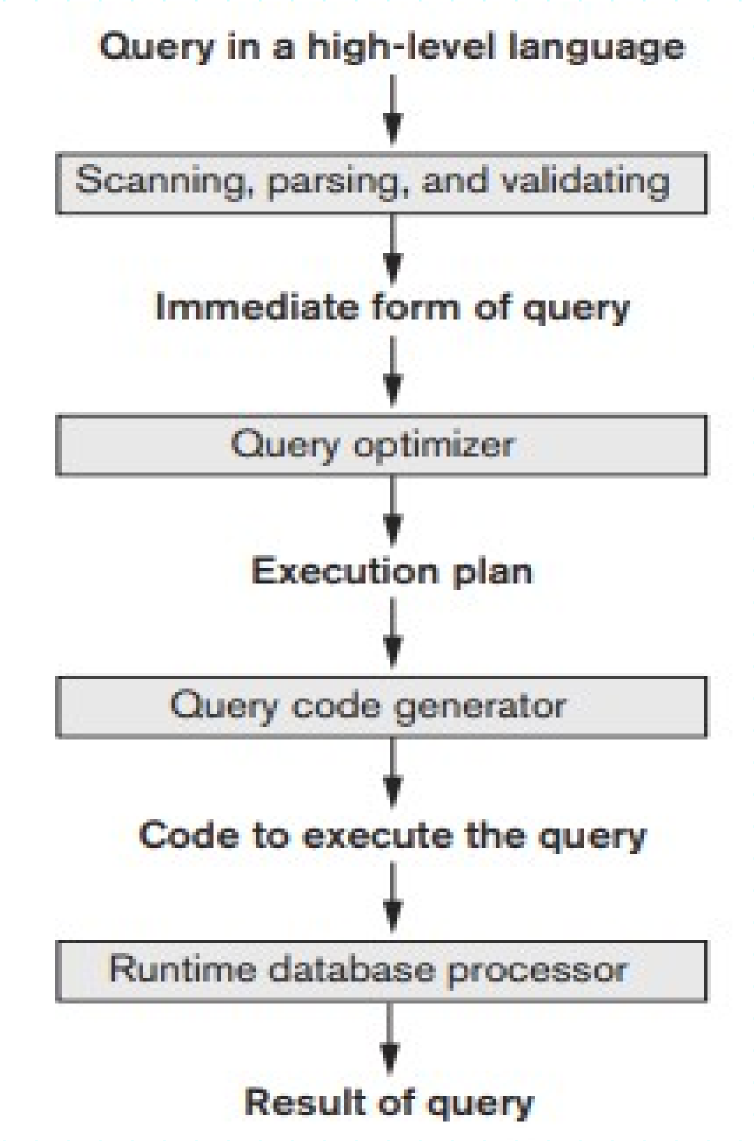
Database Management Systems

- Query Optimization

Doug Shook

Query Execution



Query in a high-level language

Scanning, parsing, and validating

Immediate form of query

Query optimizer

Execution plan

Query code generator

Code to execute the query

Runtime database processor

Result of query

Query Optimization

- Optimization is hard
  - Many possible combinations

- Maybe we aren't finding the “optimal” solution.....

- What operations should we focus on?

SELECT Optimization

- Consider a simple SELECT query:

```
SELECT
FROM
WHERE
```

- What possible approaches are there to executing this?
  - What if WHERE is more complex (uses AND)?

Selectivity

- Ratio of tuples that satisfy condition to the total number of records

- Can we compute this value exactly?

- How does this help us with optimization?

JOIN Optimization

- There's more than one way to perform a join
  - Nested Loop
  - Single Loop
  - Sort-merge
  - Partition-hash

- Which version is the simplest? Which version is the fastest?

Nested Loop Join

- How does table size affect this join?

- Example: assume one table has 10 pages, and another has 2000 pages. How many page accesses do we need for each nested loop configuration?

JOIN Selection Factor

- How many records from each table to we expect to match the join condition?

- How does this help us?
  - Which JOIN type does this affect?

