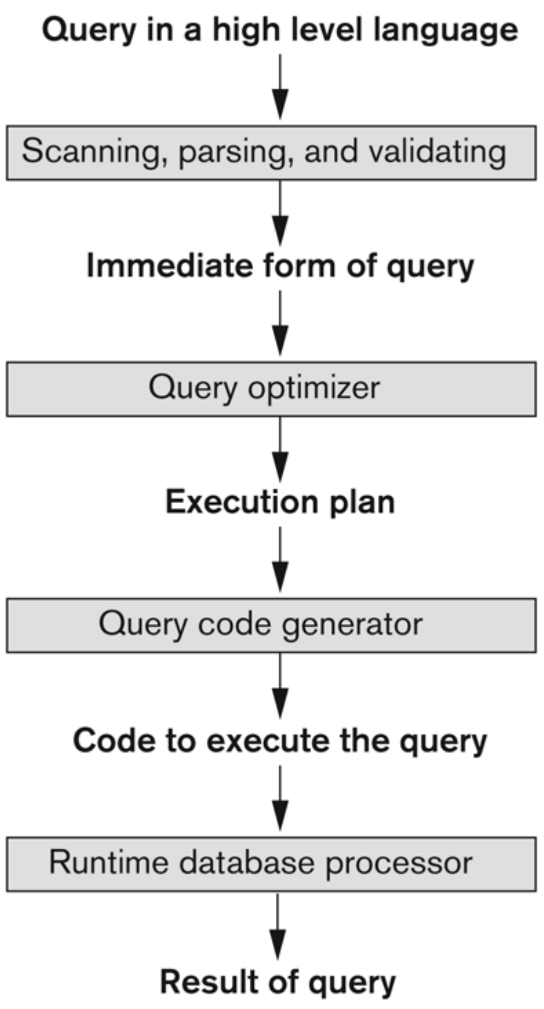
Algorithms for Query Processing and Optimization

Outline

Introduction to Query Processing

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

Introduction to Query Processing


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

Code can be:

Executed directly (interpreted mode)

Stored and executed later whenever needed (compiled mode)

Figure 15.1
Typical steps when processing a high-level query.

Translating SQL Queries into Relational Algebra

- **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

```
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
```

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

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

Algorithms for External Sorting

- **External sorting**:
  - Refers to sorting algorithms that are suitable for large files of records stored on disk that do not fit entirely in main memory, such as most database files.
- **Sort-Merge strategy**:
  - Starts by sorting small subfiles (**runs**) of the main file and then merges the sorted runs, creating larger sorted subfiles that are merged in turn.
  - Sorting phase: $n_R = (b/n_B)$
  - Merging phase: $d_M = \min (n_B-1, n_R)$; $n_P = (\log_{dM}(n_R))$
  - n_R: number of initial runs; b: number of file blocks;
  - n_B: available buffer space; d_M: degree of merging;
  - n_P: number of passes.

Algorithms for SELECT and JOIN Operations

- Implementing the SELECT Operation

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

Algorithms for SELECT and JOIN Operations

- 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.
  - 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 and JOIN Operations

- 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 (ordered but having duplicate values) 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)

Using Heuristics in Query Optimization

- Process for heuristics optimization
  1. The parser of a high-level query generates an initial internal representation;
  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

- **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.
- **Query graph**:
  - A graph data structure that corresponds to a relational calculus expression. It does *not* indicate an order on which operations to perform first. There is only a *single* graph corresponding to each query.

Using Heuristics in Query Optimization

- 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:

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}\,(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$

