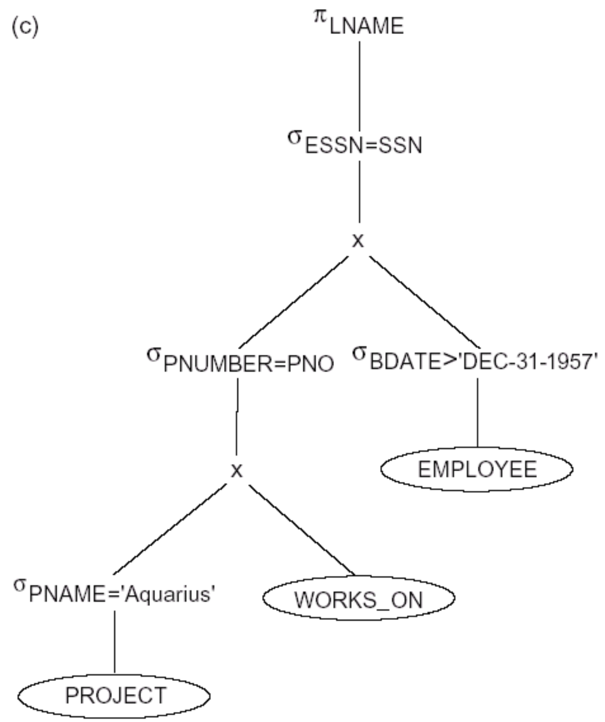
3. Using rules 5 & 9, rearrange the leaf nodes of the tree

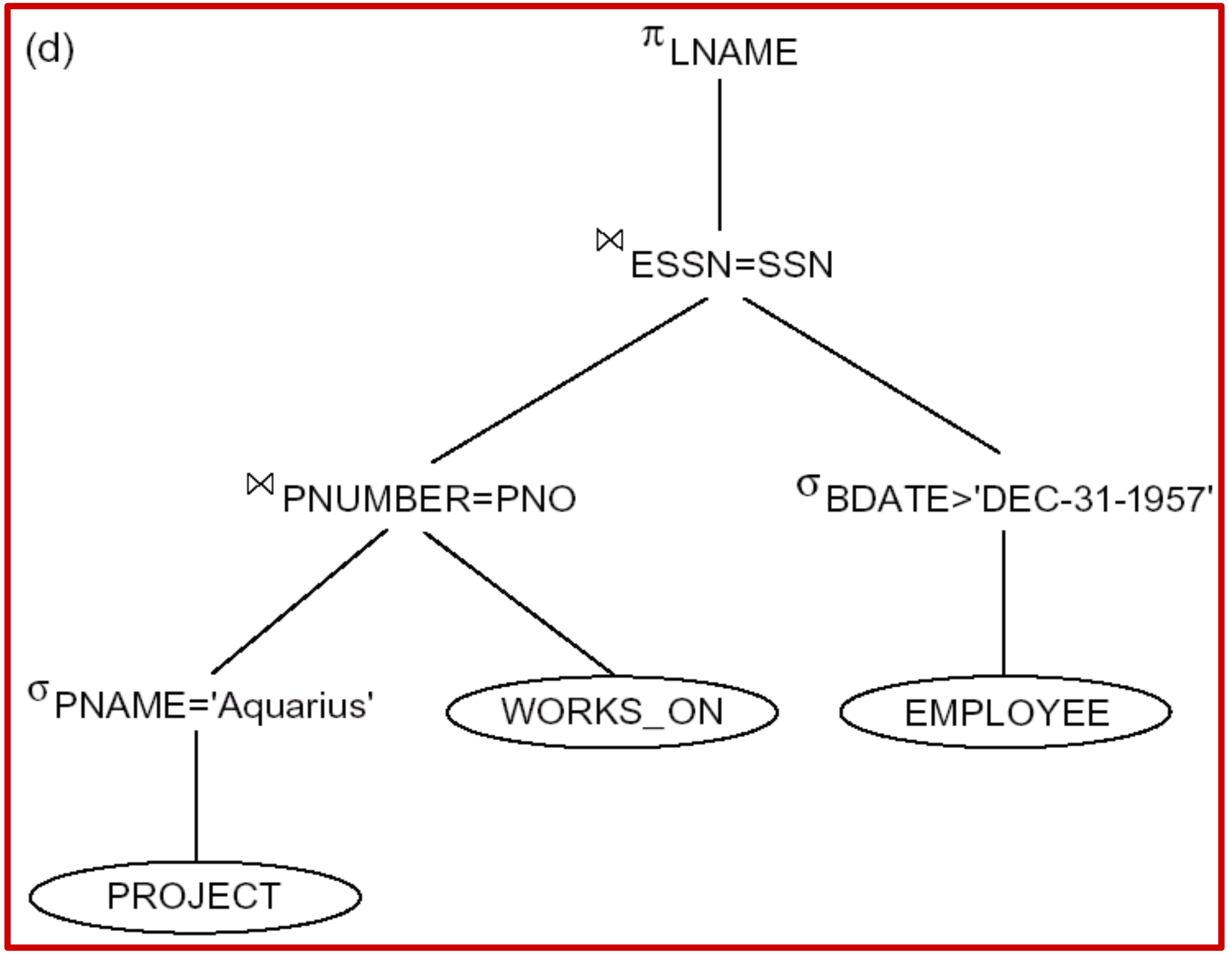
- Position the leaf node relations with the most restrictive SELECT operations
- Make sure ordering of leaf nodes does not cause Cartesian product operations



