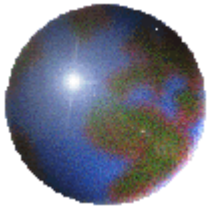
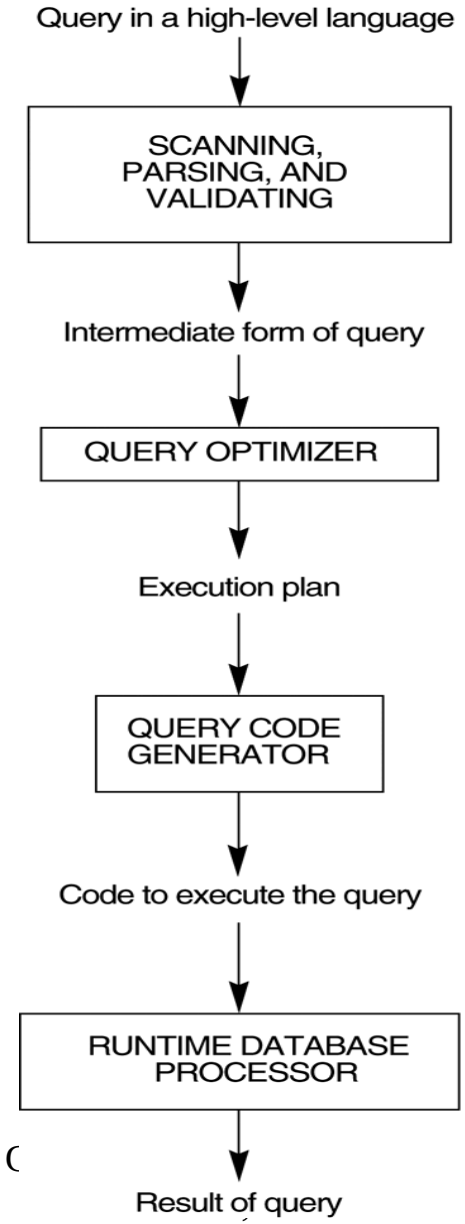


Query Processing and Optimization

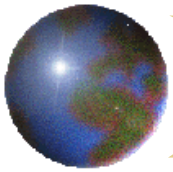
Prof. Geetha Mary A
Associate Professor,
SCOPE, VIT, Vellore

Typical steps when processing a high-level query



Code can be:

- Executed directly (interpreted mode)
- Stored and executed later whenever needed (compiled mode)

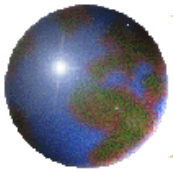


Translating SQL Queries Into Relational Algebra

- A **query block** contains a **single**

  SELECT-FROM-WHERE

  expression (may contain GROUP BY and HAVING)
- Nested queries are not query blocks, but are identified as separate query blocks
- SQL queries are first decomposed into query blocks, then translated into equivalent extended relational algebra expressions



Translating SQL Queries Into Relational Algebra

For example, the following compound query

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     SALARY > (SELECT          MAX(SALARY)
                    FROM EMPLOYEE
                    WHERE           DNO=5);
```

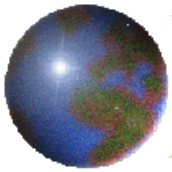
Can be decomposed into two blocks:

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     SALARY > c
```

And

```
SELECT    MAX(SALARY)
FROM      EMPLOYEE
WHERE     DNO=5
```

Where ‘c’ is the result returned from the inner query block.



Translating SQL Queries Into Relational Algebra

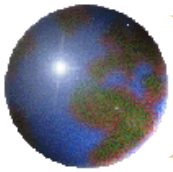
- The inner query block (which need to be calculated first) could be translated into the expression

$$\delta_{\langle \text{MAX SALARY} \rangle} (\sigma_{\langle \text{DNO}=5 \rangle} (\text{EMPLOYEE}))$$

And the outer block into the expression

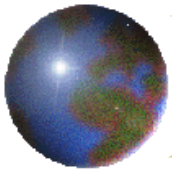
$$\pi_{\langle \text{FNAME, LNAME} \rangle} (\sigma_{\langle \text{SALARY} > c \rangle} (\text{EMPLOYEE}))$$

The query optimizer would then chooses an execution plan for each block.



