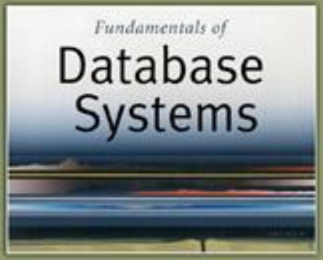
Module 4.3, 4.4

Algorithms for Query Processing and Optimization



5th Edition

Elmasri / Navathe

Chapter Outline (1)

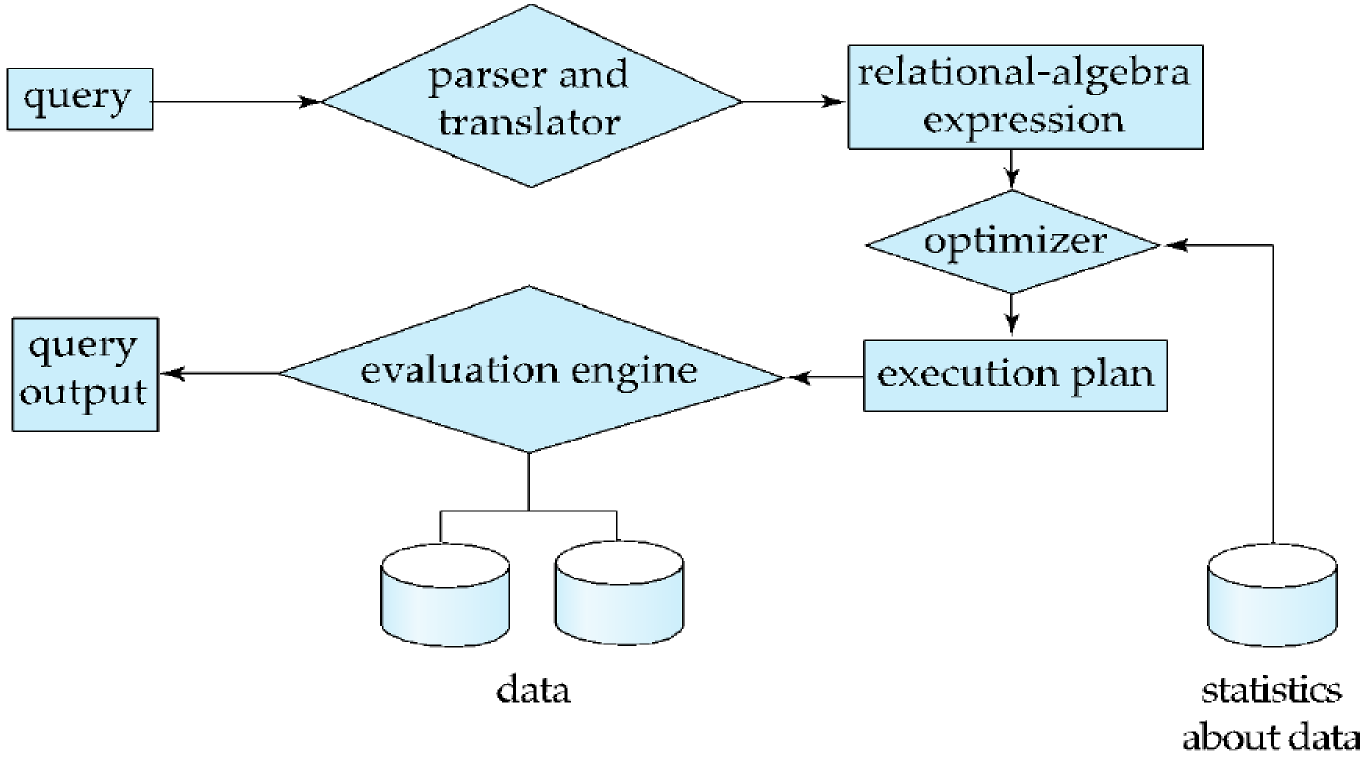
- Introduction to Query Processing
- Translating SQL Queries into Relational Algebra
- Algorithms for SELECT
- Algorithms for PROJECT
- Using Heuristics in Query Optimization

Introduction to Query Processing

Query processing refers to the range of activities involved in extracting data from a database. The activities include translation of queries in high-level database languages into expressions that can be used at the physical level of the file system, a variety of query-optimizing transformations, and actual evaluation of queries.