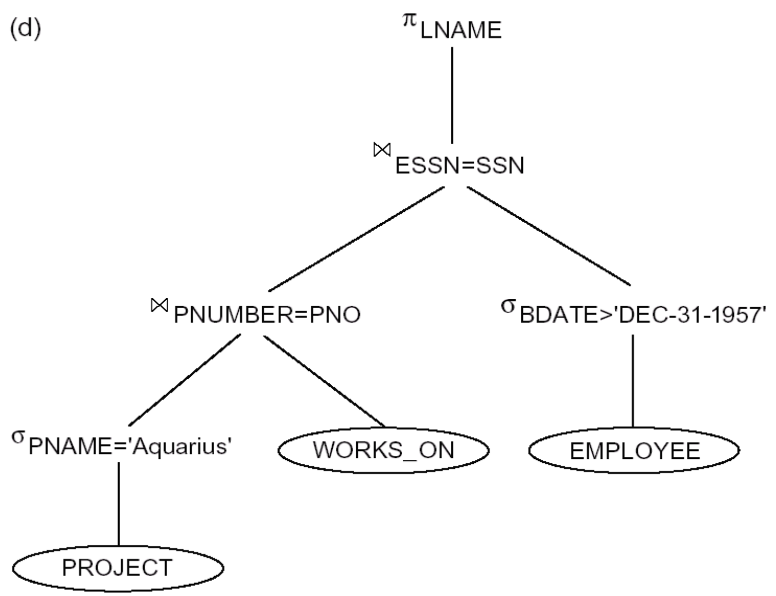


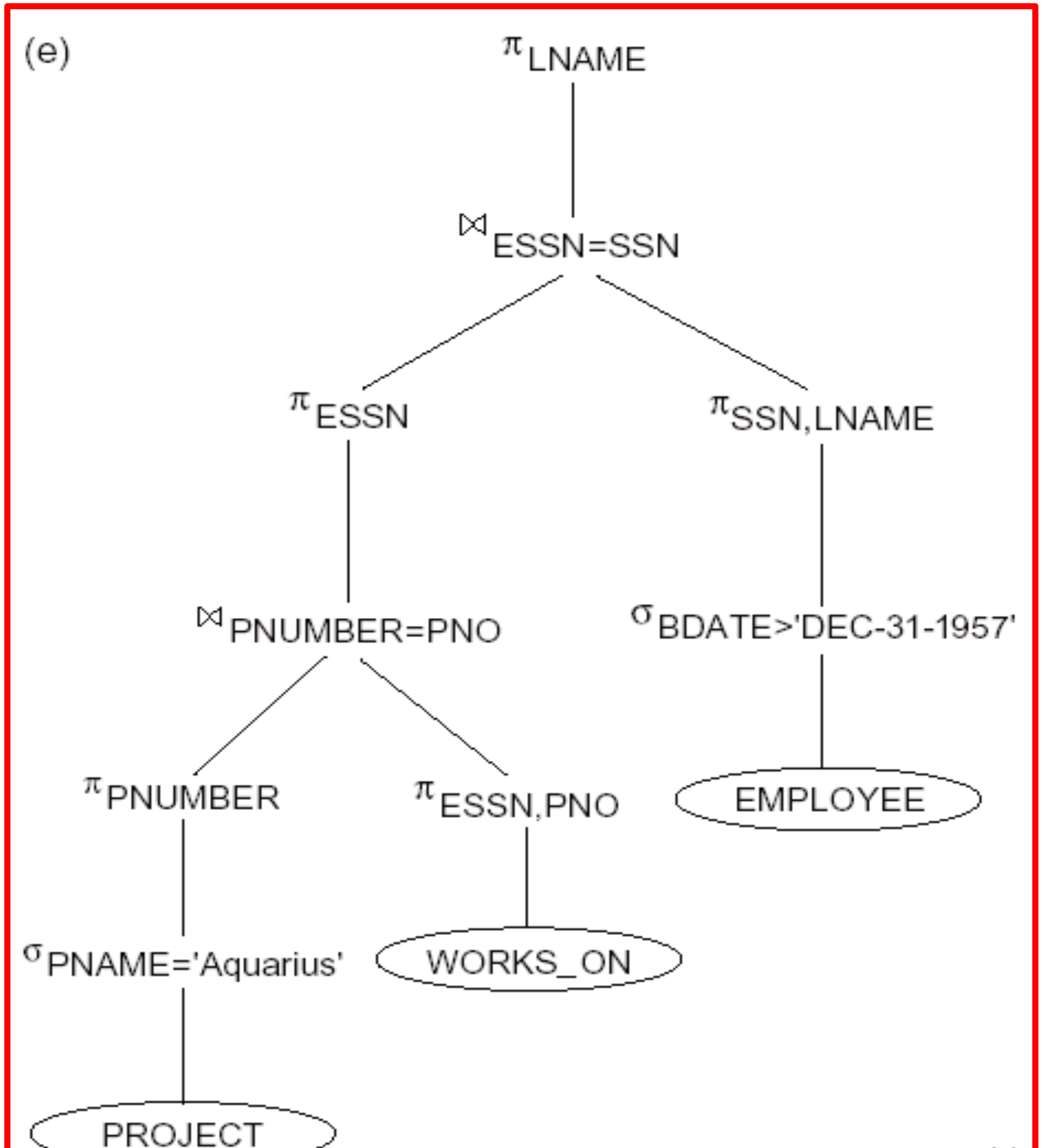
4. Using rule 12, combine Cartesian Product with SELECT into a JOIN operation





5. Using rules 3, 4, 7, 11, break down & move lists of projection attributes down the tree as far as possible by creating new Project operations as needed

