Query Processing and Optimization

Module 1

IT – TE – ADMT(DLO)

Prof. Seema S. Redekar
Assistant Professor
Dept. of Information Technology,
SIES Graduate School of Technology

Learning Objectives

Lecture No:4

- To understand steps in query processing.
- To learn different relational algebra operations used in the query processing.
- To study algorithms used for different operations in the query processing.
- To understand different query execution strategies.
- To study query optimization techniques.

Query Processing Steps

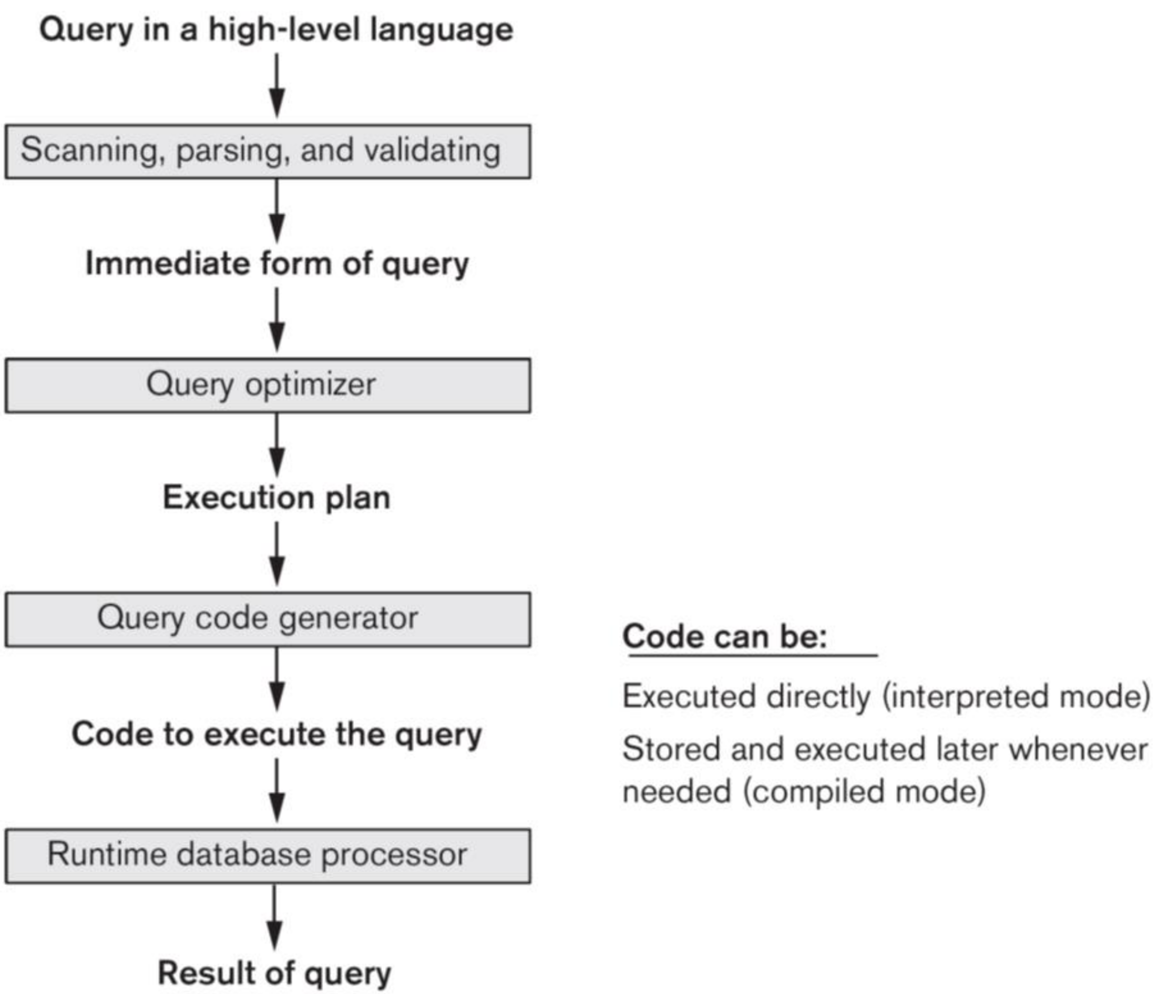


Figure 19.1
Typical steps when processing a high-level query.