Basic Steps in Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Basic Steps in Query Processing (Cont.)

- Parsing and translation
  - translate the query into its internal form. This is then translated into relational algebra.
  - Parser checks syntax, verifies relations
- Evaluation
  - The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.

Explanation…

- Each SQL query can itself be translated into a relational-algebra expression in one of several ways.
- The relational-algebra representation of a query specifies only partially how to evaluate a query;
- There are usually several ways to evaluate relational-algebra expressions. As an illustration, consider the query:

select salary

from instructor

where salary < 75000;

This query can be translated into either of the following relational-algebra expressions

- $\sigma_{salary<75000}(\Pi_{salary}(instructor))$
- $\Pi_{salary}(\sigma_{salary<75000}(instructor))$

Explanation.

- we can execute each relational-algebra operation by one of several different algorithms. For example, to implement the preceding selection, we can search every tuple in instructor to find tuples with salary less than 75000. or
- If a B+-tree index is available on the attribute salary, we can use the index instead to locate the tuples. To specify fully how to evaluate a query, we need not only to provide the relational algebra expression, but also to annotate it with instructions specifying how to evaluate each operation.
- Annotations may state the algorithm to be used for a specific operation or the particular index or indices to use.

Explanation..

- A sequence of primitive operations that can be used to evaluate a query is a query-execution plan or query-evaluation plan. Figure illustrates an evaluation plan for our example query, in which a particular index (denoted in the figure as “index 1”) is specified for the selection operation. The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.

Example of query evaluation plan

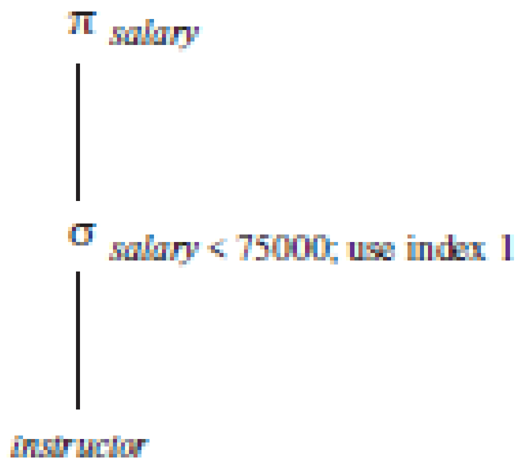


Figure 15.2 A query-evaluation plan.

Explanation….

- The different evaluation plans for a given query can have different costs. It is the responsibility of the system to construct a query-evaluation plan that minimizes the cost of query evaluation; this task is called query optimization. Once the query plan is chosen, the query is evaluated with that plan, and the result of the query is output