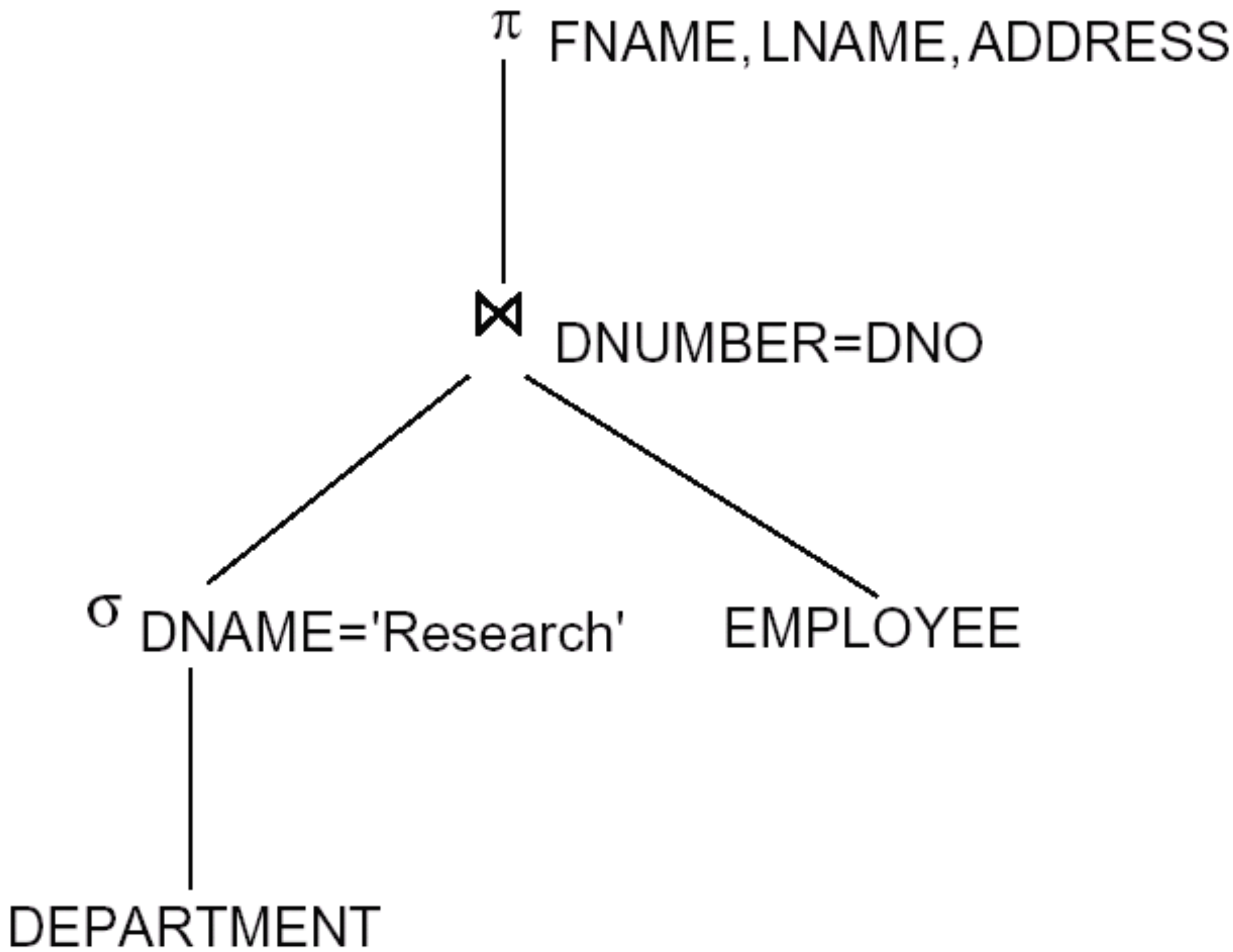




Summary of Heuristics for Algebraic Optimization

- Apply first the operation that reduce the size of intermediate results
  - Performing as early as possible SELECT operations to reduce #(tuples)
  - Performing as early as possible PROJECT operations to reduce #(attributes)
- SELECT and JOIN operations that are most restrictive should be executed before other similar operations

Converting Query Trees into Query Execution Plans



Materialized or Pipelined Evaluation

- Materialized evaluation: result of an operation is stored as a temporary relation
- Pipelined evaluation: resulting tuples of an operation are produced and forwarded directly to the next operation in the query sequence

Outline

1. Query Optimization
2. Using Heuristics in Query Optimization
3. Using Selectivity & Cost Estimates in Query Optimization
4. Semantic Query Optimization

Cost-based Query Optimization

- Use traditional optimization techniques that search the solution space to a problem for a solution that minimizes a cost function (dynamic programming)
- Cost function used in query optimization are estimates & not exact cost functions
- More suitable for compiled queries where the optimization is done at compile time

Cost of Executing a Query

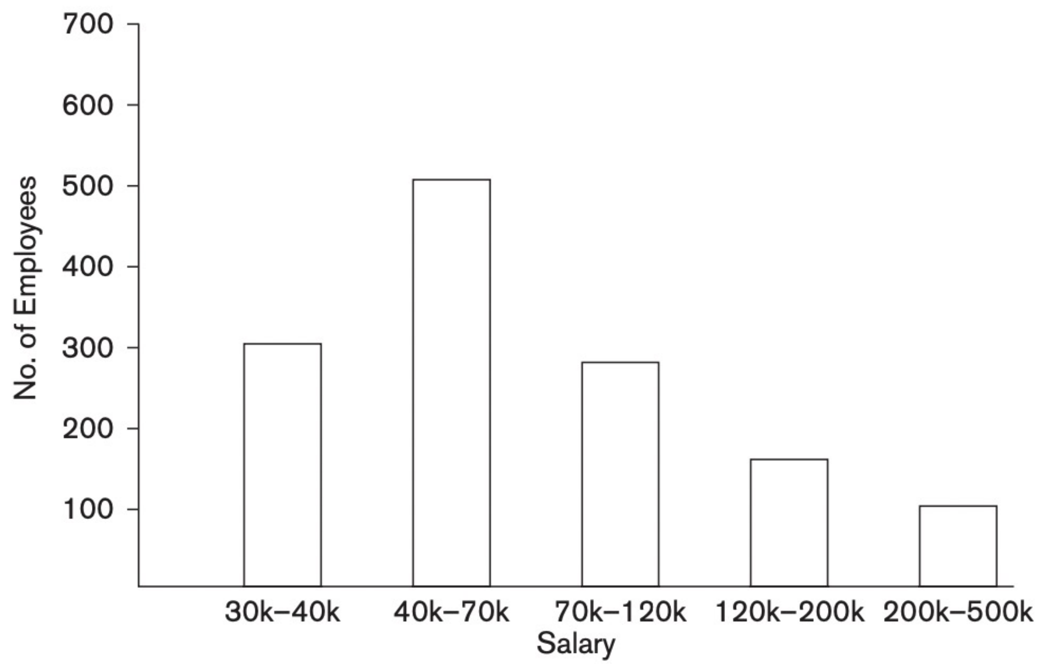
- Access cost to secondary storage (*)
- Storage cost
- Computation cost
- Memory usage cost
- Communication cost distributed databases

Catalog Information Used in Cost Functions

- Size of each file
  - #(records)
  - Average record size
  - #(blocks)
  - Blocking factor = # of records / block
- Primary access method, attribute for each file
- Secondary indexes & indexed attributes
- #(levels) of each multilevel index
- #(first level index blocks)
- #(distinct values of an attribute) & its selectivity (fraction of records satisfying an equality condition on the attribute) => selection cardinality of an attribute (average #(records) that will satisfy an equality selection condition on that attribute)