Algorithms For External Sorting

- The result of a SQL query containing ORDERED BY-clause needs to be sorted
- Internal sorting algorithms are suitable if the whole data fits in main memory.
- External sorting is suitable for large files of records that do not fit entirely in main memory.
- External sorting consists of two main phases:
  - Sorting phase – sort portions of file
  - Merging phase – merge sorted portions



Algorithms For SELECT and JOIN Operations

- Search methods for simple selection (**file scans**):
  - Linear search (brute force)
  - Binary search
  - Using a primary index (or hash key) –e.g.,

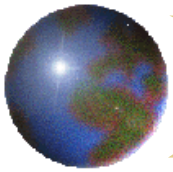
    $\sigma_{<SSN='123456789'>}$ (EMPLOYEE)
  - Using a primary index to retrieve multiple records –

    $\sigma_{<DNUMBER>5>}$ (DEPARTMENT)
  - Using a clustering index to retrieve multiple records

    $\sigma_{<DNO=5>}$ (EMPLOYEE)
  - Using a secondary (B^+–tree) index on an equality comparison.
- Linear search is applicable to any file, but all the other above searches implies having an appropriate access path on the relevant attributes.

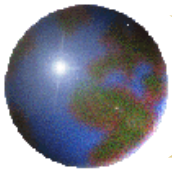
Algorithms For SELECT and JOIN Operations

- Search methods for complex selection (when the SELECT condition as a conjunctive condition –e.g., $\sigma_{<DNO=5\ AND\ Gender='F'>}(EMPLOYEE)$):
  - Conjunctive selection using an individual index
  - Conjunctive selection using a composite index
  - Conjunctive selection by intersection of record pointers
- **Selectivity** is the ratio of the number of records (tuples) that satisfy the condition to the total number of records in the file (relation).
- The optimizer chooses selection conditions with smaller selectivity first to retrieve records.



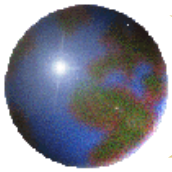
Implementing the JOIN Operation

- Methods for implementing Joins
  - Nested-loop join (brute force) –for each record in outer-loop retrieve every records from inner-loop
  - Single-loop join (using an access structure to retrieve the matching records) –use index to retrieve one record from one relation, and then uses the access structure to retrieve all matching records from the other relation
  - Sort-merge join –it is applicable if both relations are sorted by value of the join attribute
  - Hash-join –it is applicable if records of both relations are hashed to the same hash file, using the same hashing function on the join attribute



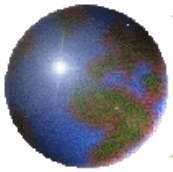
Algorithms For PROJECT and SET Operations

- Implementation of a PROJECT operation is straightforward if <attribute list> includes a key
- If <attribute list> does not include a key, duplicate elimination may be required (requires sorting).
- Implementation of {∪, ∩, –} can be done by using some variations of the sort-merge or hashing techniques
- Implementation of the Cartesian-product is expensive



Aggregate Operations and OUTER JOINS

- The aggregate operations MIN, MAX, COUNT, AVERAGE, and SUM can be computed by scanning the whole records (the worst case)
- If index exists on the attribute of MAX , MIN operation, then these operations can be done in a much more efficient way
- When a GROUP BY clause is used in a query, the aggregate function must be applied separately on each group of tuples



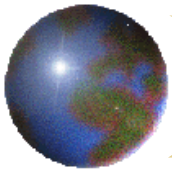
Select dno, avg(salary) from employee Group by dno;

- Sorting or Hashing on the group attribute → dno
- Aggregate function is applied on each group

Outer Join:

Select lname, fname, dname from (employee left outer join department on dno=dnumber)

- Nested loop join or single loop join (left relation as outer loop or single loop)
- Or by relational algebra (inner join)