Source: Fundamentals of Database System, Elmasri, Navathe

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 SELECT and JOIN Operations

- Implementing the SELECT Operation

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

Algorithms for SELECT and JOIN Operations

- Search Methods for Simple Selection:
  - S1 **Linear search** (brute force):

  - 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

- 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.

    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.
  - 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.

Algorithms for SELECT and JOIN Operations

- Search Methods for Simple Selection:
  - S7 **Conjunctive selection**:
    - If an attribute involved in any single simple condition 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 and JOIN Operations

- 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**

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: $n_R = \lceil (b/n_B) \rceil$
  - Merging phase: $d_M = \text{Min}\ (n_B - 1, n_R)$; $n_P = \lceil (\log_{dM}(n_R)) \rceil$
  - 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.

Learning Objectives

Lecture No:5

- To learn different relational algebra operations used in the query processing.
- To study algorithms used for different operations in the query processing.

Algorithms for SELECT and JOIN Operations

- Implementing the JOIN Operation:
  - Join (NATURAL JOIN)
    - two–way join: a join on two files
    - e.g. $R \bowtie_{A=B} S$
    - multi-way joins: joins involving more than two files.
    - e.g. $R \bowtie_{A=B} S \bowtie_{C=D} T$
- Examples
  - (OP6): EMPLOYEE $\bowtie_{DNO=DNUMBER}$ DEPARTMENT
  - (OP7): DEPARTMENT $\bowtie_{MGRSSN=SSN}$ EMPLOYEE

Algorithms for SELECT and JOIN Operations

- Implementing the JOIN Operation
- Methods for implementing joins:
  - J1 **Nested-loop join** (brute force):
    - For each record t in R (outer loop), retrieve every record s from S (inner loop) and test whether the two records satisfy the join condition t[A] = s[B].
  - J2 **Single-loop join** (Using an access structure to retrieve the matching records):
    - If an index (or hash key) exists for one of the two join attributes — say, B of S — retrieve each record t in R, one at a time, and then use the access structure to retrieve directly all matching records s from S that satisfy s[B] = t[A].

Algorithms for SELECT and JOIN Operations

- Methods for implementing joins:
  - J3 **Sort-merge join**:
    - If the records of R and S are *physically sorted* (*ordered*) by value of the join attributes A and B, respectively, we can implement the join in the most efficient way possible.
    - Both files are scanned in order of the join attributes, matching the records that have the same values for A and B.
    - In this method, the records of each file are scanned only once each for matching with the other file—unless both A and B are non-key attributes, in which case the method needs to be modified slightly.

Algorithms for SELECT and JOIN Operations

- Implementing the JOIN Operation (contd.):
- Factors affecting JOIN performance
  - Available buffer space
  - Join selection factor
  - Choice of inner VS outer relation

Algorithms for PROJECT and SET Operations

- Algorithm for PROJECT operations
- $\pi_{\text{<attribute list>}}(R)$

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

Algorithms for PROJECT and SET Operations

- **Algorithm for SET operations**
- **Set operations**:
  - UNION, INTERSECTION, SET DIFFERENCE and CARTESIAN PRODUCT
- **CARTESIAN PRODUCT** of relations R and S include all possible combinations of records from R and S. The attribute of the result include all attributes of R and S.
- **Cost analysis** of CARTESIAN PRODUCT
  - If R has n records and j attributes and S has m records and k attributes, the result relation will have n*m records and j+k attributes.
- CARTESIAN PRODUCT operation is **very expensive** and should be avoided if possible.

Algorithms for PROJECT and SET Operations

- **Algorithm for SET operations (contd.)**
- **UNION**
  - Sort the two relations on the same attributes.
  - Scan and merge both sorted files concurrently, whenever the same tuple exists in both relations, only one is kept in the merged results.
- **INTERSECTION**
  - Sort the two relations on the same attributes.
  - Scan and merge both sorted files concurrently, keep in the merged results only those tuples that appear in both relations.
- **SET DIFFERENCE R-S**
  - Keep in the merged results only those tuples that appear in relation R but not in relation S.