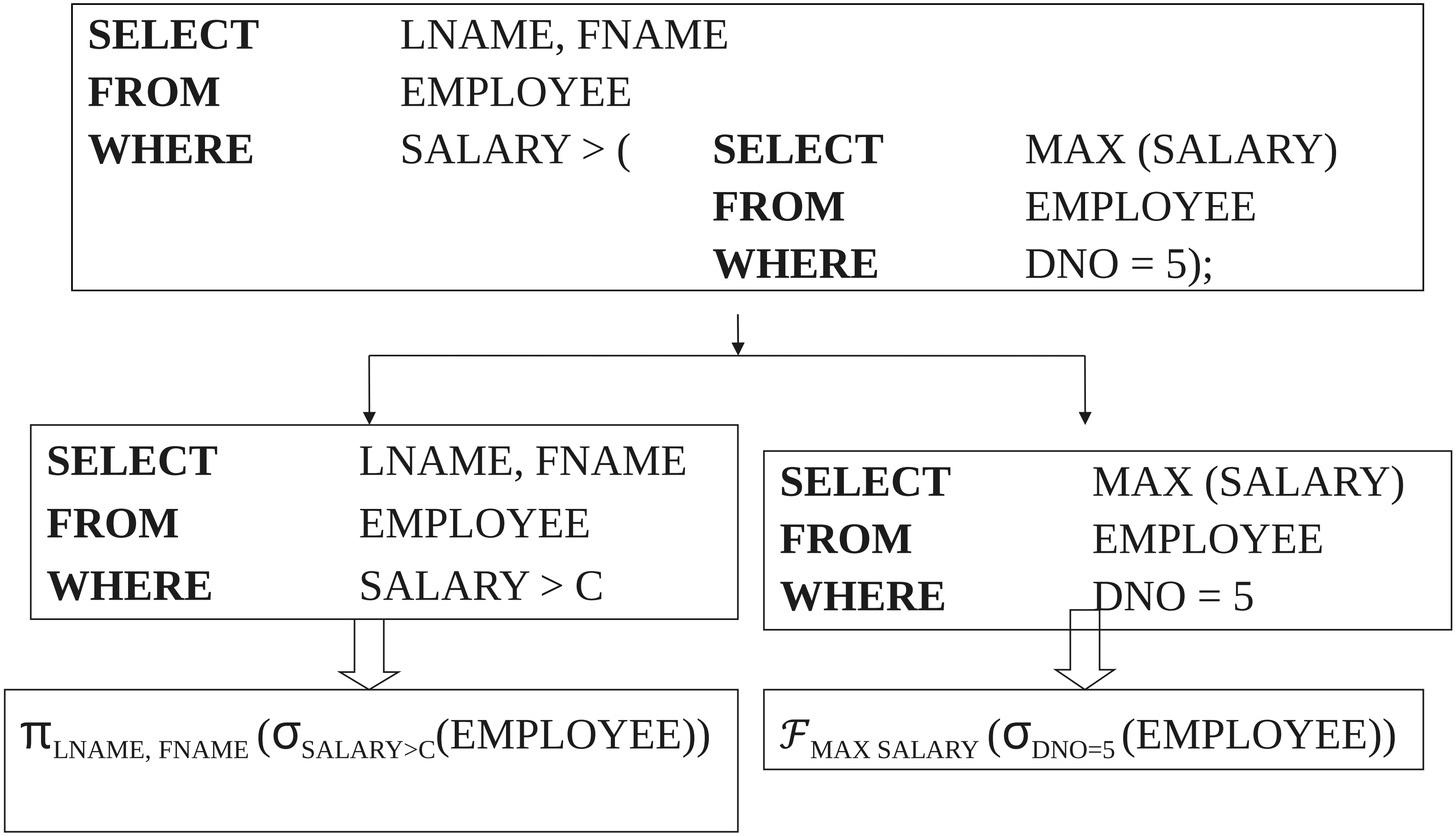
0. Introduction to Query Processing (contd..)

- **Query optimization**:
  - The process of choosing a suitable execution strategy for processing a query.
- Two internal representations of a query:
  - **Query Tree**
  - **Query Graph**

1. Translating SQL Queries into Relational Algebra (1)

- **Query block**:
  - The basic unit that can be translated into the algebraic operators and optimized.
- A query block contains a single SELECT-FROM-WHERE expression, as well as GROUP BY and HAVING clause if these are part of the block.
- **Nested queries** within a query are identified as separate query blocks.
- Aggregate operators in SQL must be included in the extended algebra.

Translating SQL Queries into Relational Algebra (2)



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

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

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

$\pi_{\text{LNAME, FNAME}} (\sigma_{\text{SALARY>C}}(\text{EMPLOYEE}))$

$\mathcal{F}_{\text{MAX SALARY}} (\sigma_{\text{DNO=5}} (\text{EMPLOYEE}))$

Algorithms for SELECT Operations (1)

- Implementing the SELECT Operation

- Examples:
  - (OP1): $\sigma_{\text{SSN='123456789'}}$ (EMPLOYEE)
  - (OP2): $\sigma_{\text{DNUMBER>5}}$(DEPARTMENT)
  - (OP3): $\sigma_{\text{DNO=5}}$(EMPLOYEE)
  - (OP4): $\sigma_{\text{DNO=5 AND SALARY>30000 AND SEX=F}}$(EMPLOYEE)
  - (OP5): $\sigma_{\text{ESSN=123456789 AND PNO=10}}$(WORKS_ON)

Algorithms for SELECT (2)

- Implementing the SELECT Operation (contd.):
- Search Methods for Simple Selection:
  - S1 **Linear search** (brute force):
    - Retrieve every record in the file, and test whether its attribute values satisfy the selection condition.
  - S2 **Binary search**:
    - If the selection condition involves an equality comparison on a key attribute on which the file is ordered, binary search (which is more efficient than linear search) can be used. (See OP1).
  - S3 **Using a primary index or hash key to retrieve a single record**:
    - If the selection condition involves an equality comparison on a key attribute with a primary index (or a hash key), use the primary index (or the hash key) to retrieve the record.