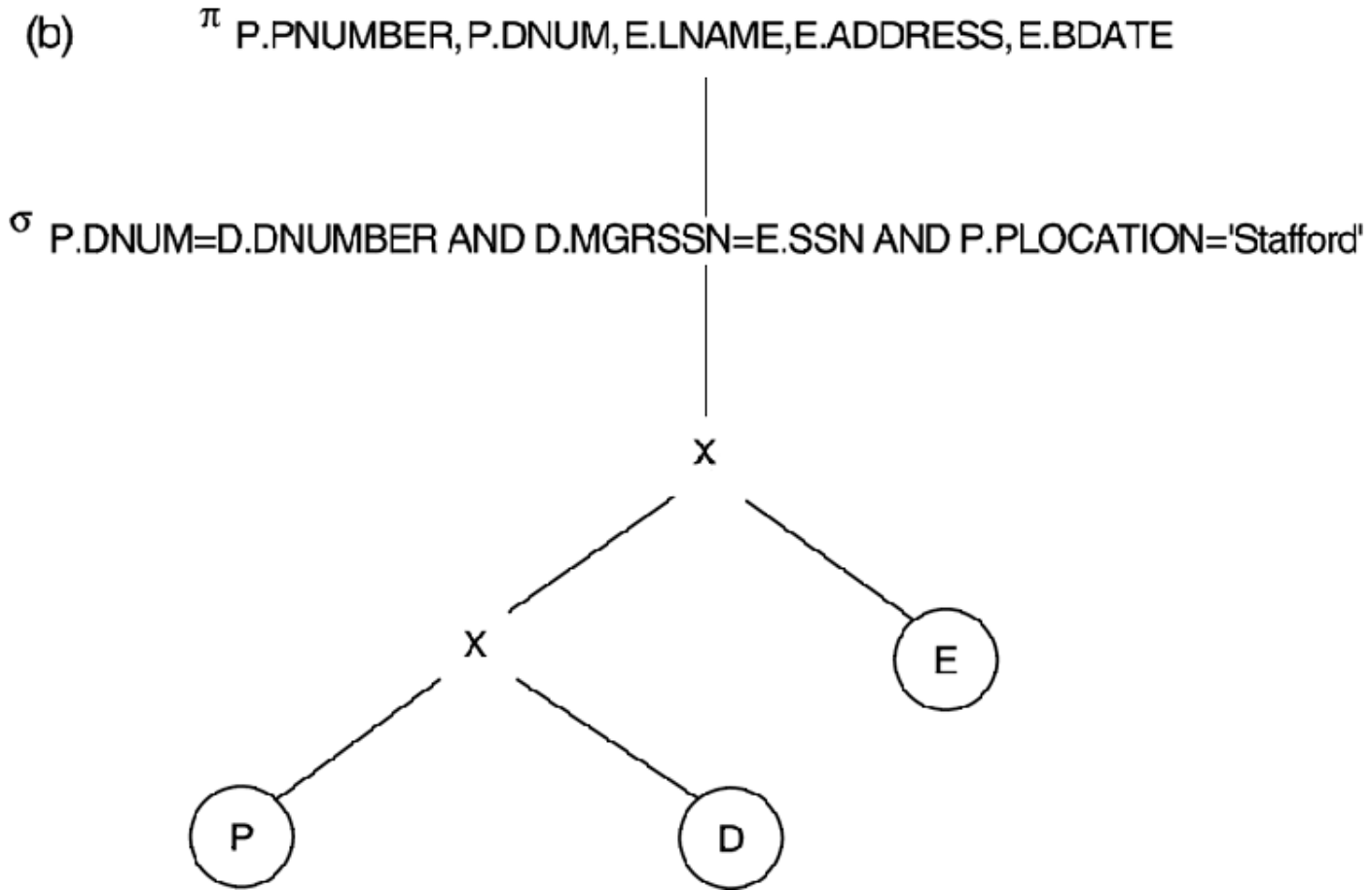
Using Heuristics in Query Optimization

- The parser first generates an initial internal representation, the uses heuristic rules to optimize
- One of the main **heuristic rules** is to apply the unary operations ‘σ’ and ‘π’ before ⋈ or other binary operations
- A **query tree** is a tree data structure that represents the input relations of the query as **leaf nodes** and the relational algebra operations as **internal nodes**.

Initial (canonical) query tree for SQL query

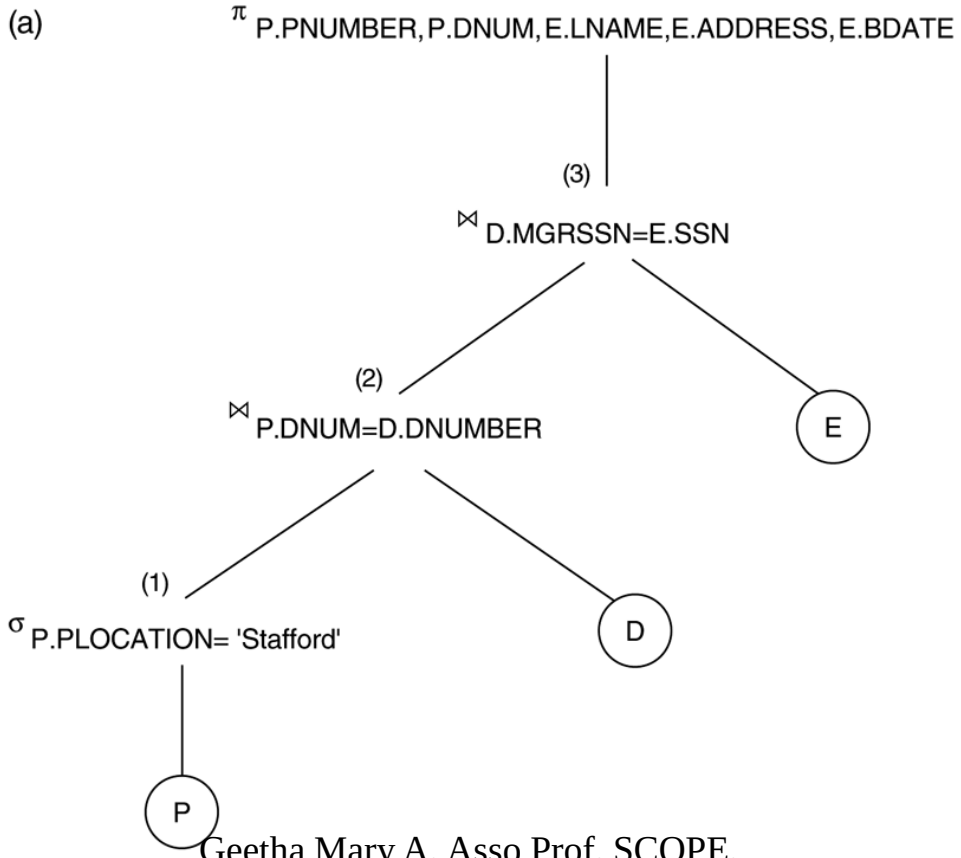
SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND
PLOCATION='Stafford'