Heuristics

- We can optimize queries by manipulating the query tree directly
  - Must satisfy order of operations
- Idea: one tree can be rewritten in numerous ways
  - Let's find the fastest version

Heuristics

- SELECT:
  - Can cascade conjunctive SELECT operations
  - Operations are commutatitve

- PROJECT
  - Cascading PROJECTS are somewhat irrelevant
  - Can be commuted with a SELECT
    - When?

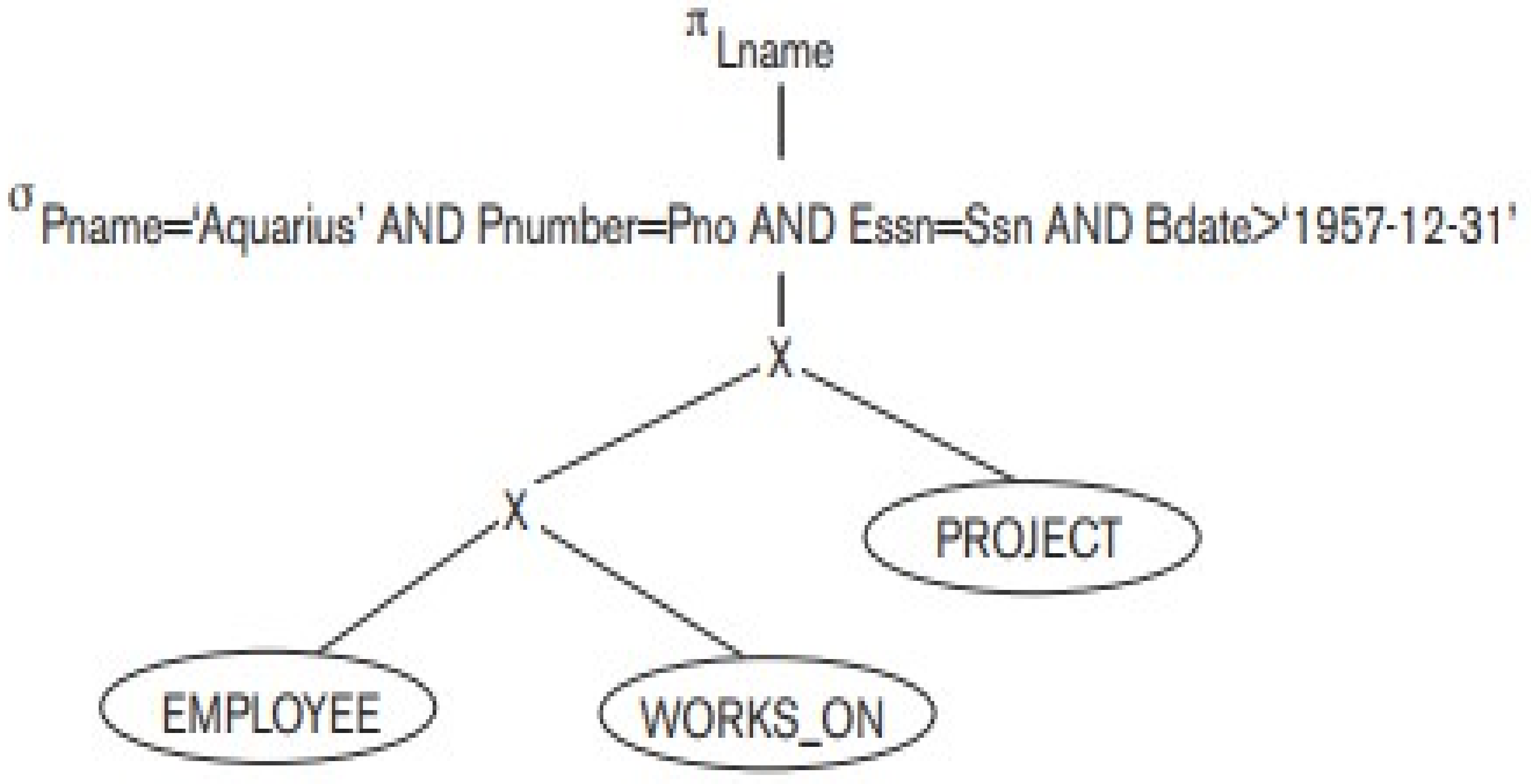
Heuristics

- JOINs
  - Can be commuted with SELECT if select only affects one side of the join
  - Can be commuted with PROJECT if all projected columns are part of the tables being joined
- Set operations
  - Union and intersection are commutative
  - Union, intersection, and join are associative
  - All set operations are commutative with SELECT
  - PROJECT is commutative with union
- Cartesian product + SELECT = JOIN