Algorithms for SELECT operations(3)

- Implementing the SELECT Operation (contd.):
- Search Methods for Simple Selection:
  - S4 **Using a primary index to retrieve multiple records**:
    - If the comparison condition is >, ≥, <, or ≤ on a key field with a primary index, use the index to find the record satisfying the corresponding equality condition, then retrieve all subsequent records in the (ordered) file.
  - S5 **Using a clustering index to retrieve multiple records**:
    - If the selection condition involves an equality comparison on a non-key attribute with a clustering index, use the clustering index to retrieve all the records satisfying the selection condition.
  - S6 **Using a secondary (B+-tree) index**:
    - On an equality comparison, this search method can be used to retrieve a single record if the indexing field has unique values (is a key) or to retrieve multiple records if the indexing field is not a key.
    - In addition, it can be used to retrieve records on conditions involving >,>=, <, or <=. (FOR RANGE QUERIES)

Algorithms for SELECT Operations (4)

- Implementing the SELECT Operation (contd.):
- Search Methods for Simple Selection:
  - S7 **Conjunctive selection**:
    - If an attribute involved in any single simple condition in the conjunctive condition has an access path that permits the use of one of the methods S2 to S6, use that condition to retrieve the records and then check whether each retrieved record satisfies the remaining simple conditions in the conjunctive condition.
  - S8 **Conjunctive selection using a composite index**
    - If two or more attributes are involved in equality conditions in the conjunctive condition and a composite index (or hash structure) exists on the combined field, we can use the index directly.

Algorithms for SELECT Operations (5)

- Implementing the SELECT Operation (contd.):
- Search Methods for Complex Selection:
  - S9 **Conjunctive selection by intersection of record pointers**:
    - This method is possible if secondary indexes are available on all (or some of) the fields involved in equality comparison conditions in the conjunctive condition and if the indexes include record pointers (rather than block pointers).
    - Each index can be used to retrieve the record pointers that satisfy the individual condition.
    - The intersection of these sets of record pointers gives the record pointers that satisfy the conjunctive condition, which are then used to retrieve those records directly.
    - If only some of the conditions have secondary indexes, each retrieved record is further tested to determine whether it satisfies the remaining conditions.

Algorithms for SELECT Operations (7)

- Implementing the SELECT Operation (contd.):
  - Whenever a **single condition** specifies the selection, we can only check whether an access path exists on the attribute involved in that condition.
    - If an access path exists, the method corresponding to that access path is used; otherwise, the “brute force” linear search approach of method S1 is used. (See OP1, OP2 and OP3)
  - For **conjunctive selection conditions**, whenever *more than one* of the attributes involved in the conditions have an access path, query optimization should be done to choose the access path that *retrieves the fewest records* in the most efficient way.
  - **Disjunctive selection conditions**

4. Algorithms for PROJECT Operations (1)

- Algorithm for PROJECT operations (Figure 15.3b)

  $\pi_{\text{<attribute list>}}(R)$

  1. If <attribute list> has a key of relation R, extract all tuples from R with only the values for the attributes in <attribute list>.
  2. 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

7. Using Heuristics in Query Optimization(1)

- Process for heuristics optimization
  1. The parser of a high-level query generates an initial internal representation of query;
  2. Apply heuristics rules to optimize the internal representation.
  3. A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