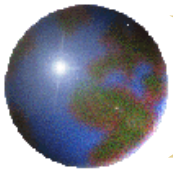
(b)



Query tree corresponding to the relational algebra expression for the SQL query

SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND PLOCATION='Stafford'





Using Heuristics in Query Optimization

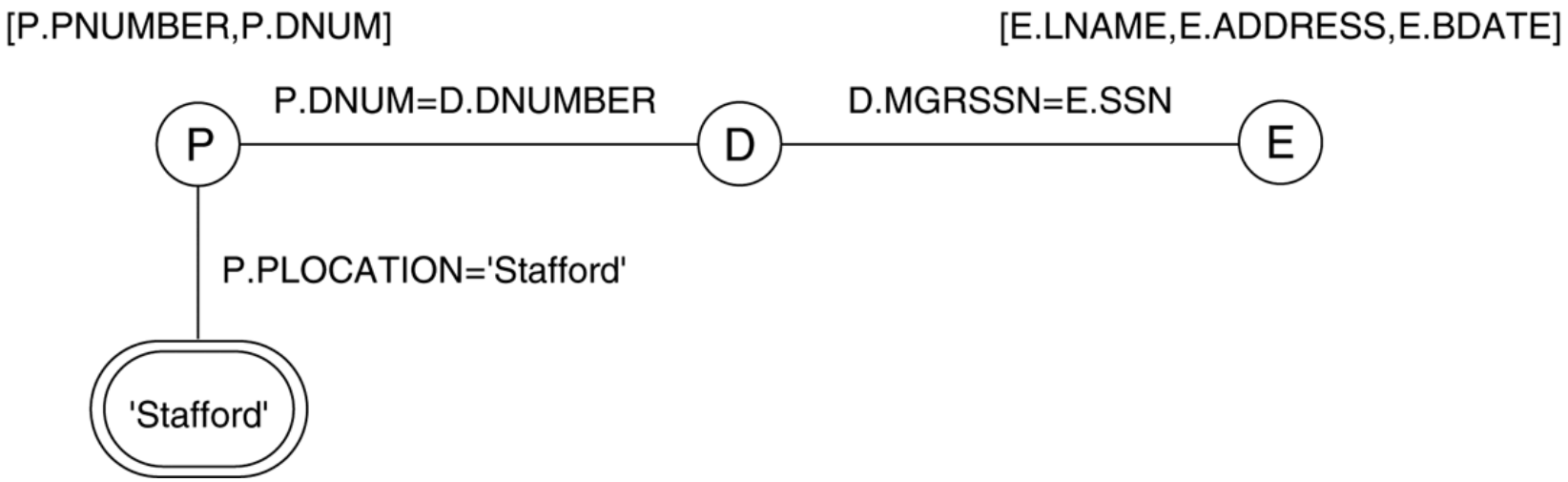
- Execution of the query tree:
  1. Execute an internal node operation whenever its operands are available and then replace the internal node by the resulting operation.
  2. Repeat step 1 as long as there are leaves in the tree, that is, the execution terminates the root node is executed and produces the result relation for the query.
- A more natural representation of a query is the **query graph** notation.

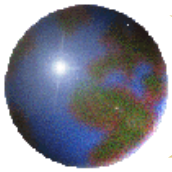
Query graph corresponding to the SQL query

SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE

FROM PROJECT, DEPARTMENT, EMPLOYEE

WHERE DNUM=DNUMBER AND MGRSSN=SSN AND PLOCATION='Stafford'





Using Heuristics in Query Optimization

- Example of Transforming a Query:

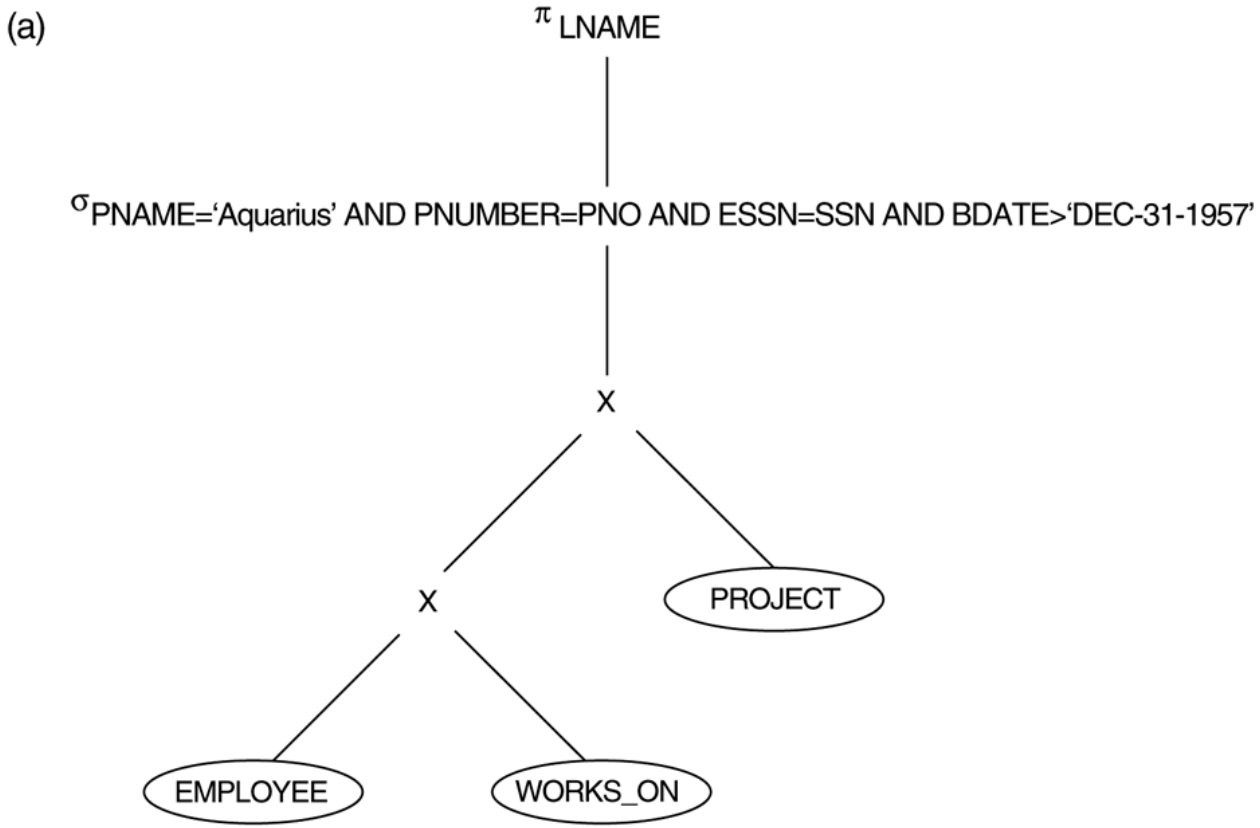
  Consider the query Q that states “Find the last names of employees born after 1957 who work on a project named ‘Aquarius’.”

In SQL, this query can be specified as:

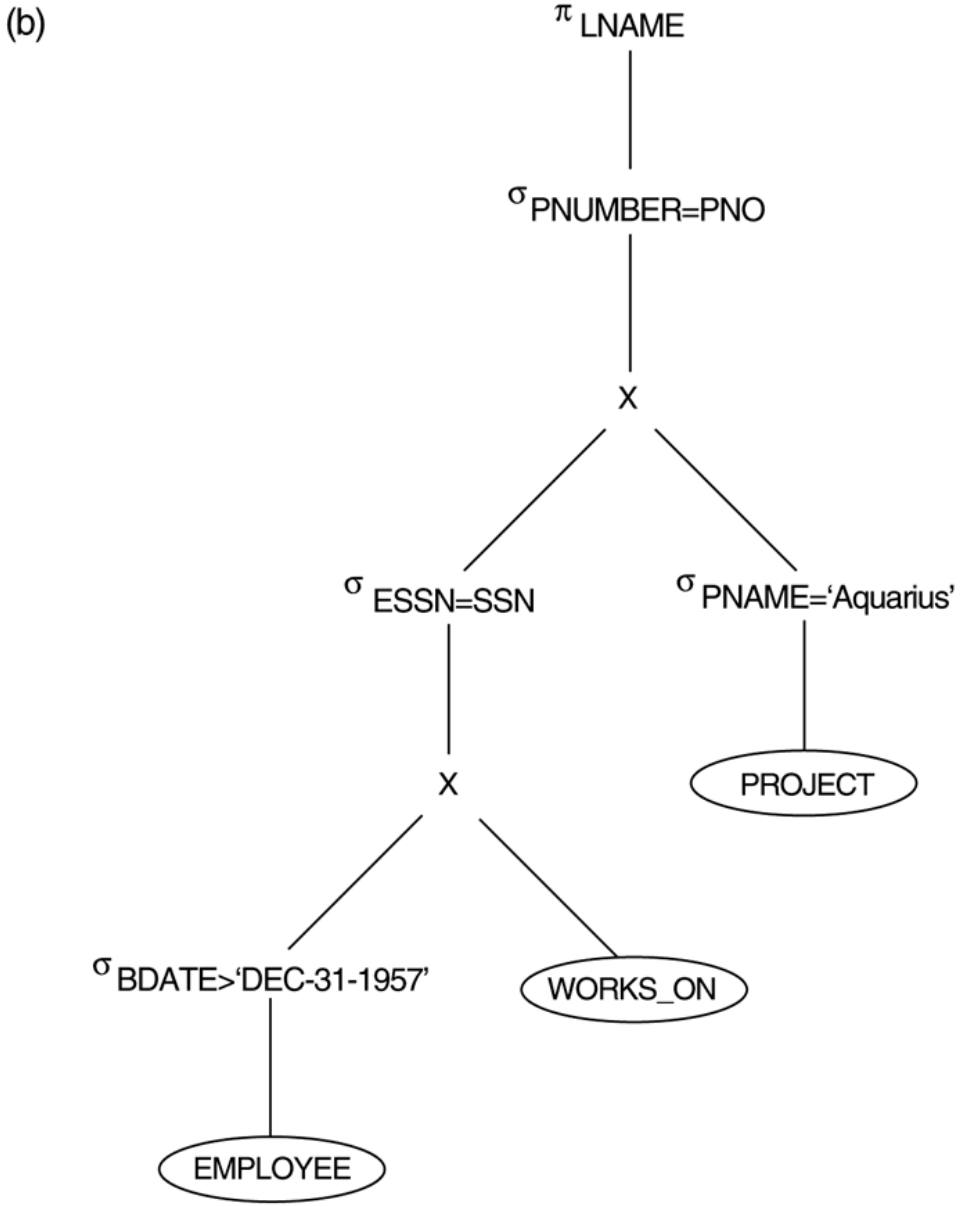
```
SELECT   LNAME  FROM   EMPLOYEE, WORKS_ON, PROJECT
WHERE    PNAME=‘Aquarius’ AND ESSN=SSN
                  AND PNUMBER=PNO
                  AND BDATE > ‘1957-12-31’;
```

Steps in converting a query tree during heuristic optimization.

(a) Initial (canonical) query tree for SQL query Q.