Histogram

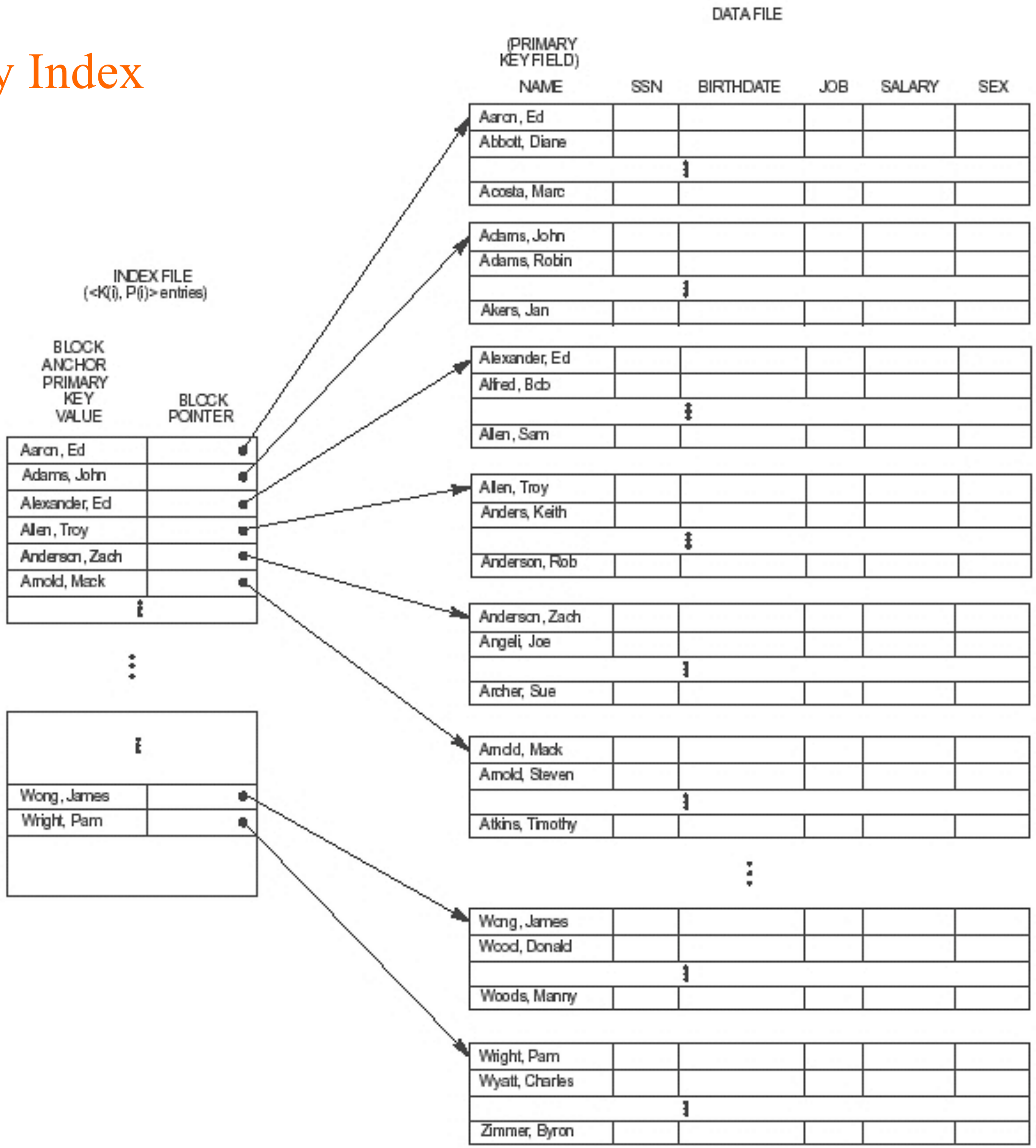
- tables or data structures maintained by the DBMS to record information about the distribution of data.
- Without a histogram, the best assumption is that values of an attribute are uniformly distributed over its range from high to low.
- Histograms divide the attribute over important ranges (called buckets) and store the total number of records that belong to that bucket in that relation.



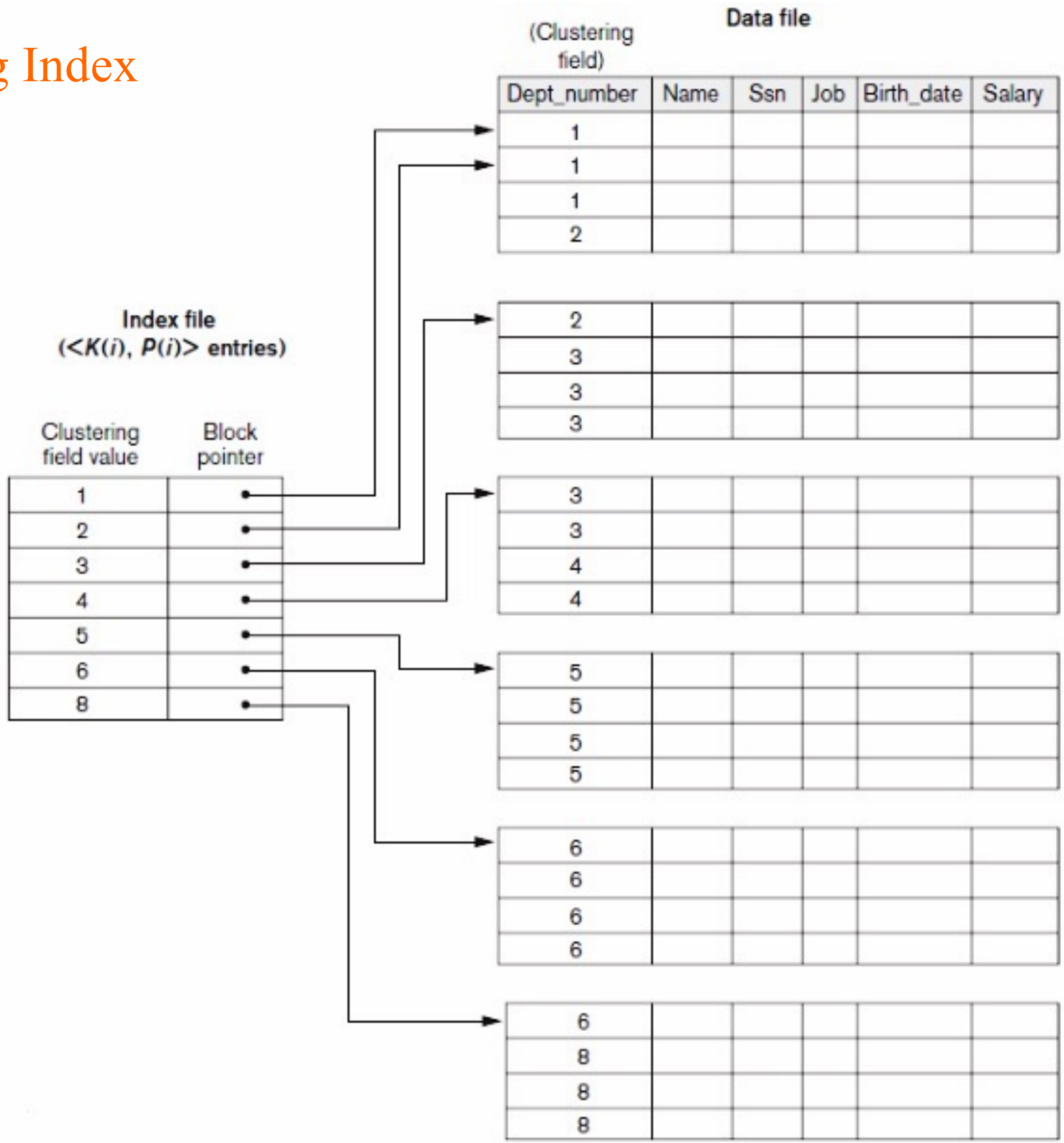
Sorted File

	Name	Ssn	Birth_date	Job	Salary	Sex
Block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
Block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
Block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
Block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
Block 5	Anderson, Zach					
	Angeli, Joe					
	⋮					
	Archer, Sue					
Block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					