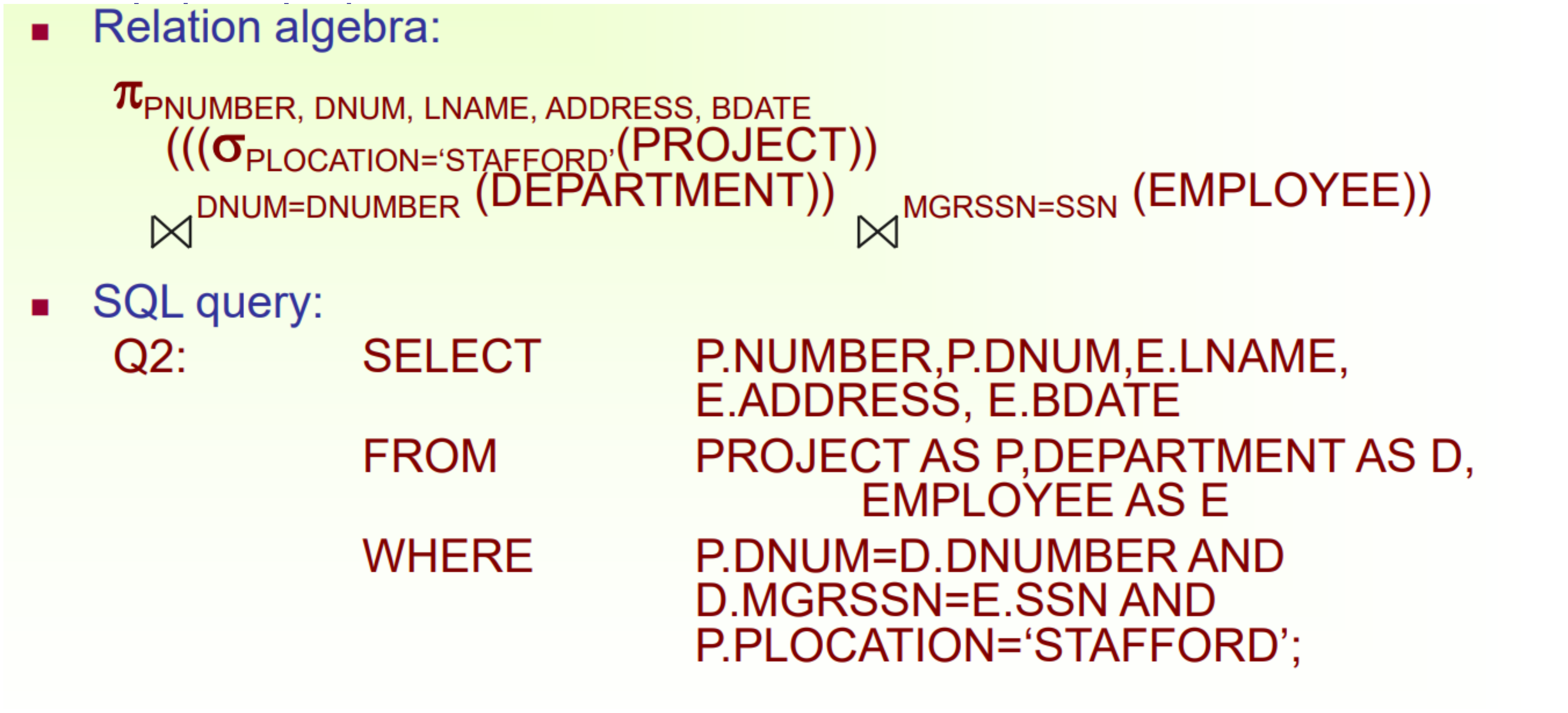
- The main heuristic is to apply first the operations that reduce the size of intermediate results.
  - E.g., Apply SELECT and PROJECT operations before applying the JOIN or other binary operations.

Using Heuristics in Query Optimization (2)

- **Query tree**:
  - A tree data structure that corresponds to a relational algebra expression. It represents the input relations of the query as **leaf nodes** of the **tree**, and represents the relational algebra operations as internal nodes.
- An execution of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation.

Using Heuristics in Query Optimization (3)

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



- Relation algebra:

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

- SQL query:

```
Q2:        SELECT      P.NUMBER,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';
```

Using Heuristics in Query Optimization (4)