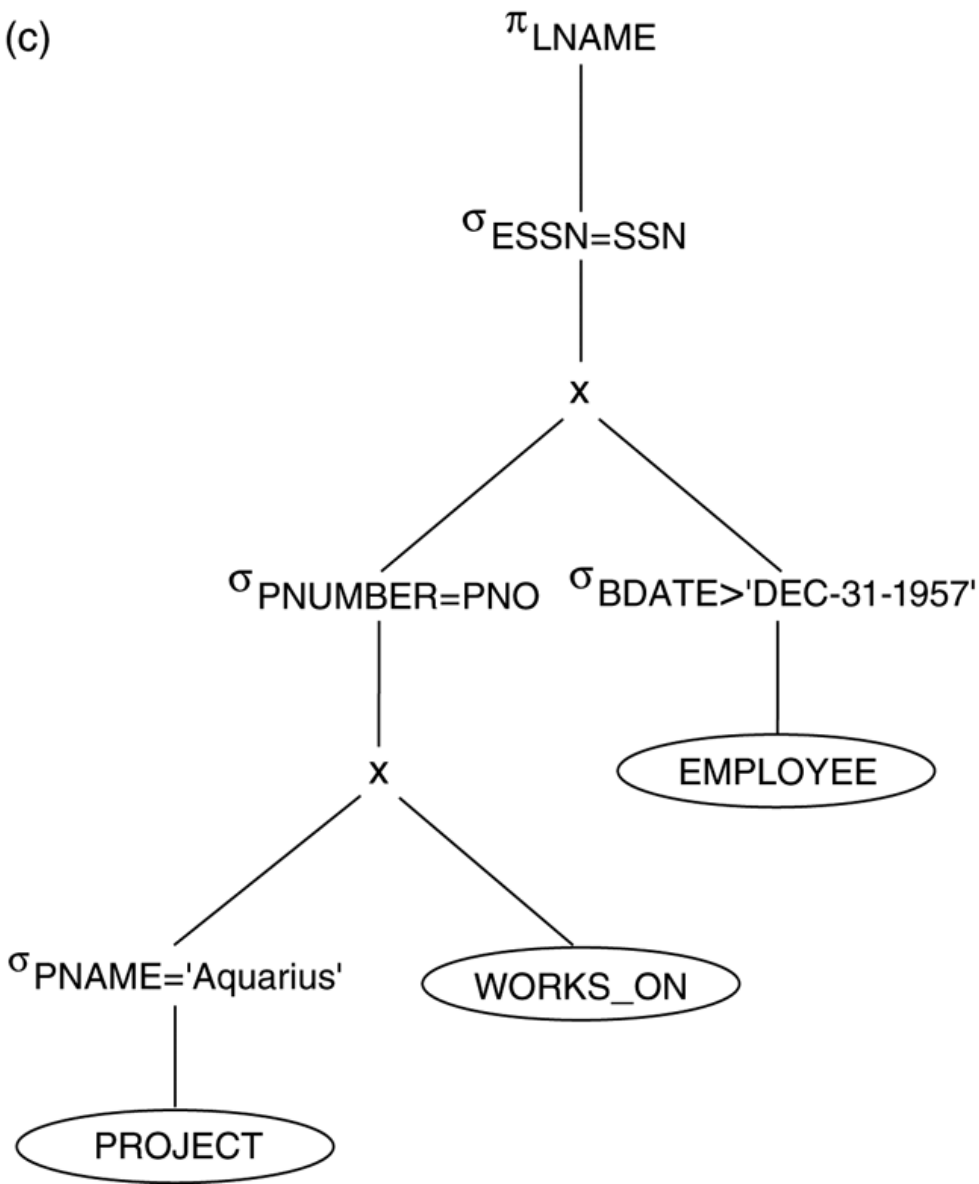


Moving SELECT operations down the query tree.

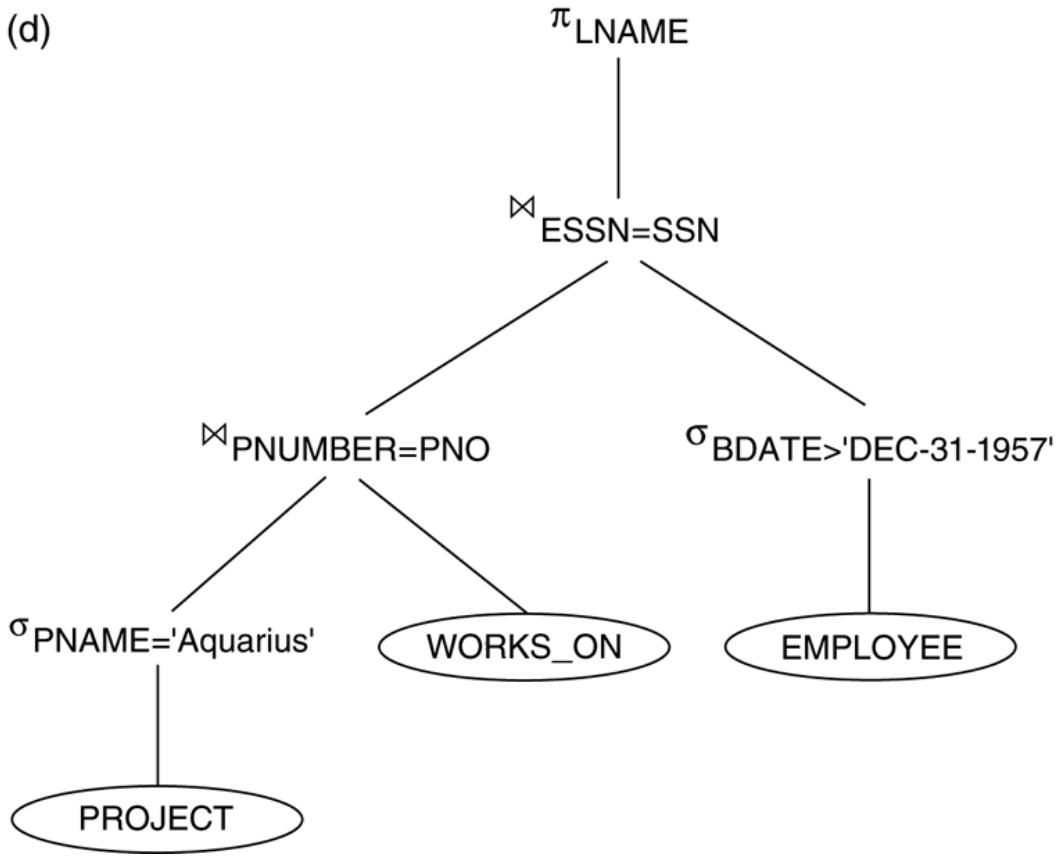


Applying the more restrictive SELECT operation first.