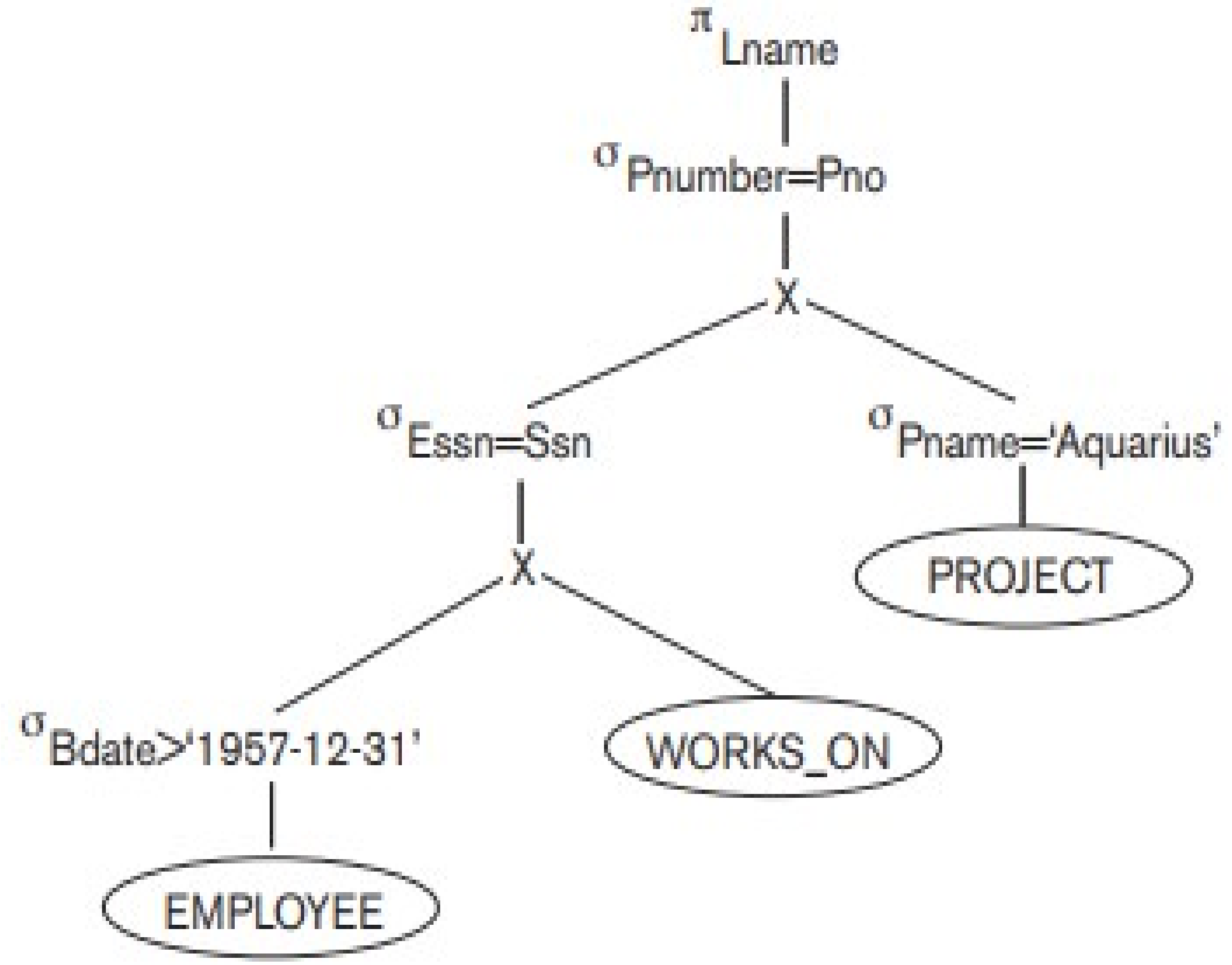
Optimization using Heuristics

- Break up conjunctive SELECTS
  - Allows us to move them around more easily
- Use SELECT commutativity to move SELECT operations as far down the tree as possible
- When performing operations on multiple tables, move relations with SELECT restrictions as far down the tree as possible
- Combine cartesian products with selects when applicable
- Move PROJECTs down the tree as far as possible