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

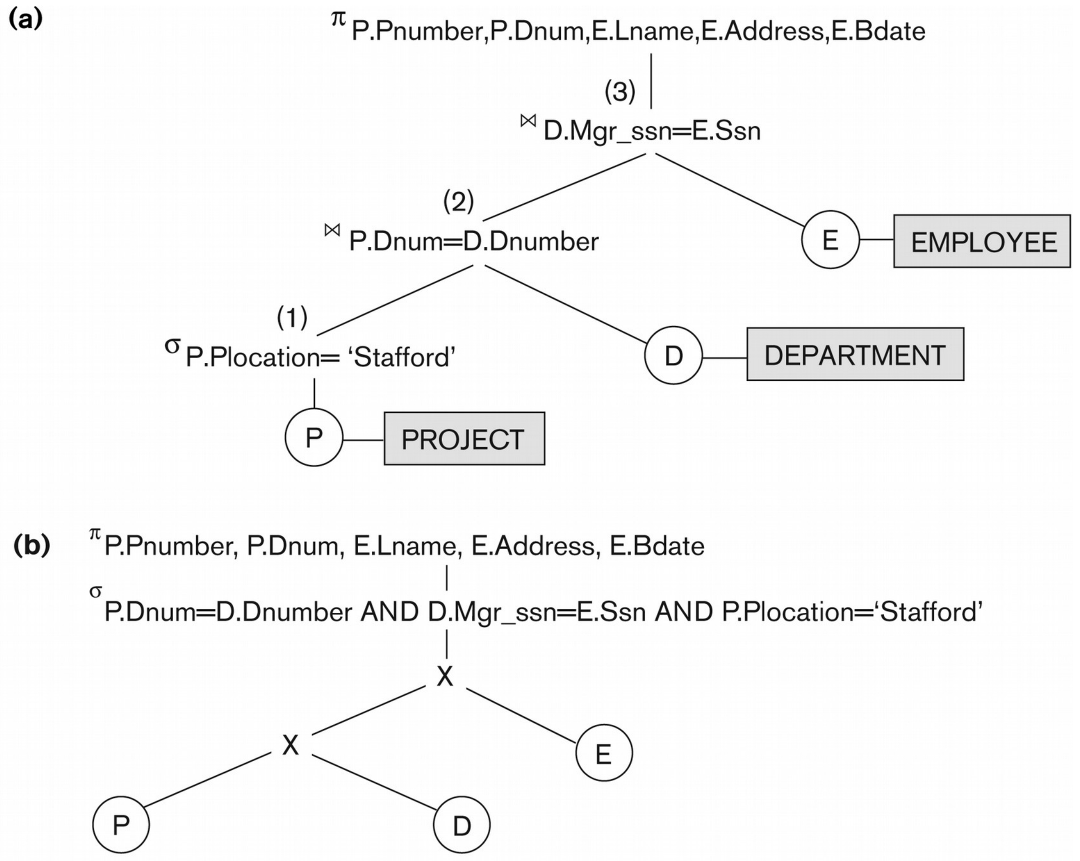

(a)
π P.Pnumber,P.Dnum,E.Lname,E.Address,E.Bdate
(3)
⋈ D.Mgr_ssn=E.Ssn
(2)
⋈ P.Dnum=D.Dnumber
E
EMPLOYEE
(1)
σ P.Plocation= 'Stafford'
D
DEPARTMENT
P
PROJECT
(b)
π P.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate
σ P.Dnum=D.Dnumber AND D.Mgr_ssn=E.Ssn AND P.Plocation='Stafford'
X
X
E
P
D

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

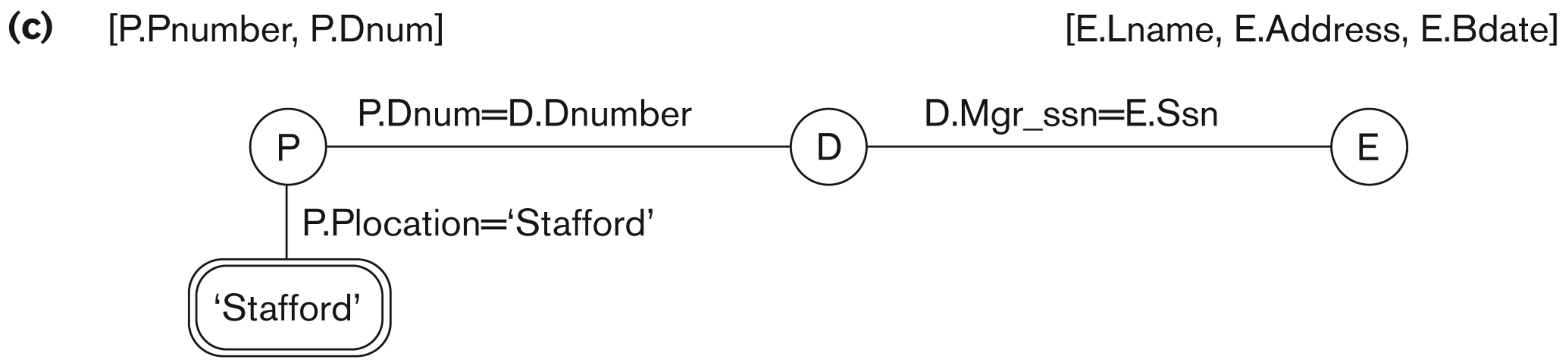


Figure 15.4
Two query trees for the query Q2. (a) Query tree corresponding to the relational algebra expression for Q2. (b) Initial (canonical) query tree for SQL query Q2. (c) Query graph for Q2.

