CSE2004 – Database Management Systems

Text Books :

1.R. Elmasri & S. B. Navathe, **Fundamentals of Database Systems**, Addison Wesley, 7 th Edition, 2015

2.Raghu Ramakrishnan, **Database Management Systems**,Mcgraw-Hill,4th edition,2015

Reference Book

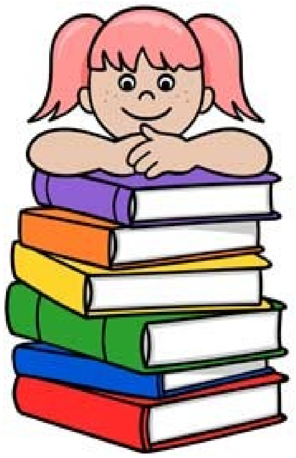
3.A. Silberschatz, H. F. Korth & S. Sudershan, **Database System Concepts**, McGraw Hill, 6th Edition 2010

4.Thomas Connolly, Carolyn Begg,” **Database Systems : A Practical Approach to Design, Implementation and Management**”,6th Edition,2012

5.Shashank Tiwari ,“Professional NoSql”,Wiley ,2011

Unit – 4 : QUERY PROCESSING AND TRANSACTION PROCESSING

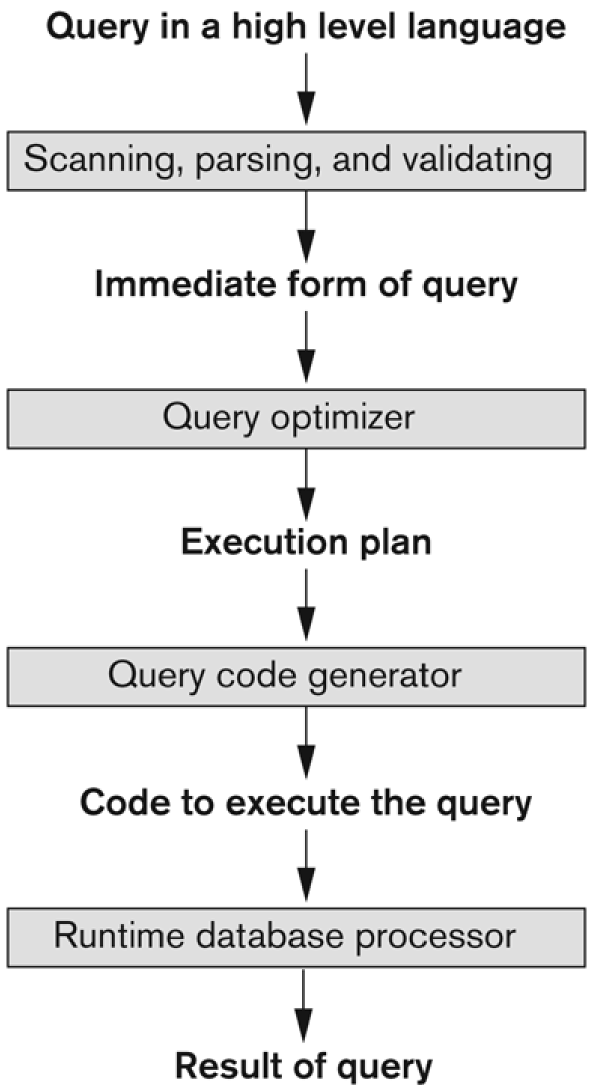
- ✓ Translating SQL Queries into Relational Algebra
- ✓ Heuristic query optimization
- ✓ Introduction to Transaction Processing
- ✓ Transaction and System concepts
- ✓ Desirable properties of Transactions
- ✓ Characterizing schedules based on recoverability
- ✓ Characterizing schedules based on serializability



Introduction to Query Processing

- A query expressed in a HLL(SQL) must first be scanned, parsed, and validated.
- The Scanner identifies the query tokens(SQL keywords, attribute names, and relation names).
- The Parser checks the query syntax to determine whether it is formulated according to the syntax rules (rules of grammar) of the query language.
- The Validated by checking that all attribute and relation names are valid and semantically meaningful names in the schema.
- An internal representation of the query is then created(Query Tree/Graph).
- The DBMS must then devise an execution strategy/query plan for retrieving the results of the query from the database.
- A query typically has many possible execution strategies, and the process of choosing a suitable one for processing a query is known as query optimization.