Primary Index



Clustering Index



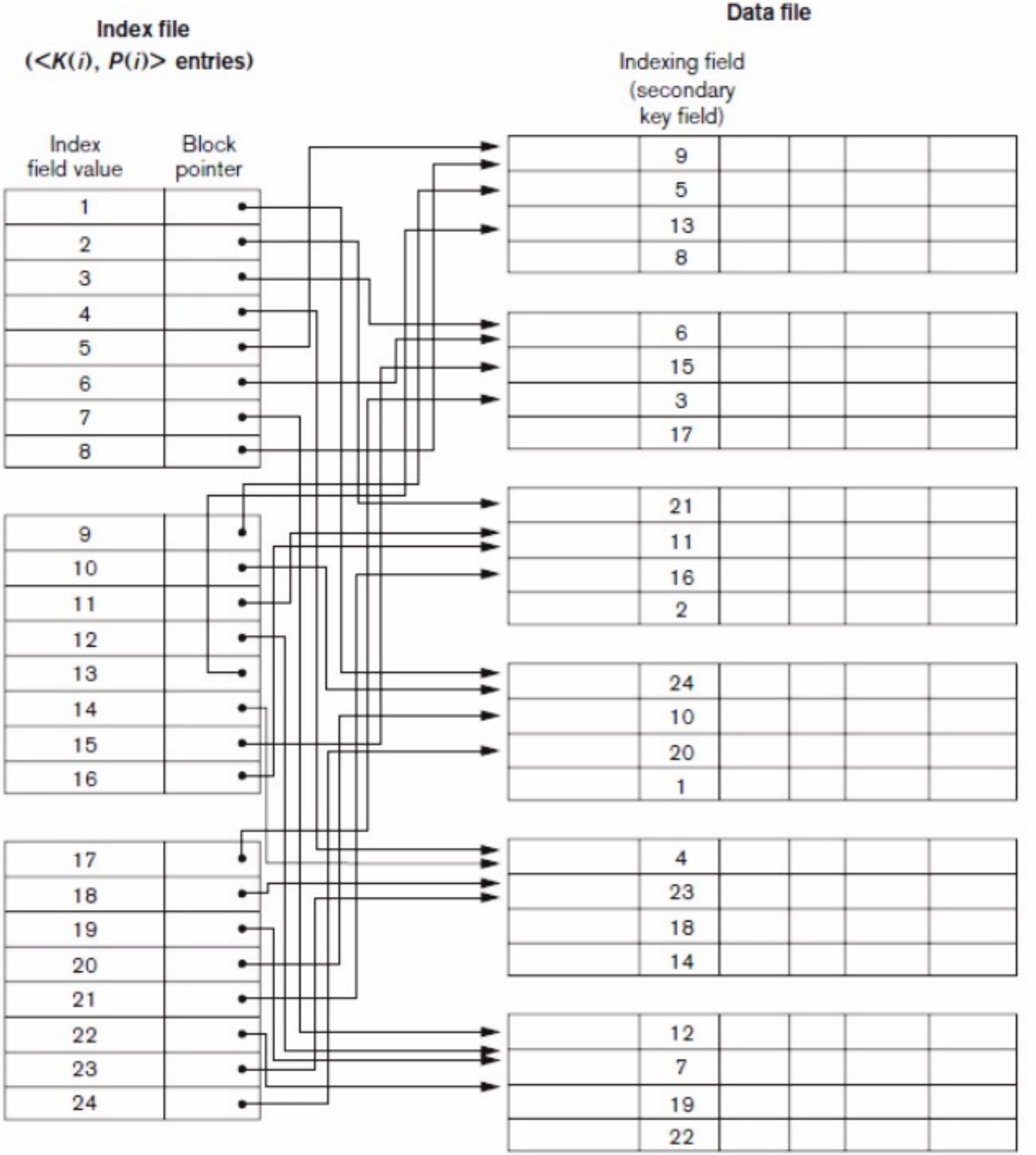
Example

- EMPLOYEE Table
  - sorted file on **Salary,** 10,000 records stored 2,000 disk blocks with blocking factor 5 records/block,
  - A clustering index on **Salary**,
    with 3 levels and average selection cardinality 20 (selectivity = 0.2).
  - A secondary index on the key attribute **SSN** with 4 levels
  - A secondary index on the non-key attribute **Dno**, with 2 levels.
    The first-level index blocks 4. There are 125 distinct values for Dno, selection cardinality 80 (selectivity = 0.008)
  - A secondary index on **Sex**, with 1 level. There are 2 values for the Sex attribute. The average selection cardinality is 5000 (selectivity = 0.5)

$\sigma_{SSN='123456789'}$ (EMPLOYEE)

- Option 1: Linear Search
  - Average cost for linear search on key attribute: 2000/2 = 1000 block I/O
- Option 2: Using Secondary Index
  - 4 (levels) +1 (record) = 5 block I/O

Secondary Index on Key Field (unique)