Implementing Aggregate Operations and Outer Joins

- Implementing Aggregate Operations:
- **Aggregate operators**:
  - **MIN, MAX, SUM, COUNT** and **AVG**
- Options to implement aggregate operators:
  - **Table Scan**
  - **Index**
- Example
  - SELECT MAX (SALARY)
    FROM EMPLOYEE;

Implementing Aggregate Operations and Outer Joins

- **Outer Join Operators**:
  - **LEFT OUTER JOIN**
  - **RIGHT OUTER JOIN**
  - **FULL OUTER JOIN**.

Q. The full outer join produces a result which is equivalent to the—

a. Intersection of the results of the left and right outer joins.

b. Union of the results of the left and right outer joins.

c. Difference of the results of the left and right outer joins.

d. All of the above

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.

- **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.

Examples of Queries in Relational Algebra

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

$$\text{RESEARCH\_DEPT} \leftarrow \sigma_{\text{DNAME='Research'}}(\text{DEPARTMENT})$$

$$\text{RESEARCH\_EMPS} \leftarrow (\text{RESEARCH\_DEPT} \bowtie_{\text{DNUMBER=DNOEMPLOYEE}} \text{EMPLOYEE})$$

$$\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, ADDRESS}}(\text{RESEARCH\_EMPS})$$

Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain calculus**).

- A simple tuple relational calculus query is of the form

  {t | COND(t)}

  where t is a tuple variable and COND (t) is a conditional expression involving t. The result of such a query is the set of all tuples t that satisfy COND (t).

  Example: To find the first and last names of all employees whose salary is above $50,000, we can write the following tuple calculus expression:

  {t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}

Learning Objectives

Lecture No:6

- To study algorithms used for different operations in the query processing.