Using Heuristics in Query Optimization

- 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

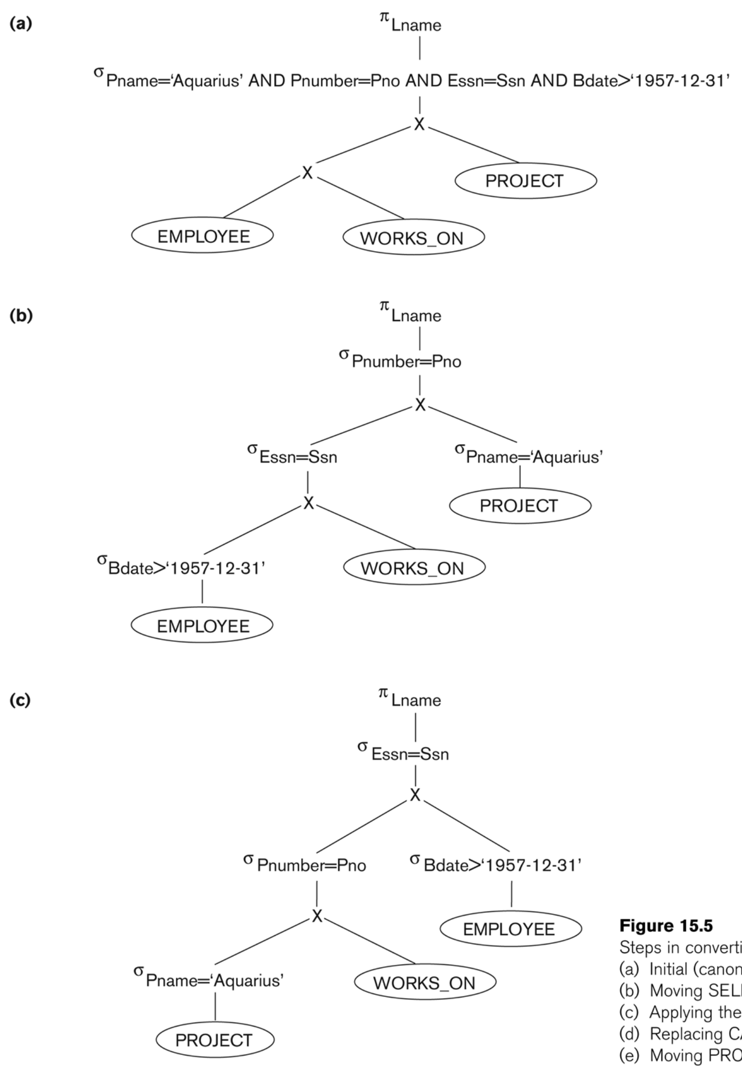
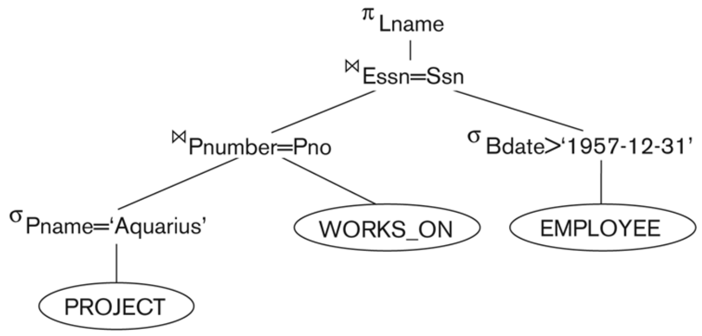

(a)
π_{Lname}
$\sigma_{\text{Pname='Aquarius' AND Pnumber=Pno AND Essn=Ssn AND Bdate>'1957-12-31'}}$
X
X
PROJECT
EMPLOYEE
WORKS_ON
(b)
π_{Lname}
$\sigma_{\text{Pnumber=Pno}}$
X
$\sigma_{\text{Essn=Ssn}}$
$\sigma_{\text{Pname='Aquarius'}}$
PROJECT
X
$\sigma_{\text{Bdate>'1957-12-31'}}$
WORKS_ON
EMPLOYEE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