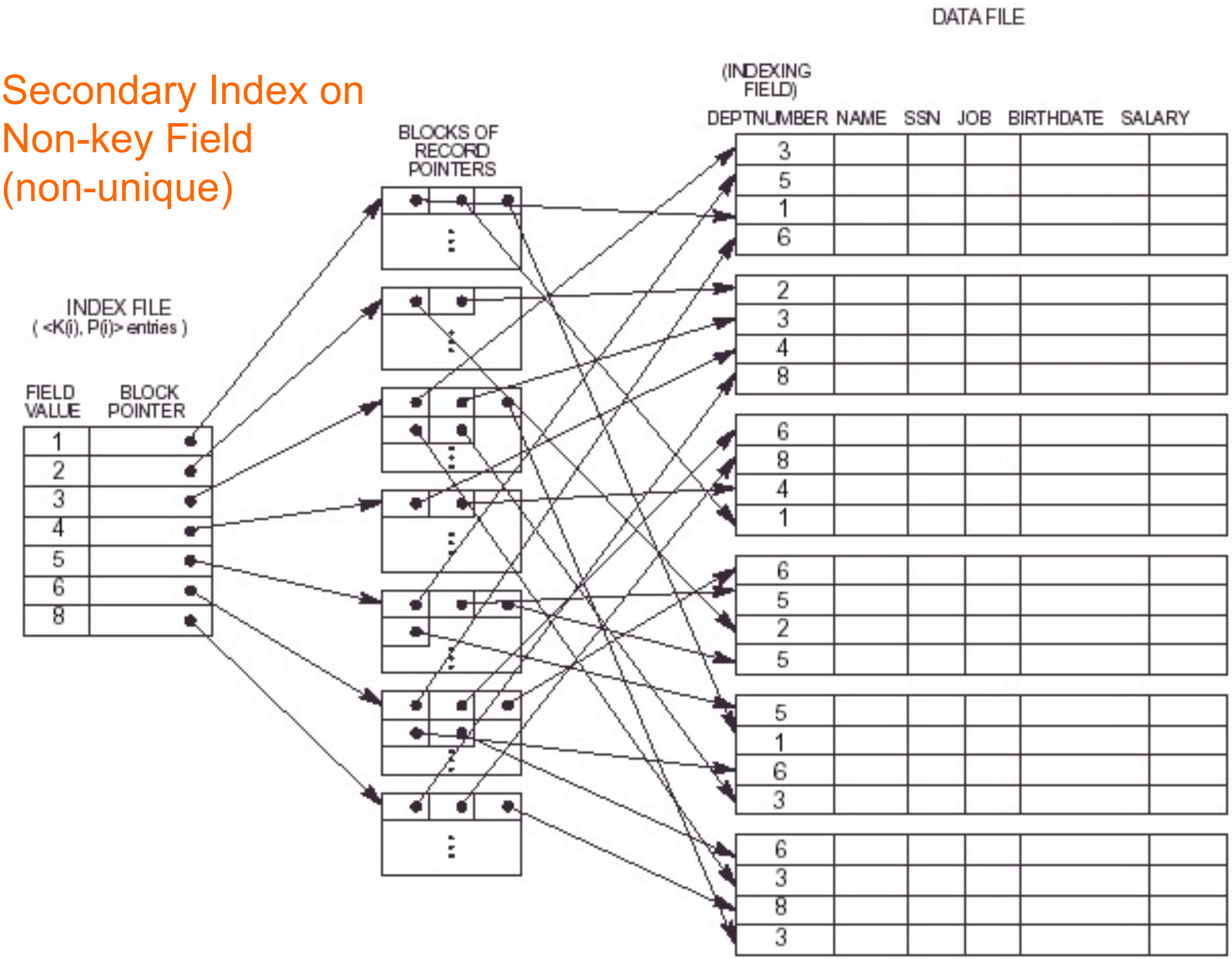
$\sigma_{Dno=5}$(EMPLOYEE)

- Option 1: Linear Search
  - Average cost for linear search on non-key attribute: 2000 block I/O
- Option 2: Using Secondary Index on Dno

  +1 (block of record pointers)
  - Worst case cost: 2 (levels) +80 (records, blocks) = ~~82~~ block I/O

    83

Secondary Index on Non-key Field (non-unique)

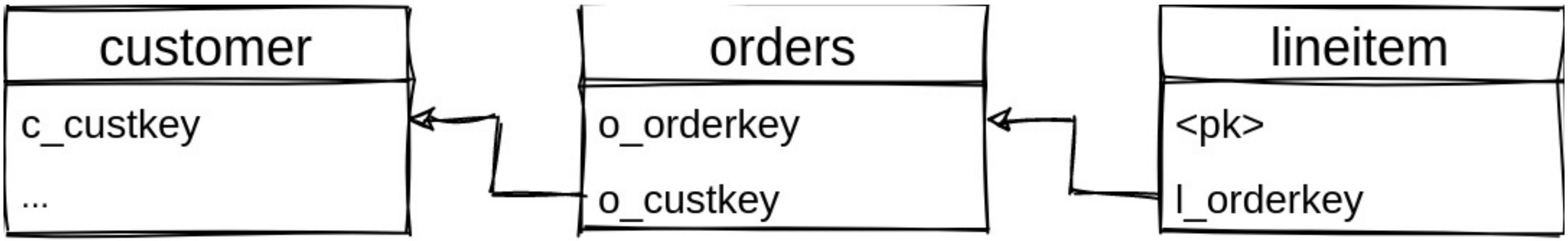


$\sigma_{Dno=2}$(Employee)

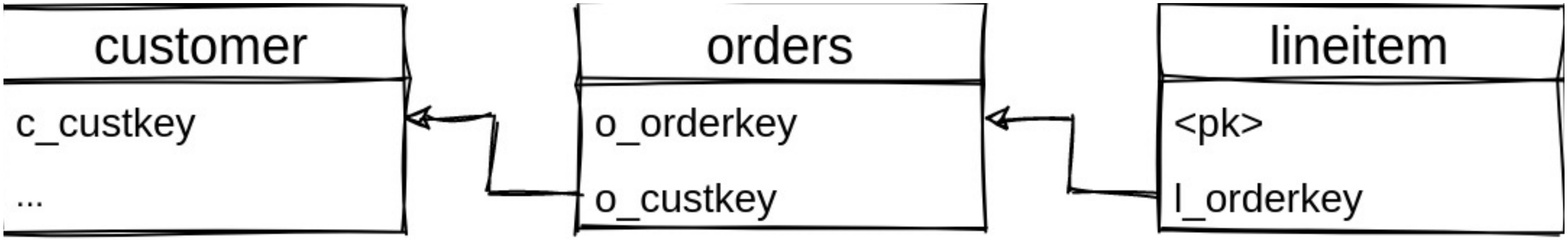
Cost = #(index level) + #(selection cardinality) + 1