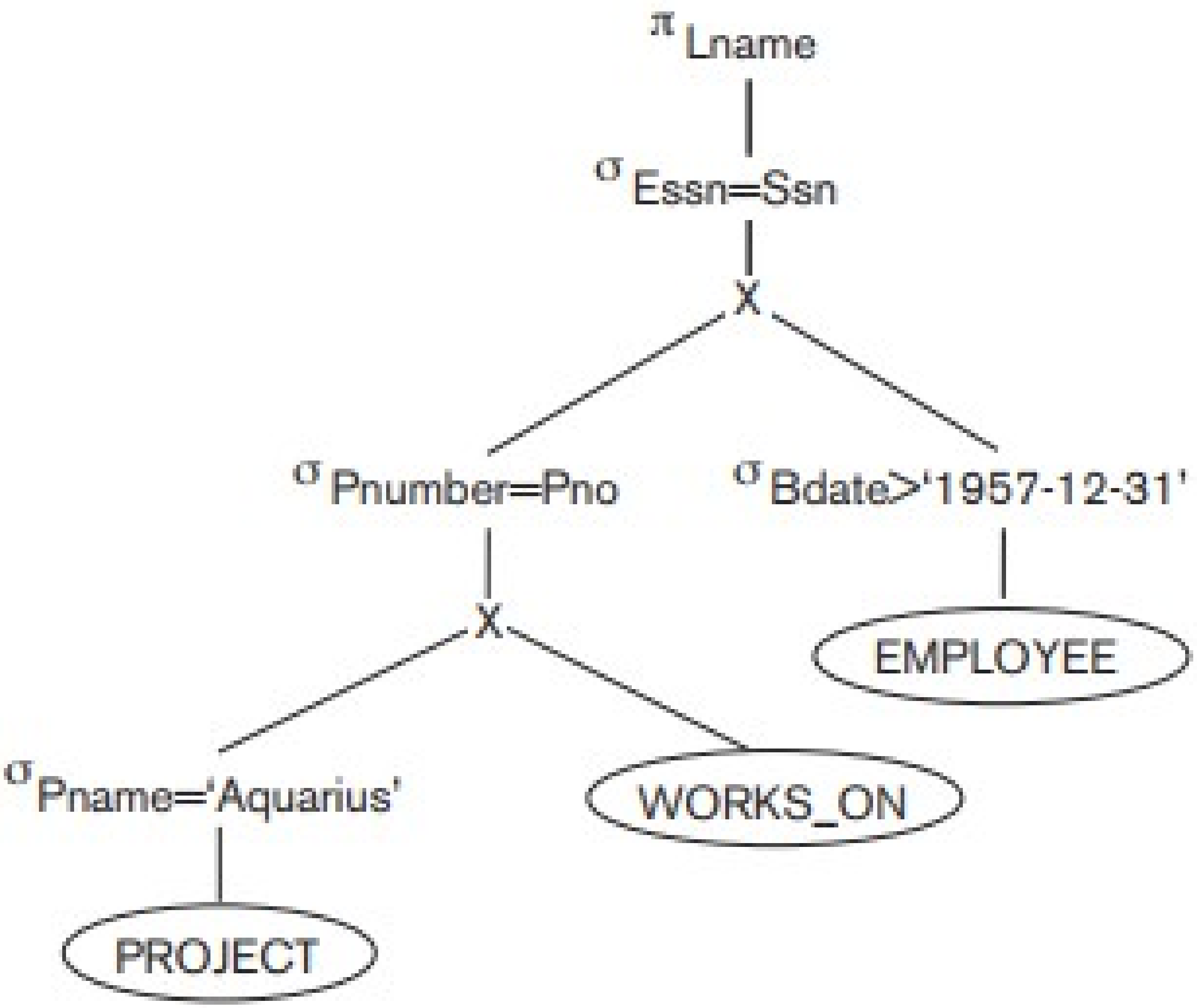
Heuristic Example


π_{Lname}
$\sigma_{\text{Pname='Aquarius' AND Pnumber=Pno AND Essn=Ssn AND Bdate>'1957-12-31'}}$
X
X
PROJECT
EMPLOYEE
WORKS_ON