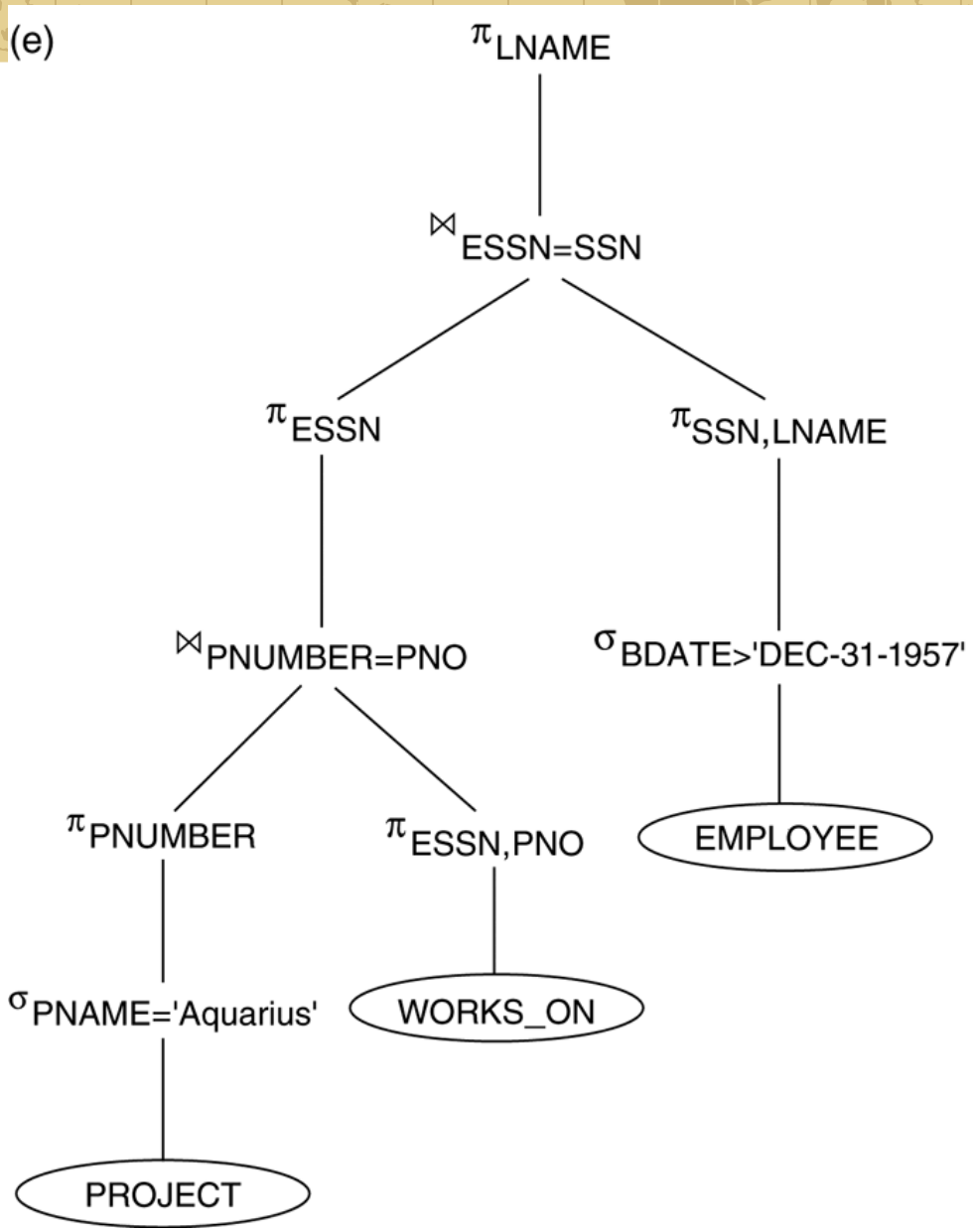
(c)

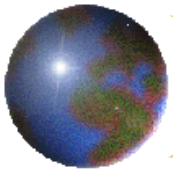


Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.



Moving PROJECT operations down the query tree.





Outline of a Heuristic Algebraic optimization Algorithm

- Break up conjunctive selection condition, that is,

$\sigma_{<c1 \text{ AND } c2 \text{ AND } \ldots \text{ AND } cn>}(R) \equiv \sigma_{<c1>}(\sigma_{<c2>}(\ldots(\sigma_{<cn>}(R))\ldots))$

- Move each 'σ' operation as far down the query tree as is permitted by the attribute involved in the 'σ' condition
- Rearrange the leaf nodes of the tree using;

• Position the leaf node relation with the most restrictive σ operations so they are executed first,

• Make sure that the ordering of leaf nodes does not cause CARTESIAN PRODUCT operations

- Combine a X with a subsequent σ in the tree into a ⋈
- Break down and move lists of projection attributes down the tree as far as possible
- Identify subtrees that represent groups of operations that can be executed by a single algorithm.