(d)



(e)

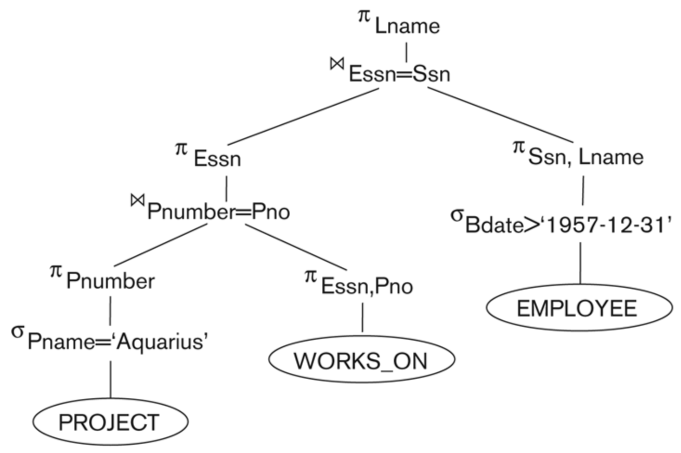


Figure 15.5
Steps in converting a query tree during heuristic optimization.
(a) Initial (canonical) query tree for SQL query Q.
(b) Moving SELECT operations down the query tree.
(c) Applying the more restrictive SELECT operation first.
(d) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.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

Using Heuristics in Query Optimization

- 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.)

Using Cost Estimates in Query Optimization

- **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.
  - (Compare to heuristic query optimization)

- Issues
  - Cost function
  - Number of execution strategies to be considered

Using Cost Estimates in Query Optimization

- Cost Components for Query Execution
  1. Access cost to secondary storage
  2. Storage cost
  3. Computation cost
  4. Memory usage cost
  5. Communication cost

- Note: Different database systems may focus on different cost components.

Using Cost Estimates in Query Optimization

- Catalog Information Used in Cost Functions
  - Information about the size of a file
    - number of records (tuples) (r),
    - record size (R),
    - number of blocks (b)
    - blocking factor (bfr)
  - Information about indexes and indexing attributes of a file
    - Number of levels (x) of each multilevel index
    - Number of first-level index blocks (bI1)
    - Number of distinct values (d) of an attribute
    - Selectivity (sl) of an attribute
    - Selection cardinality (s) of an attribute. (s = sl * r)

Using Cost Estimates in Query Optimization

- Examples of Cost Functions for SELECT
- S1. Linear search (brute force) approach
  - $C_{S1a} = b$;
  - For an equality condition on a key, $C_{S1a} = (b/2)$ if the record is found; otherwise $C_{S1a} = b$.
- S2. Binary search:
  - $C_{S2} = \log_2 b + (s/bfr) - 1$
  - For an equality condition on a unique (key) attribute, $C_{S2} = \log_2 b$
- S3. Using a primary index (S3a) or hash key (S3b) to retrieve a single record
  - $C_{S3a} = x + 1$; $C_{S3b} = 1$ for static or linear hashing;
  - $C_{S3b} = 1$ for extendible hashing;

Using Cost Estimates in Query Optimization

- Examples of Cost Functions for SELECT (contd.)
- S4. Using an ordering index to retrieve multiple records:
  - For the comparison condition on a key field with an ordering index, $C_{S4} = x + (b/2)$
- S5. Using a clustering index to retrieve multiple records:
  - $C_{S5} = x + (s/bfr)$
- S6. Using a secondary (B+-tree) index:
  - For an equality comparison, $C_{S6a} = x + s$;
  - For an comparison condition such as >, <, >=, or <=,
  - $C_{S6a} = x + (b_{I1}/2) + (r/2)$

Query Optimization in Oracle

- Oracle DBMS V8
  - **Rule-based query optimization**: the optimizer chooses execution plans based on heuristically ranked operations.
    - (Currently it is being phased out)
  - **Cost-based query optimization**: the optimizer examines alternative access paths and operator algorithms and chooses the execution plan with lowest estimate cost.
    - The query cost is calculated based on the estimated usage of resources such as I/O, CPU and memory needed.
  - Application developers could specify hints to the ORACLE query optimizer.
  - The idea is that an application developer might know more information about the data.