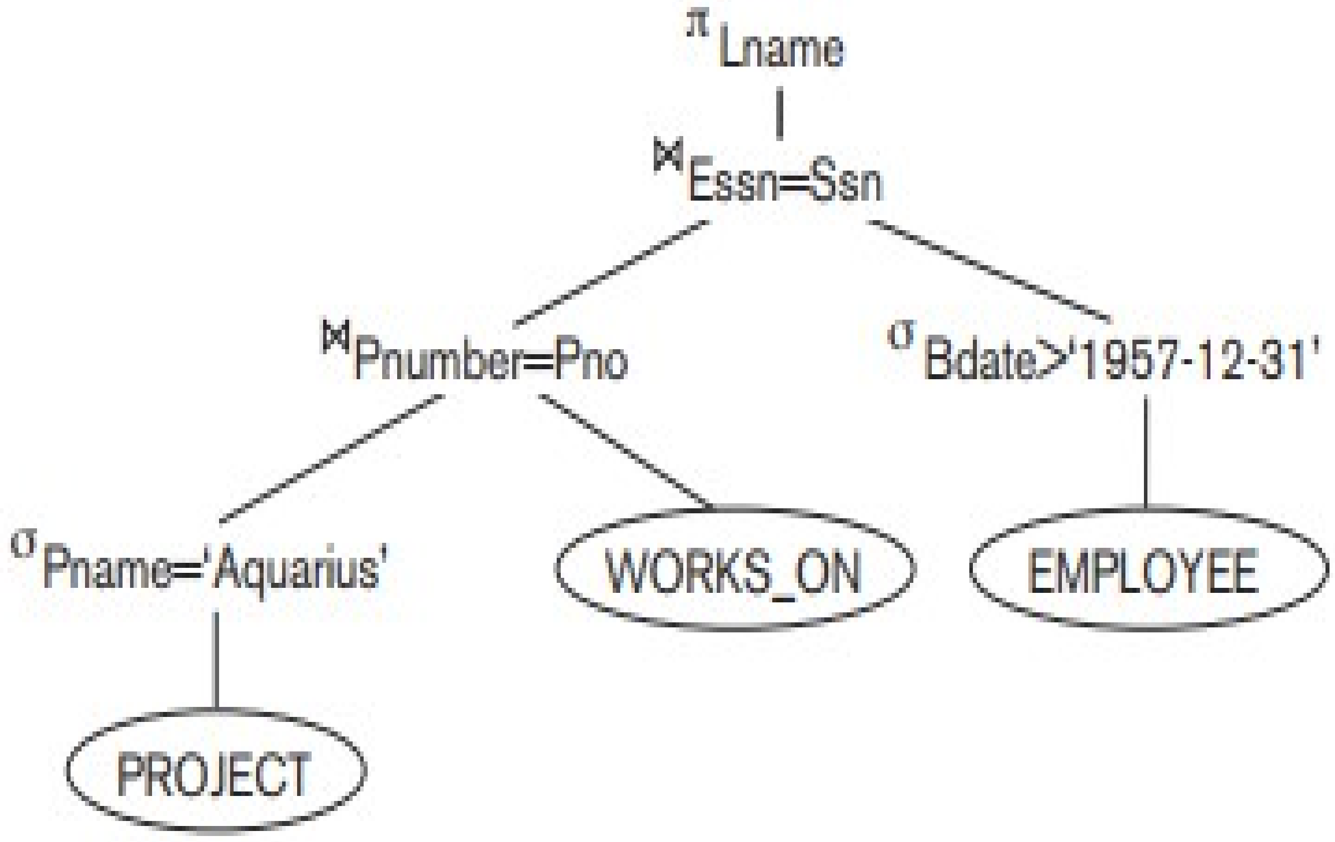
Heuristic Example



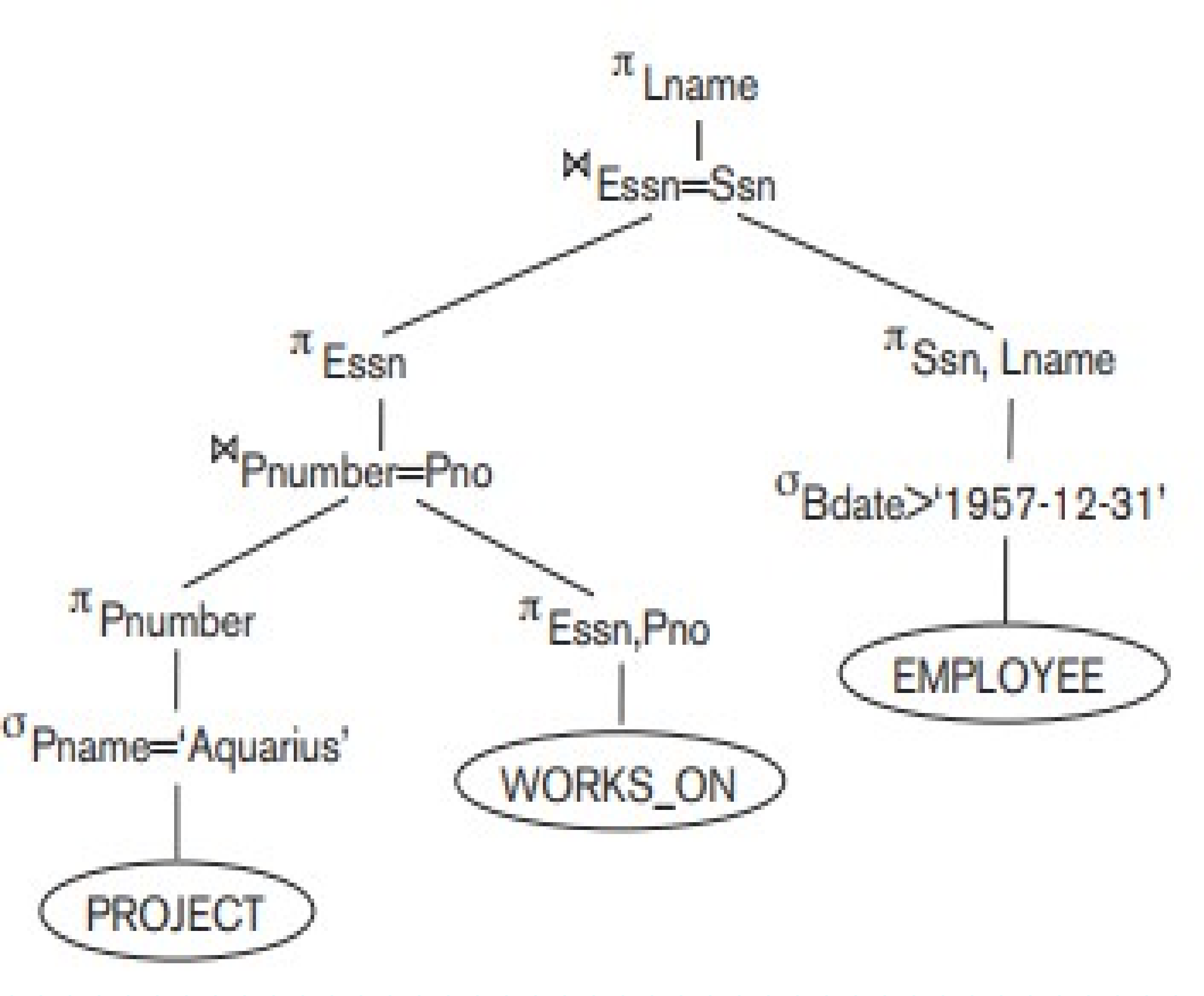
Heuristic Example


π_{Lname}
$\sigma_{\text{Essn=Ssn}}$
X
$\sigma_{\text{Pnumber=Pno}}$
$\sigma_{\text{Bdate>'1957-12-31'}}$
X
EMPLOYEE
$\sigma_{\text{Pname='Aquarius'}}$
WORKS_ON
PROJECT