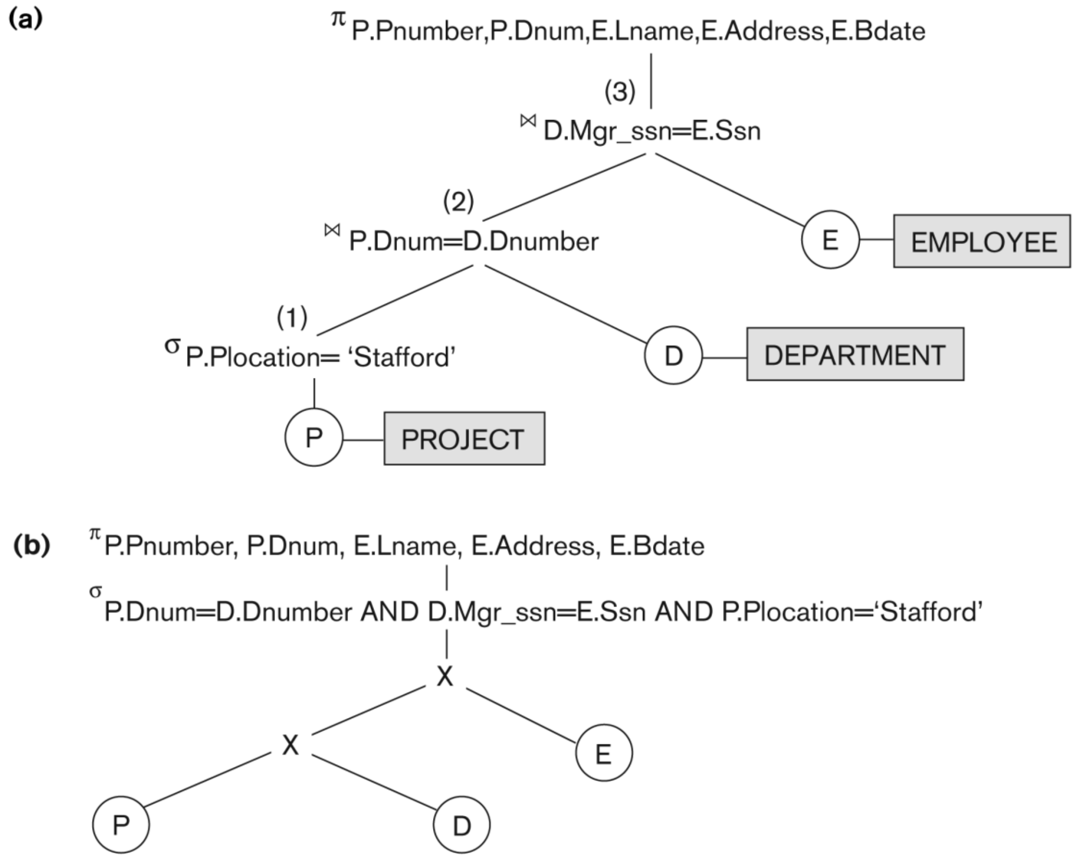

(a)
π P.Pnumber,P.Dnum,E.Lname,E.Address,E.Bdate
(3)
⋈ D.Mgr_ssn=E.Ssn
(2)
⋈ P.Dnum=D.Dnumber
E
EMPLOYEE
(1)
σ P.Plocation= 'Stafford'
D
DEPARTMENT
P
PROJECT
(b)
π P.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate
σ P.Dnum=D.Dnumber AND D.Mgr_ssn=E.Ssn AND P.Plocation='Stafford'
X
X
E
P
D

Figure 15.4
Two query trees for the query Q2. (a) Query tree corresponding to the relational algebra expression for Q2. (b) Initial (canonical) query tree for SQL query Q2. (c) Query graph for Q2.

Using Heuristics in Query Optimization (6)

- Heuristic Optimization of Query Trees:
  - The same query could correspond to many different relational algebra expressions — and hence many different query trees.
  - The task of heuristic optimization of query trees is to find a **final query tree** that is efficient to execute.
- Example:

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

Using Heuristics in Query Optimization (7)