Company Relational Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

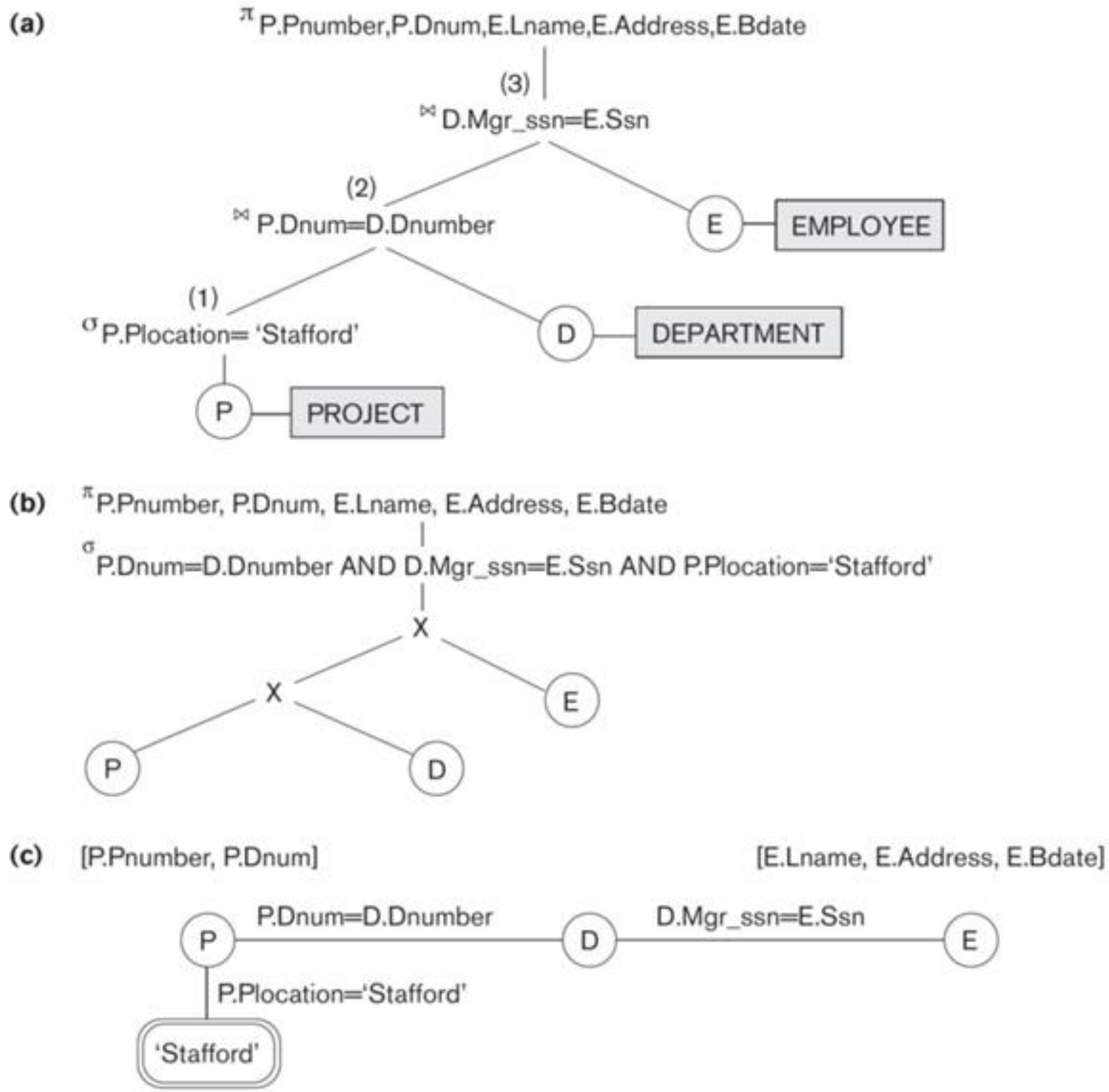
Using Heuristics in Query Optimization

- 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:

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

- 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';
```



(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
(c)
[P.Pnumber, P.Dnum]
[E.Lname, E.Address, E.Bdate]
P.Dnum=D.Dnumber
D.Mgr_ssn=E.Ssn
P
D
E
P.Plocation='Stafford'
'Stafford'

Learning Objectives

Lecture No:7

- To study intermediate query form Query Tree in the query processing.

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.

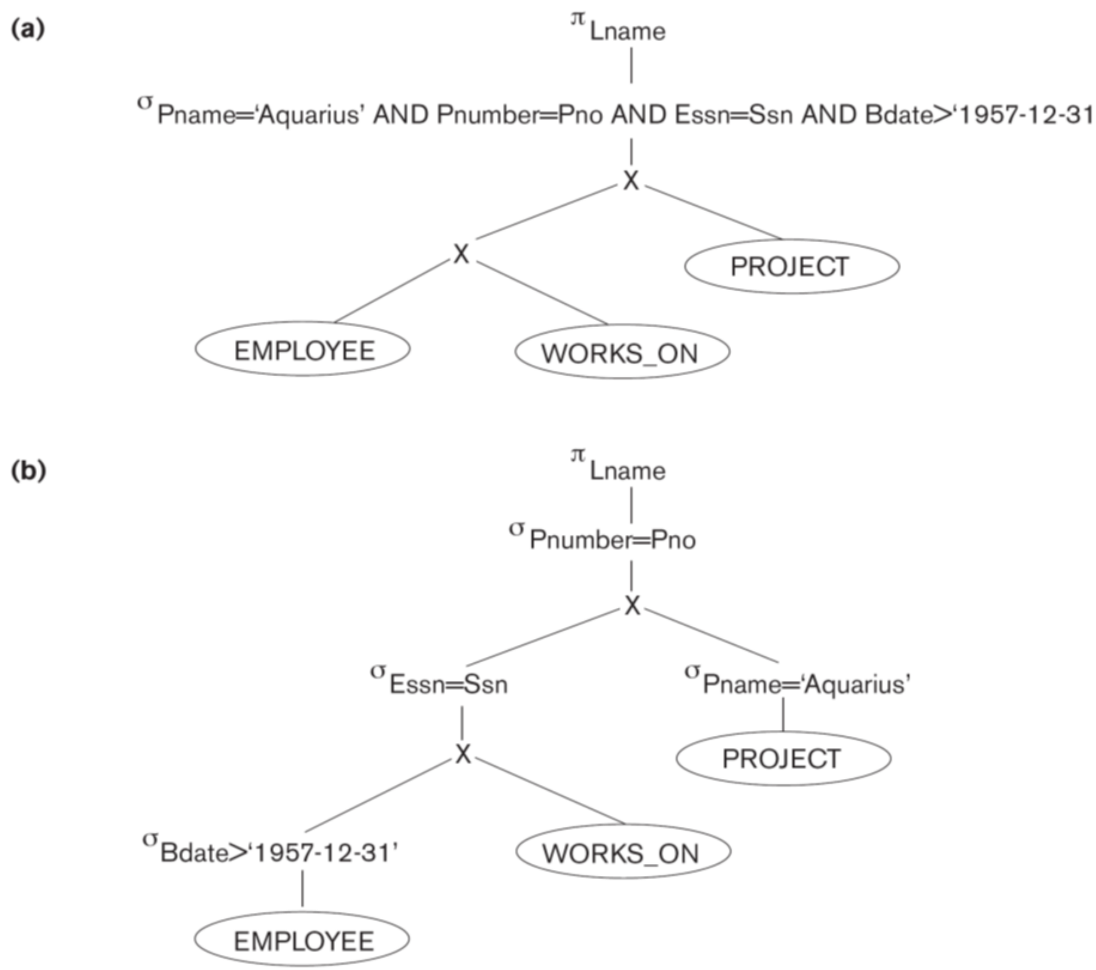
Example: **Find the last names of employees born after 1957, who work on a project named “Aquarius”.**

Using Heuristics in Query Optimization

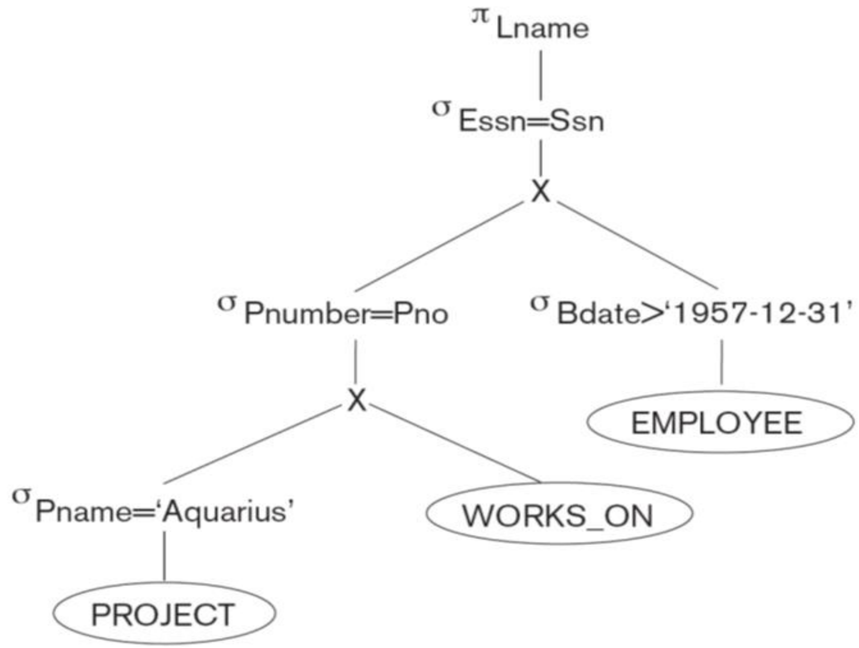