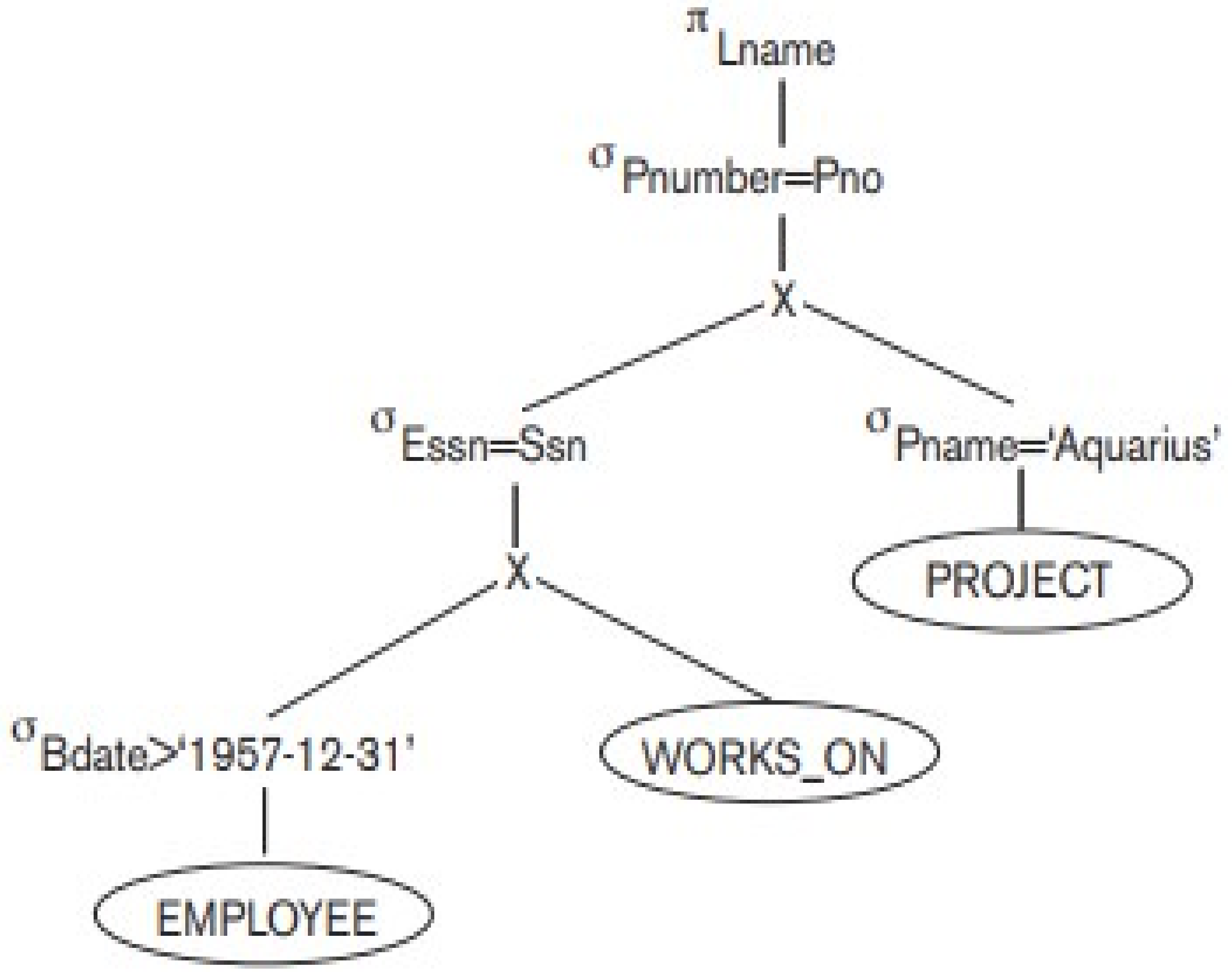
Heuristic Example



Heuristic Example



Heuristic Example


π_{Lname}
$\sigma_{\text{Pnumber=Pno}}$
X
$\sigma_{\text{Essn=Ssn}}$
$\sigma_{\text{Pname='Aquarius'}}$
X
PROJECT
$\sigma_{\text{Bdate>'1957-12-31'}}$
WORKS_ON
EMPLOYEE