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)

Translating SQL Queries into Relational Algebra

- An SQL query is first translated into an equivalent extended Relational algebra expression, represented as a query Tree/Graph data structure and then optimized.
- SQL queries are decomposed into query blocks(Units/Chunks).
- Query Block : It is 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
```

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

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

Finally, The query optimizer will choose an execution plan for each query block.

Translating SQL Queries into Relational Algebra

For Converting/Translating a Query, written in HLL(SQL) into Relational Algebra, need to have an appropriate strategies for the following:

- Algorithm for Selection Operation
- Algorithm for Projection and Set Operation
- Algorithm for External Sorting
- Implementation of JOIN, SET and Aggregate Operations
- Combining Operations Using Pipeling
- Parallel Algorithms for Query Processing

Query Trees and Heuristics for Query Optimization

Heuristic : Problem solving by Experimental (Trail-and-Error)
Heuristic Rules : Used to Modify the Internal Representation of Query, to improve the performance. i.e Query Tree/ Query Graph (Data Structure)

Process for heuristics optimization

- The parser of a high-level query generates an initial internal representation;
- Apply heuristics rules to optimize the internal representation.
- 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.

Query Trees and Heuristics for 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.

Query Trees and Heuristics for Query Optimization

For Eg :

– For every project located in 'Stafford', retrieve the project number, department number, and department manager's last name, address, and birth-date.

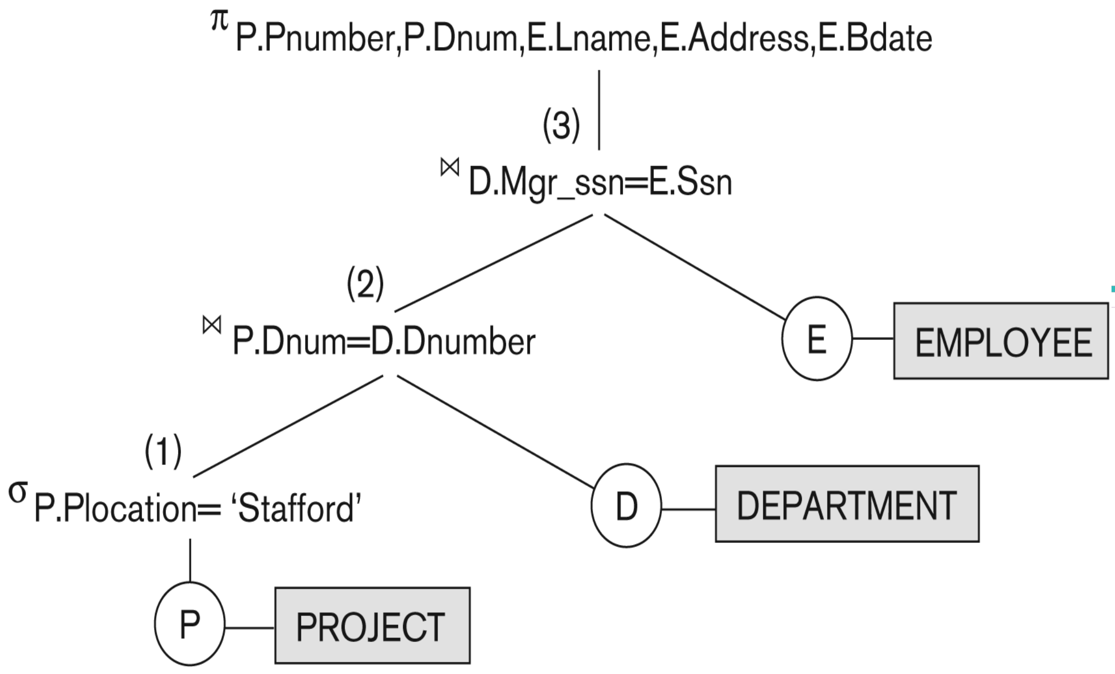
Relation Algebra :

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

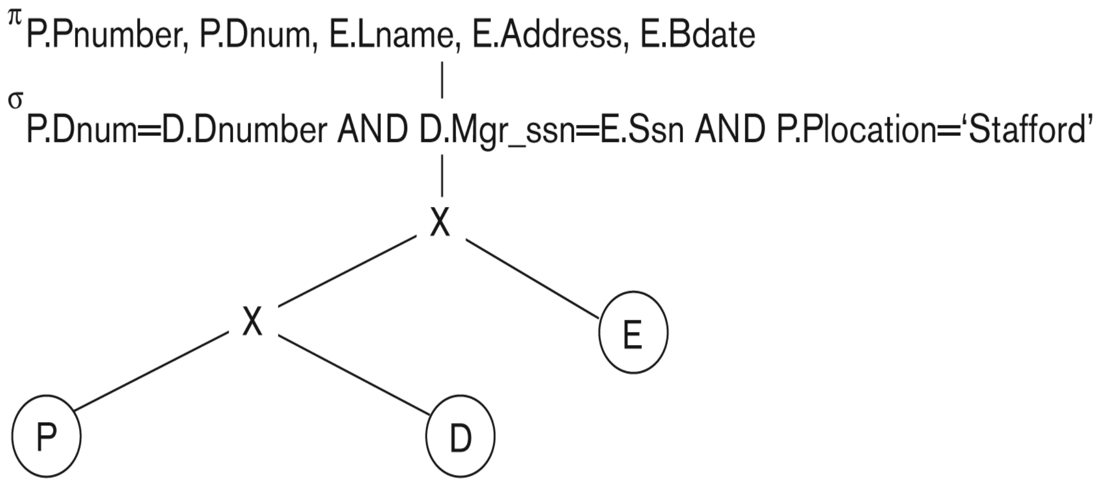
SQL Query :