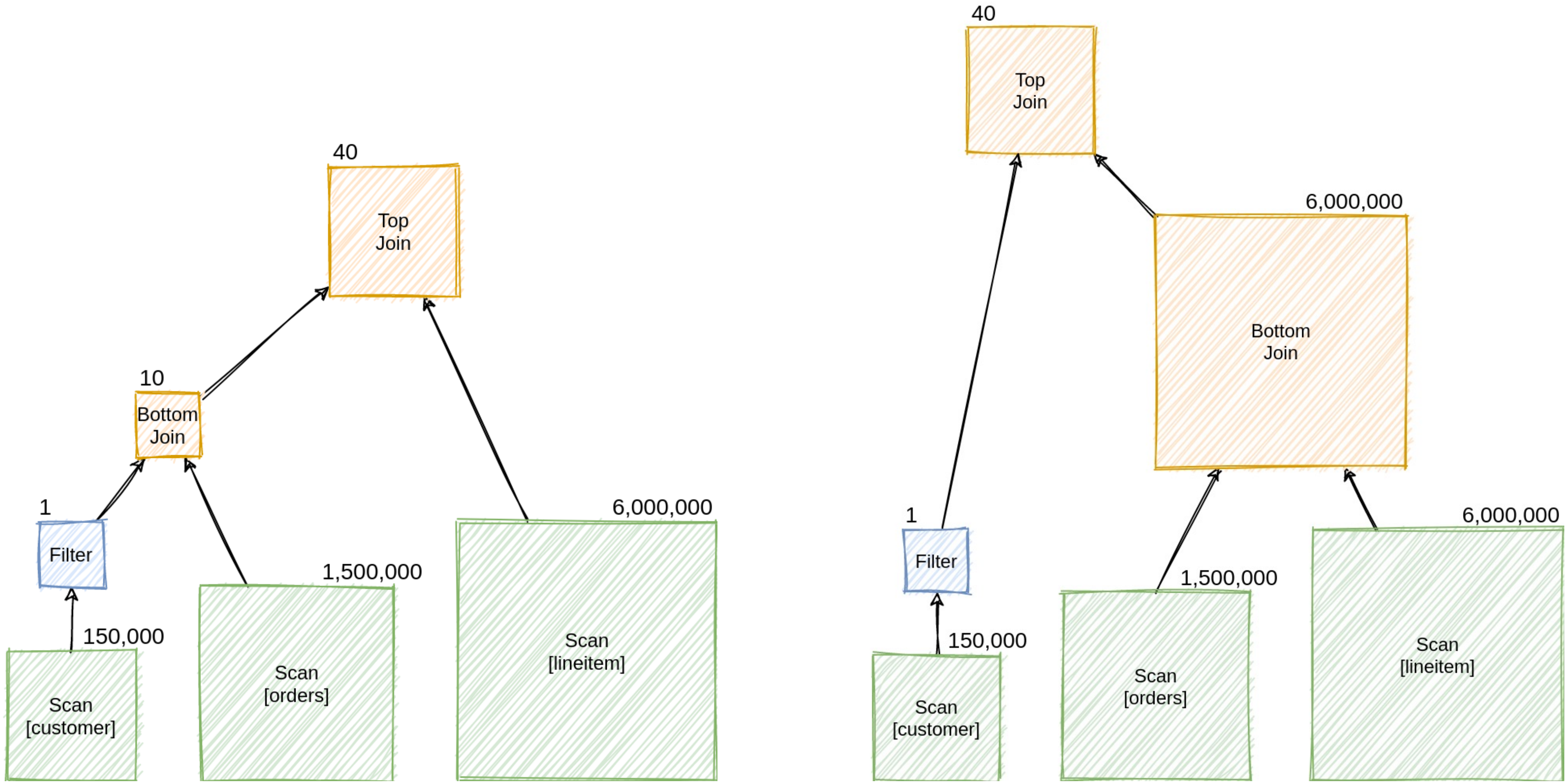
Multiple Relation Queries & Join Ordering

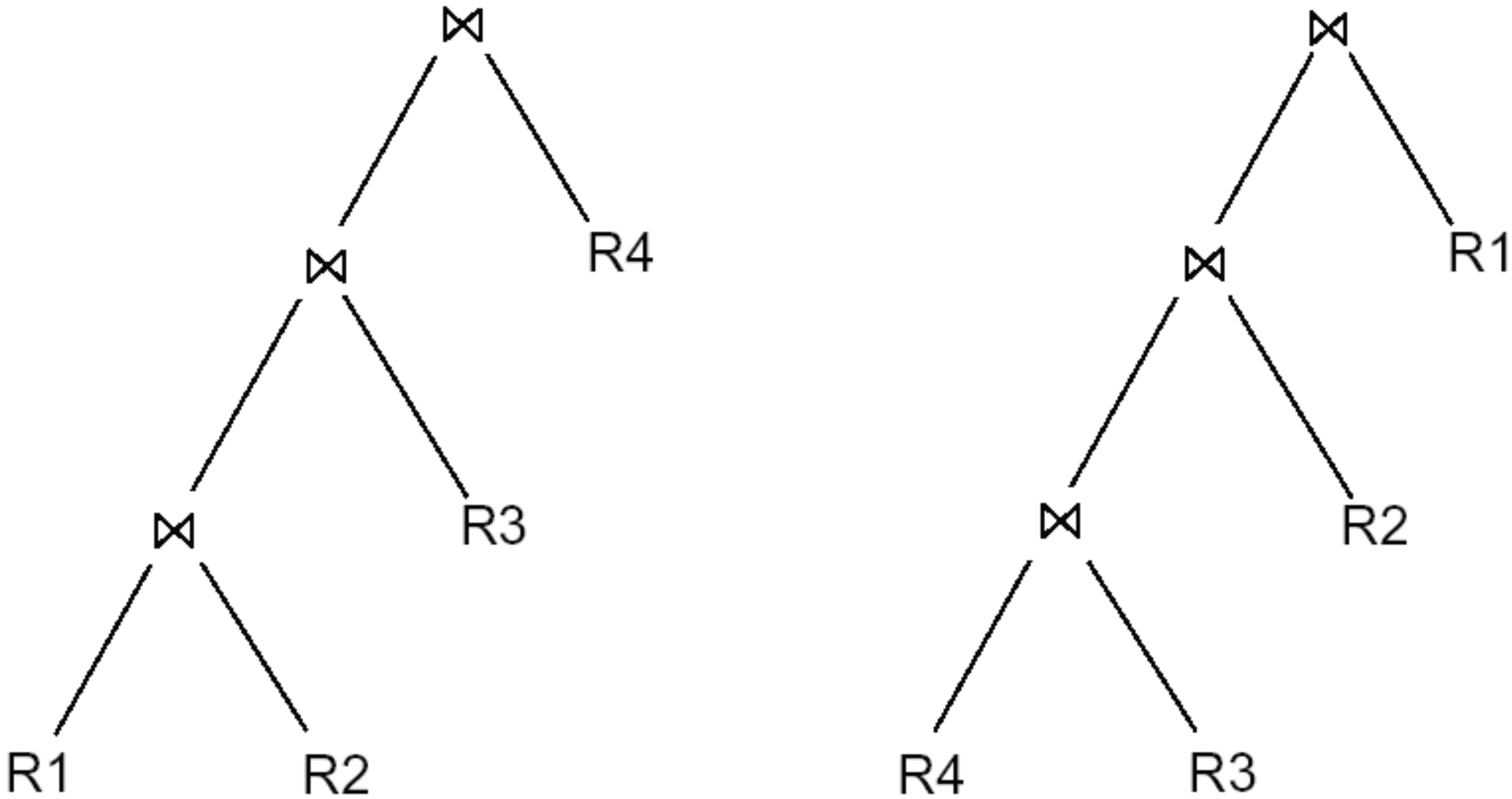
- A query that join *n* relations will have (*n*-1) join operations => a large number of join orders (n-1)! or more
- Left-deep query trees are generally preferred because of pipelining
- Query optimizer
  - limit the structure of a query tree to that of left-deep trees
  - Choose the particular left-deep tree with the lowest estimated cost
  - Find an ordering that will reduce the size of the temporary result by dynamic programming