- Example:

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

Figure 19.5
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.

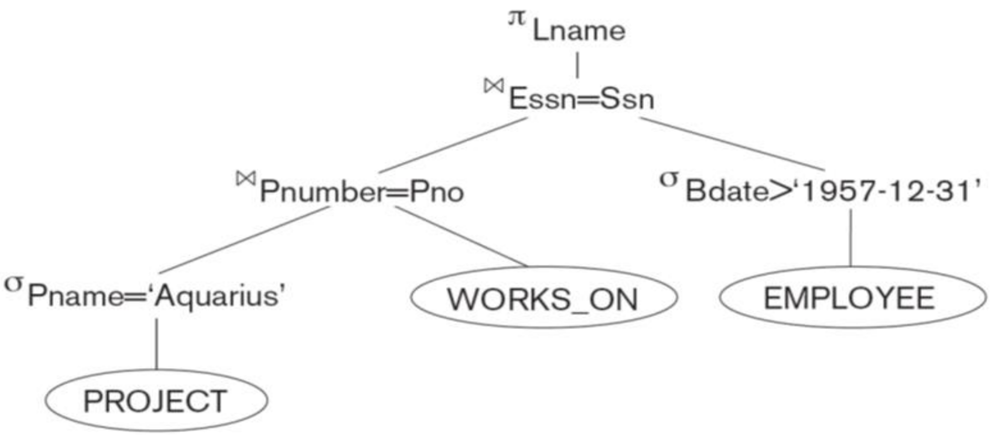


(c)



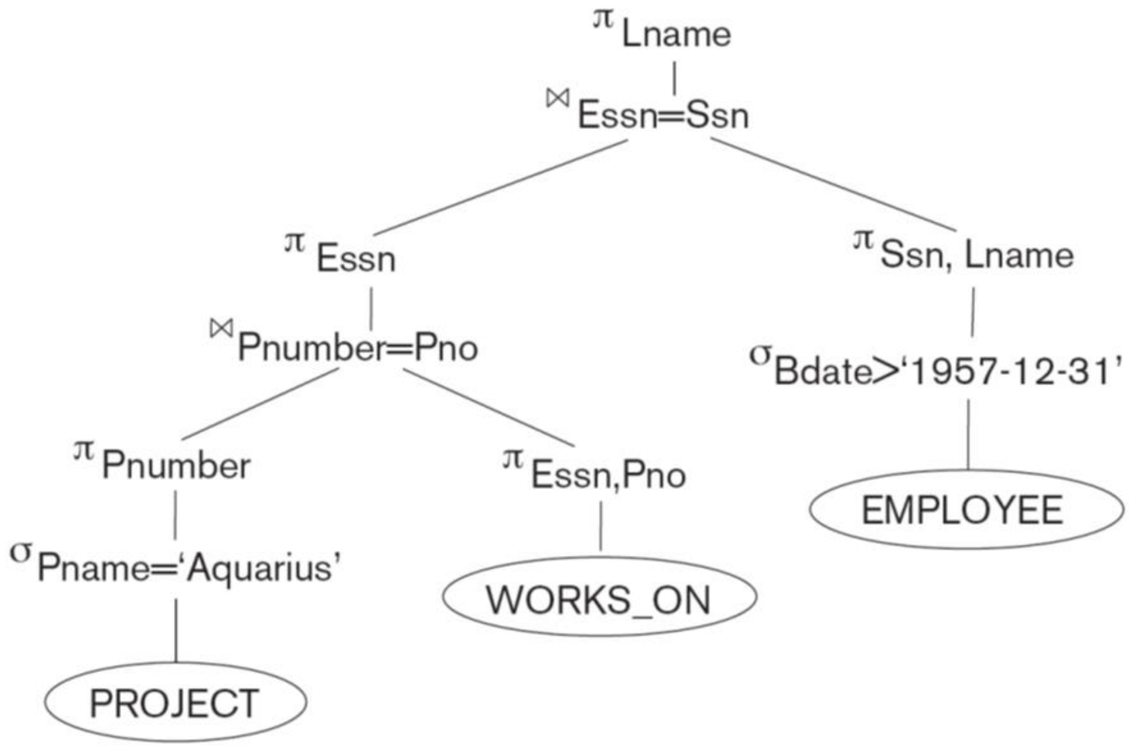
π Lname
σ Essn=Ssn
X
σ Pnumber=Pno
σ Bdate>'1957-12-31'
X
EMPLOYEE
σ Pname='Aquarius'
WORKS_ON
PROJECT

(d)



π Lname
⋈ Essn=Ssn
⋈ Pnumber=Pno
σ Bdate>'1957-12-31'
σ Pname='Aquarius'
WORKS_ON
EMPLOYEE
PROJECT

(e)

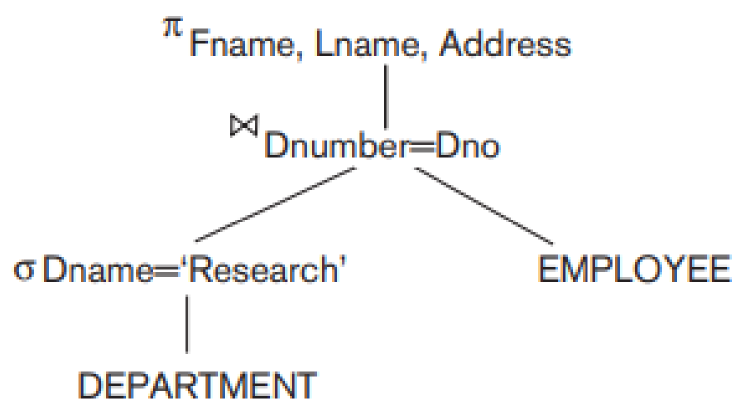

π Lname
⋈ Essn=Ssn
π Essn
π Ssn, Lname
⋈ Pnumber=Pno
σ Bdate>'1957-12-31'
π Pnumber
π Essn,Pno
EMPLOYEE
σ Pname='Aquarius'
WORKS_ON
PROJECT

Examples of Query Tree

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