```
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';
```

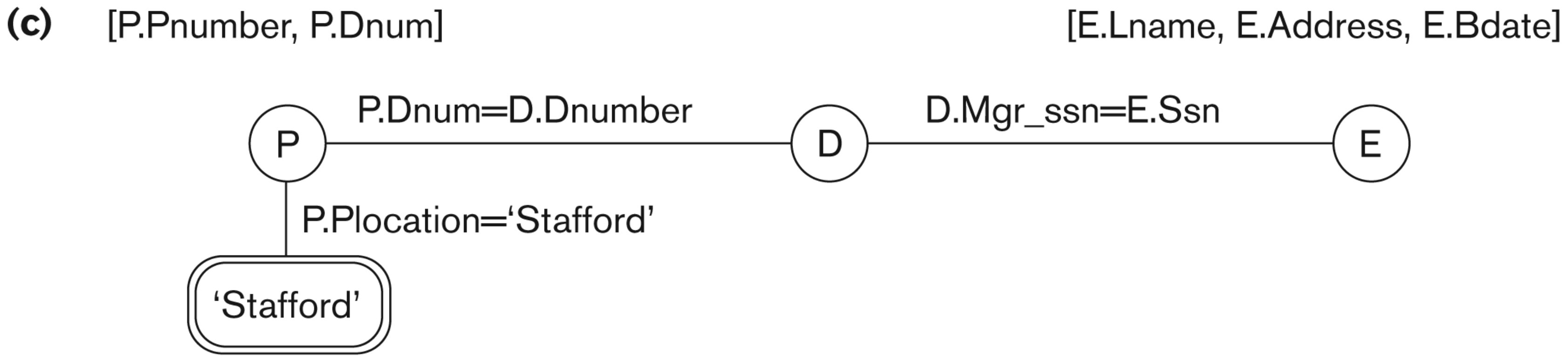
(a)


π P.Pnumber,P.Dnum,E.Lname,E.Address,E.Bdate
(3)
$\bowtie$ D.Mgr_ssn=E.Ssn
(2)
$\bowtie$ 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

Query Trees and Heuristics for Query Optimization



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.

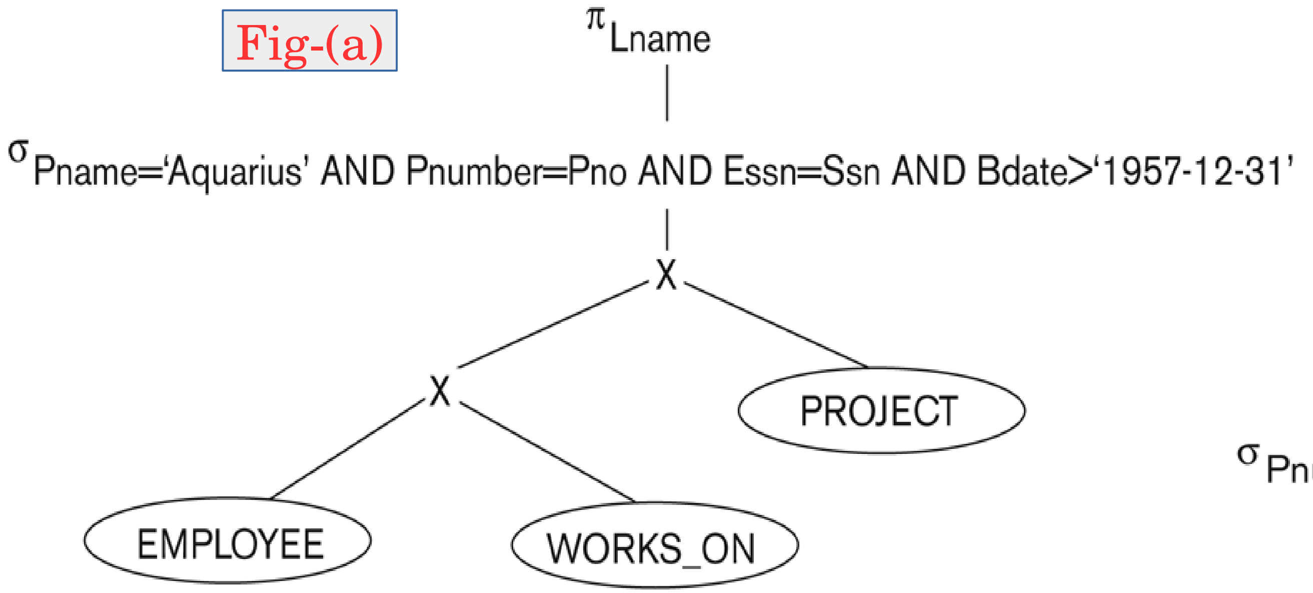
The task of heuristic optimization of query trees is to find a **final query tree** that is efficient to execute.

Example:

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

Using Heuristics in Query Optimization

Fig-(a)


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

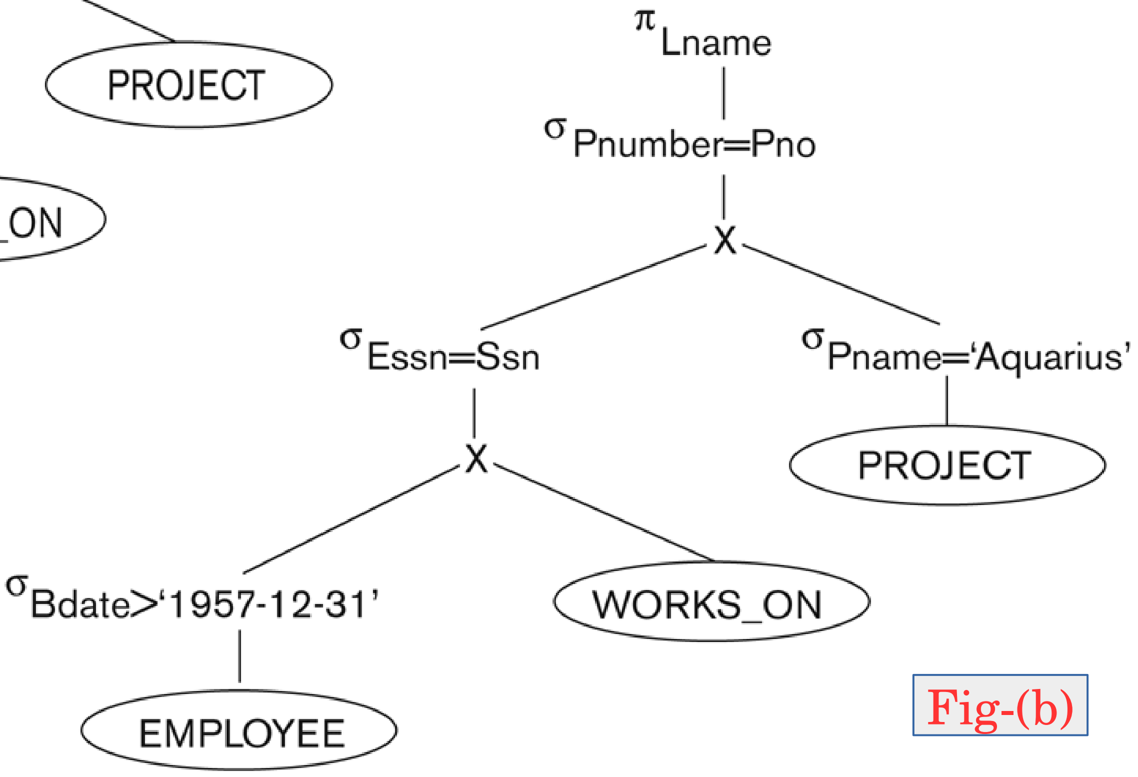
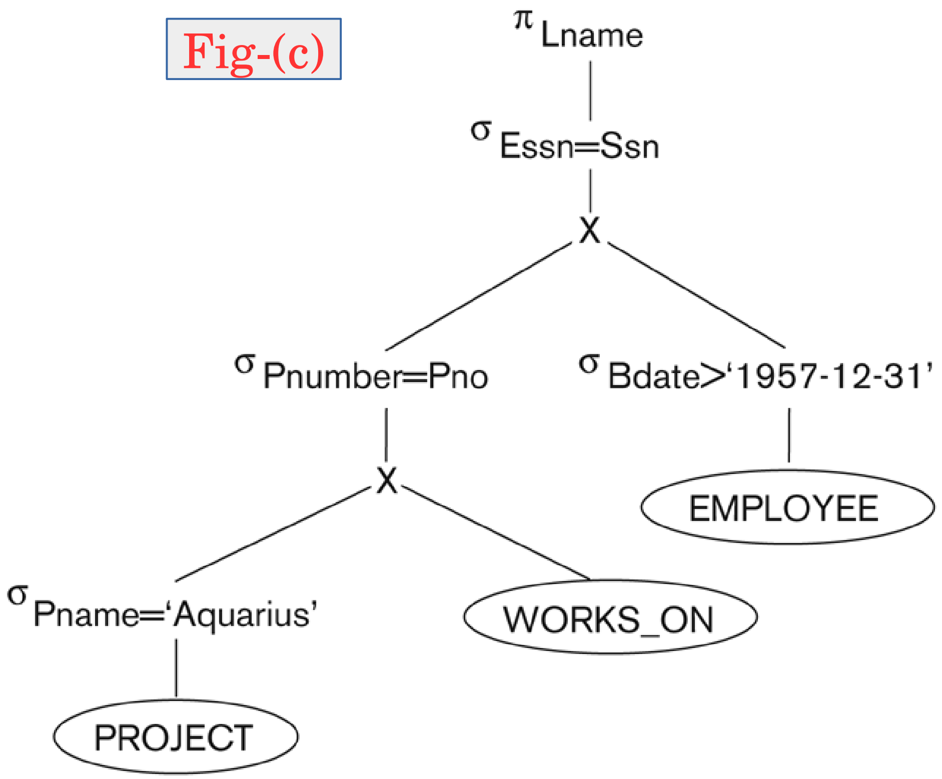


Fig-(b)