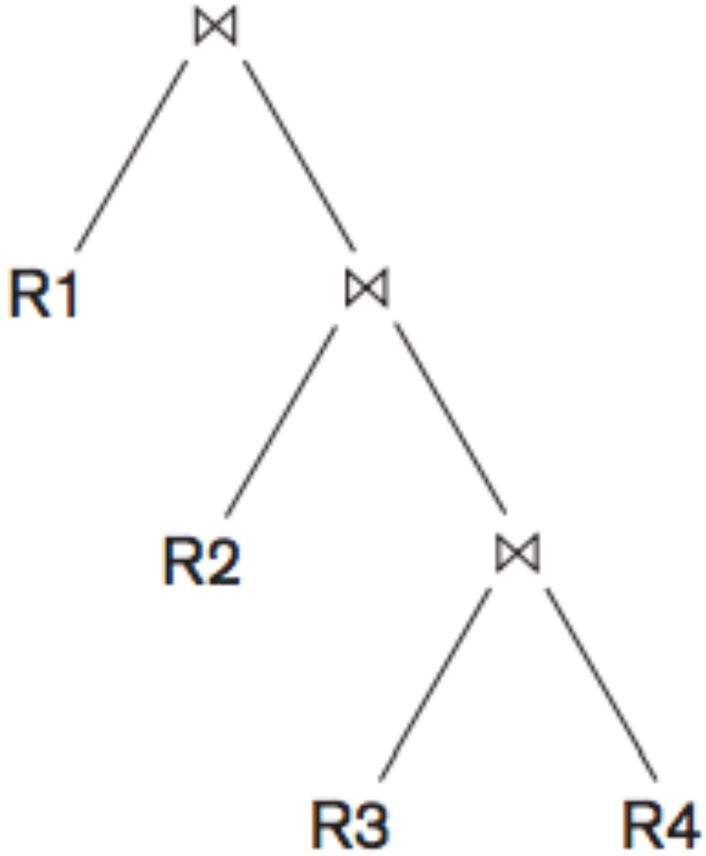

```
SELECT   lineitem.*
FROM     customer, orders, lineitem
WHERE    c_custkey = 101 AND
         c_custkey = o_custkey AND
         o_orderkey = l_orderkey
```


customer
c_custkey
...
orders
o_orderkey
o_custkey
lineitem
<pk>
l_orderkey

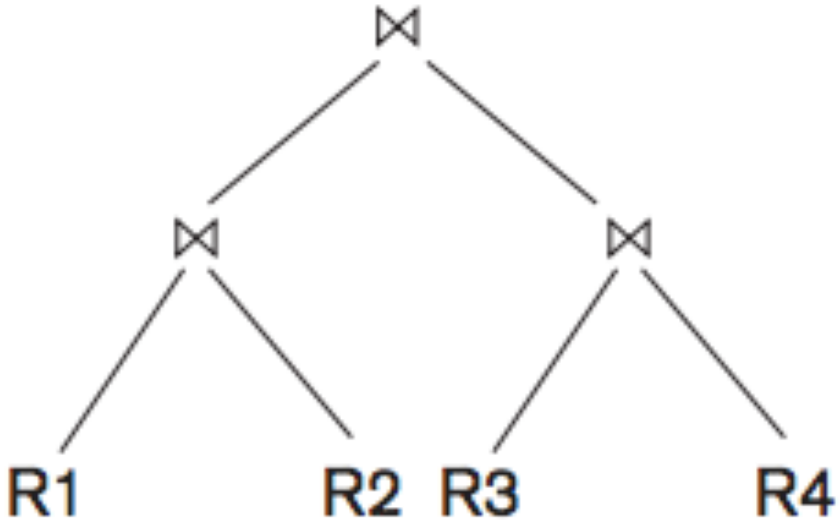

40
Top Join
10
Bottom Join
1
Filter
150,000
Scan [customer]
1,500,000
Scan [orders]
6,000,000
Scan [lineitem]
40
Top Join
6,000,000
Bottom Join
1
Filter
150,000
Scan [customer]
1,500,000
Scan [orders]
6,000,000
Scan [lineitem]



left deep join query tree



right deep join query tree



Bushy join query tree

Table19.1 Number of Permutations of Left-Deep and Bushy Join Trees of n Relations

No. of Relations N	No. of Left-Deep Trees $N!$	No. of Bushy Shapes $S(N)$	No. of Bushy Trees $(2N-2)!/(N-1)!$
2	2	1	2
3	6	2	12
4	24	5	120
5	120	14	1,680
6	720	42	30,240
7	5,040	132	665,280

Outline

1. Query Optimiation
2. Using Heuristics in Query Optimization
3. Using Selectivity & Cost Estimates in Query Optimization
4. Semantic Query Optimization

Semantic Query Optimization

- Use constraints specified on schema in order to modify one query into another more efficient query

```
SELECT E.Lname, M.Lname
FROM Employee as E, Employee as M
WHERE E.SuperSSN=M.SSN AND
            E.Salary > M.Salary ;
```

find employees with salary > their direct supervisor

- If there is a constraint that

  no employee can earn more than direct supervisor

  then it is unnecessary to execute the query

result = ϕ

Semantic Query Optimization (cont.)

- Use referential integrity constraints specified on schema in order to modify one query into another more efficient query

```
SELECT Lname, Salary
FROM Employee, Department
WHERE Employee.Dno=Department.DNumber AND
        Employee.Salary > 100000 ;
```

```
SELECT Lname, Salary
FROM Employee
WHERE Employee.Salary > 100000 AND
        Employee.Dno IS NOT NULL;
```

Displaying Query Plan

- Most commercial RDBMS have a provision to display the execution plan produced by query optimizer
- DBA can view the execution plans & try to understand the decision made by the optimizer.
- Common syntax

  EXPLAIN <query>

Displaying Query Plan (cont.)

```
EXPLAIN SELECT * FROM tenk1 WHERE unique1 < 100 AND unique2 > 9000;

                                     QUERY PLAN
-------------------------------------------------------------------------------------
 Bitmap Heap Scan on tenk1  (cost=25.08..60.21 rows=10 width=244)
   Recheck Cond: ((unique1 < 100) AND (unique2 > 9000))
   ->  BitmapAnd  (cost=25.08..25.08 rows=10 width=0)
         ->  Bitmap Index Scan on tenk1_unique1  (cost=0.00..5.04 rows=101
               Index Cond: (unique1 < 100)
         ->  Bitmap Index Scan on tenk1_unique2  (cost=0.00..19.78 rows=999
               Index Cond: (unique2 > 9000))
```

Example of PostgreSQL

Conclusions

- Query Optimization
- Query Tree & Equivalence-Preserving
- Heuristic query optimization
- Cost-based query optimization
- Semantic query optimization