```
SELECT   Fname, Lname, Address
FROM     EMPLOYEE, DEPARTMENT
WHERE    Dname='Research' AND Dnumber=Dno;
```

$$\pi_{\text{Fname, Lname, Address}}(\sigma_{\text{Dname='Research'}}(\text{DEPARTMENT}) \bowtie_{\text{Dnumber=Dno}} \text{EMPLOYEE})$$



Learning Objectives

Lecture No:8

- To study heuristics based query optimization.

Using Heuristics in Query Optimization

- **General Transformation Rules for Relational Algebra Operations:**

1. **Cascade of σ**: A conjunctive selection condition can be broken up into a cascade (sequence) of individual σ operations:
   - $\sigma_{c1 \text{ AND } c2 \text{ AND } \ldots \text{ AND } cn}(R) = \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$

2. **Commutativity of σ**: The σ operation is commutative:
   - $\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R))$

3. **Cascade of π**: In a cascade (sequence) of π operations, all but the last one can be ignored:
   - $\pi_{List1}(\pi_{List2}(\ldots(\pi_{Listn}(R))\ldots)) = \pi_{List1}(R)$

4. **Commuting σ with π:** If the selection condition c involves only the attributes A1, ..., An in the projection list, the two operations can be commuted:
   - $\pi_{A1, A2, \ldots, An}(\sigma_c(R)) = \sigma_c(\pi_{A1, A2, \ldots, An}(R))$

Using Heuristics in Query Optimization

- General Transformation Rules for Relational Algebra Operations (contd.):

5. **Commutativity of $\bowtie$ (and x):** The $\bowtie$ operation is commutative as is the x operation:
   - $R \bowtie_C S = S \bowtie_C R$; $R \times S = S \times R$

6. **Commuting σ with $\bowtie$ (or x):** If all the attributes in the selection condition c involve only the attributes of one of the relations being joined—say, R—the two operations can be commuted as follows:
   - $\sigma_c (R \bowtie S) = (\sigma_c (R)) \bowtie S$

- Alternatively, if the selection condition c can be written as (c1 and c2), where condition c1 involves only the attributes of R and condition c2 involves only the attributes of S, the operations commute as follows:
   - $\sigma_c (R \bowtie S) = (\sigma_{c1} (R)) \bowtie (\sigma_{c2} (S))$

SIES Graduate School of Technology
RISE WITH EDUCATION

Using Heuristics in Query Optimization

- General Transformation Rules for Relational Algebra Operations (contd.):

5. **Commutativity of $\bowtie$ (and x):** The $\bowtie$ operation is commutative as is the x operation:
   - $R \bowtie_C S = S \bowtie_C R$; $R \times S = S \times R$

6. **Commuting σ with $\bowtie$ (or x):** If all the attributes in the selection condition c involve only the attributes of one of the relations being joined—say, R—the two operations can be commuted as follows:
   - $\sigma_c (R \bowtie S) = (\sigma_c (R)) \bowtie S$

- Alternatively, if the selection condition c can be written as (c1 and c2), where condition c1 involves only the attributes of R and condition c2 involves only the attributes of S, the operations commute as follows:
   - $\sigma_c (R \bowtie S) = (\sigma_{c1} (R)) \bowtie (\sigma_{c2} (S))$

Using Heuristics in Query Optimization

- General Transformation Rules for Relational Algebra Operations (contd.):

7. **Commuting π with $\bowtie$ (or x):** Suppose that the projection list is L = {A1, ..., An, B1, ..., Bm}, where A1, ..., An are attributes of R and B1, ..., Bm are attributes of S. If the join condition c involves only attributes in L, the two operations can be commuted as follows:

  - $\pi_L (R \bowtie_C S) = (\pi_{A1, ..., An} (R)) \bowtie_C (\pi_{B1, ..., Bm} (S))$

- If the join condition C contains additional attributes not in L, these must be added to the projection list, and a final π operation is needed.

Using Heuristics in Query Optimization

- General Transformation Rules for Relational Algebra Operations (contd.):

8. **Commutativity of set operations:** The set operations $\cup$ and $\cap$ are commutative but "–" is not.

9. **Associativity of $\bowtie$, x, $\cup$, and $\cap$:** These four operations are individually associative; that is, if θ stands for any one of these four operations (throughout the expression), we have
   - $(R \theta S) \theta T = R \theta (S \theta T)$

10. **Commuting σ with set operations:** The σ operation commutes with $\cup$, $\cap$, and –. If θ stands for any one of these three operations, we have
   - $\sigma_c (R \theta S) = (\sigma_c (R)) \theta (\sigma_c (S))$

Converting Query Trees into Query Execution Plan

- Query Execution Plans
  - An execution plan for a relational algebra query consists of a combination of the relational algebra query tree and information about the access methods to be used for each relation as well as the methods to be used in computing the relational operators stored in the tree.
  - **Materialized evaluation**: the result of an operation is stored as a temporary relation.
  - **Pipelined evaluation**: as the result of an operator is produced, it is forwarded to the next operator in sequence.

Examples to Solve

Query 8. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