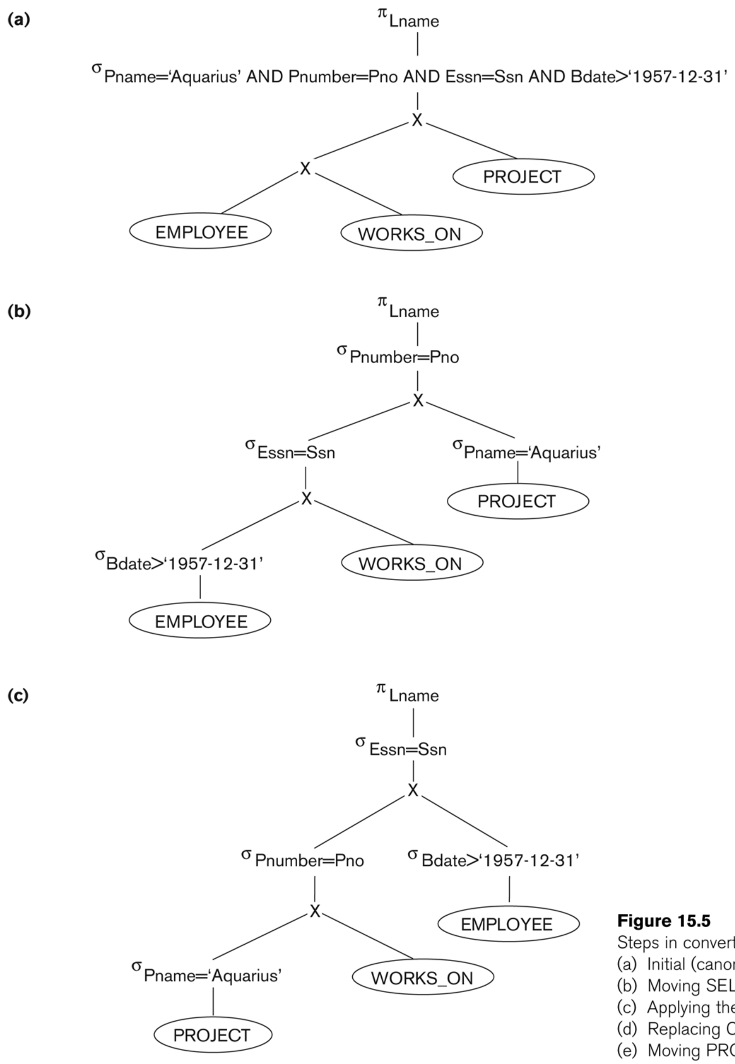
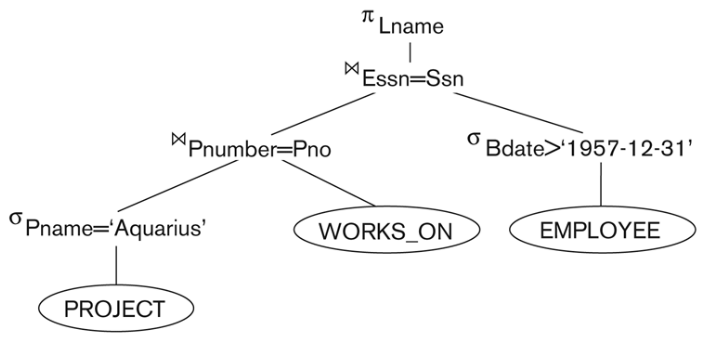


Figure 15.5
Steps in converting a query tree during heuristic optimization.
(a) Initial (canonical) query tree for SQL query Q.
(b) Moving SELECT operations down the query tree.
(c) Applying the more restrictive SELECT operation first.
(d) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.
(e) Moving PROJECT operations down the query tree.

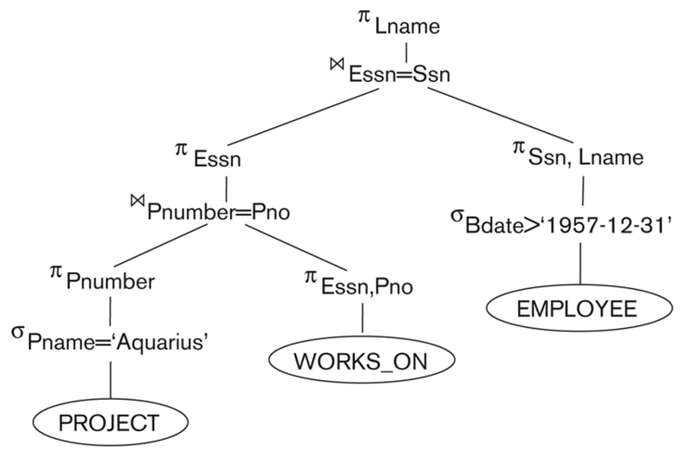
Using Heuristics in Query Optimization (8)

(d)



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

(e)



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

Figure 15.5
Steps in converting a query tree during heuristic optimization.
(a) Initial (canonical) query tree for SQL query Q.
(b) Moving SELECT operations down the query tree.
(c) Applying the more restrictive SELECT operation first.
(d) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.
(e) Moving PROJECT operations down the query tree.

Using Heuristics in Query Optimization (15)

- Summary of Heuristics for Algebraic Optimization:
  1. The main heuristic is to apply first the operations that reduce the size of intermediate results.
  2. Perform select operations as early as possible to reduce the number of tuples and perform project operations as early as possible to reduce the number of attributes. (This is done by moving select and project operations as far down the tree as possible.)
  3. The select and join operations that are most restrictive should be executed before other similar operations. (This is done by reordering the leaf nodes of the tree among themselves and adjusting the rest of the tree appropriately.)