```
Q8:   SELECT   E.Fname, E.Lname, S.Fname, S.Lname
      FROM     EMPLOYEE AS E, EMPLOYEE AS S
      WHERE    E.Super_ssn=S.Ssn;
```

Query 27. For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.

```
Q27:  SELECT    Pnumber, Pname, COUNT (*)
      FROM      PROJECT, WORKS_ON, EMPLOYEE
      WHERE     Pnumber=Pno AND Ssn=Essn AND Dno=5
      GROUP BY  Pnumber, Pname;
```

Cost-Based Query Optimization

Lecture-9

Learning Objectives:

- To understand different cost components considered in cost based query optimization.
- To learn number of execution strategies to be considered in the query optimization.
- To study Cost Functions for different operations.
- To learn Cost Based Query Optimization in detail.

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.

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

Cost Components for Query Execution

1. Access cost to secondary storage
2. Storage cost
3. Computation cost
4. Memory usage cost
5. Communication cost

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)

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:
  - $C_{S2} = \log_2 b + (s/bfr)\rceil - 1$
  - 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;

Examples of Cost Functions for JOIN

- Join selectivity (js)
- $js = |(R \bowtie_C S)| / |R \times S| = |(R \bowtie_C S)| / (|R| * |S|)$
  - If condition C does not exist, js = 1;
  - If no tuples from the relations satisfy condition C, js = 0;
  - Usually, 0 <= js <= 1;

- Size of the result file after join operation
  - $|(R \bowtie_C S)| = js * |R| * |S|$

Examples

Query Optimization

Given the following SQL query:

```
Student (sid, name, age, address)
Book(bid, title, author)
Checkout(sid, bid, date)

SELECT S.name
FROM Student S, Book B, Checkout C
WHERE S.sid = C.sid
AND B.bid = C.bid
AND B.author = 'Olden Fames'
AND S.age > 12
AND S.age < 20
```

Thank You!

(seemasr@sies.edu.in)