Using Heuristics in Query Optimization

Fig-(c)



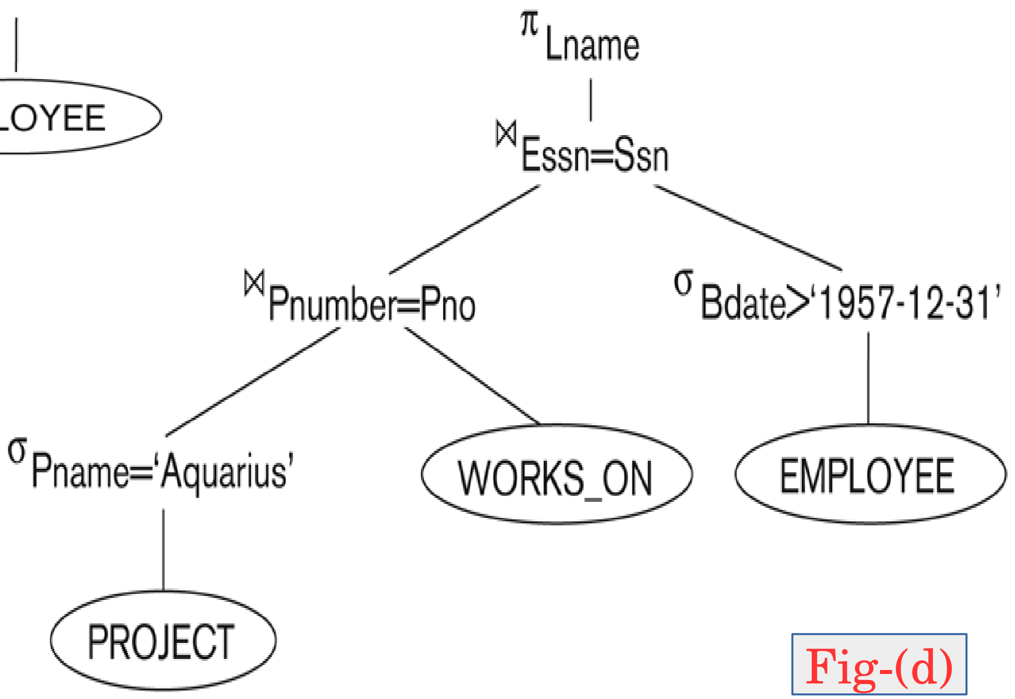
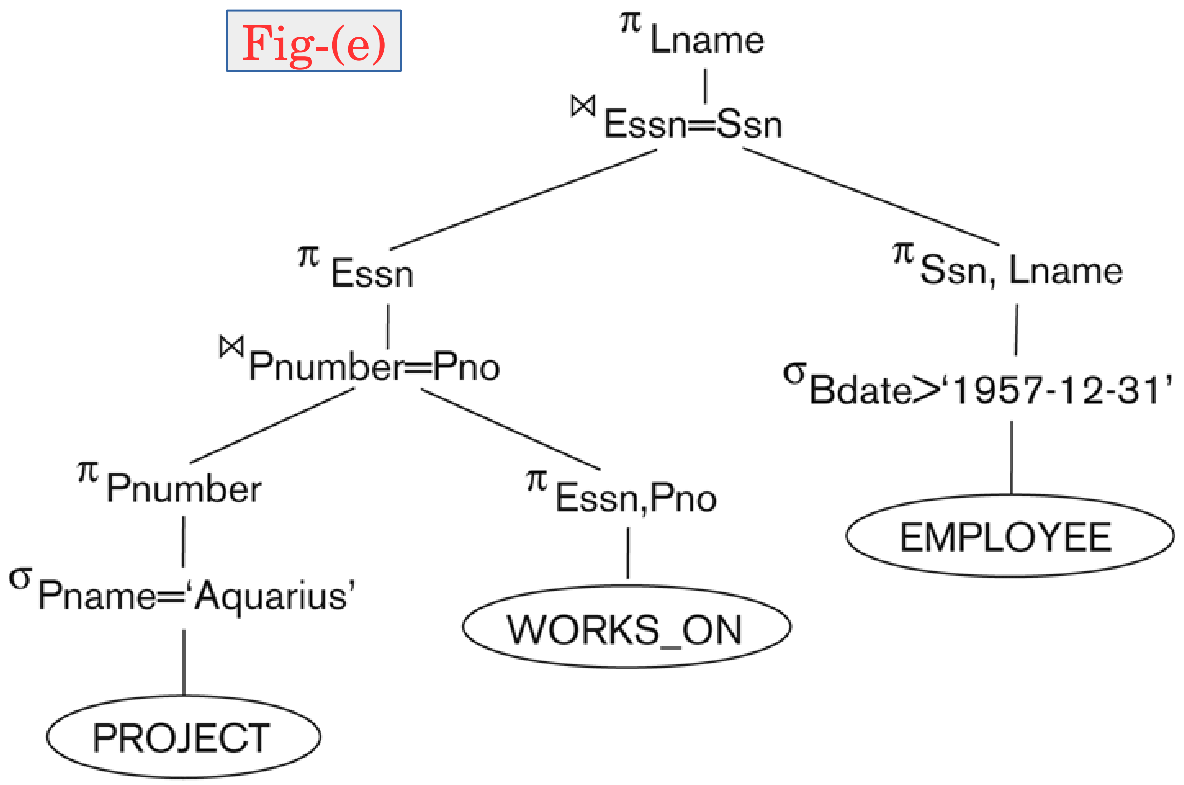


Fig-(d)

VIT® UNIVERSITY (Estd. u/s 3 of UGC Act 1956) www.vit.ac.in Vellore ▪ Chennai CHENNAI CAMPUS Vandalur - Kelambakkam Road, Chennai - 600127

Using Heuristics in Query Optimization

Fig-(e)



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.

Transformation Rules for Relational Algebra Operations

1. Cascade of σ : A conjunctive selection condition can be broken up into a cascade (that is, a sequence) of individual σ operations:

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

2. Commutativity of σ. The σ operation is commutative:

$$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$$

3. Cascade of π. In a cascade (sequence) of π operations, all but the last one can be ignored:

$$\pi_{List1}(\pi_{List2}(\ldots(\pi_{List\ n}(R))\ldots)) \equiv \pi_{List1}(R)$$

4. Commuting σ with π : If the selection condition c involves only those attributes A1 , … , An in the projection list, the two operations can be commuted:

$$\pi_{A1, A2, \ldots, An}(\sigma_c(R)) \equiv \sigma_c(\pi_{A1, A2, \ldots, An}(R))$$

Transformation Rules for Relational Algebra Operations

5. Commutativity of $\bowtie$ (and X). The join operation is commutative, as is the $\times$ operation:

$$R \bowtie_c S \equiv S \bowtie_c R$$

$$R \times S \equiv S \times R$$

Notice that although the order of attributes may not be the same in the relations resulting from the two joins (or two Cartesian products), the meaning is the same because the order of attributes is not important in the alternative definition of relation.

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: $\sigma_c (R \bowtie S) = (\sigma_{c1} (R)) \bowtie (\sigma_{c2} (S))$.

Transformation Rules for Relational Algebra Operations

7. Commuting π with ⋈ (or x): Suppose that the projection list is L = {$A_1, ..., A_n, B_1, ..., B_m$}, where $A_1, ..., A_n$ are attributes of R and $B_1, ..., B_m$ 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.

8. Commutativity of set operations: The set operations ∪ and ∩ are commutative but "–" is not.

9. Associativity of ⋈, x, ∪, and ∩ : These four operations are individually associative; that is, if θ stands for any one of these four operations (throughout the expression), we have

$$(R \theta S) \theta T = R \theta (S \theta T)$$

Transformation Rules for Relational Algebra Operations

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

11. The π operation commutes with **:** $\pi_L (R \cup S) = (\pi_L (R)) \cup (\pi_L (S))$

12. Converting a (σ, x) sequence into ⋈ : If the condition c of a σ that follows a x Corresponds to a join condition, convert the (σ, x) sequence into a as follows:

$$(\sigma_C (R \times S)) = (R \bowtie_C S)$$

13. Pushing σ in conjunction with set difference. $\sigma_c(R - S) = \sigma_c(R) - \sigma_c(S)$

However, σ may be applied to only one relation: $\sigma_c(R - S) = \sigma_c(R) - S$

14. Pushing **σ** to only one argument in ∩.

If in the condition σ c all attributes are from relation R, then: $\sigma_c(R \cap S) = \sigma_c(R) \cap S$

15. If S is empty, then **R ∪ S = R**

If the condition c in σ_c is true for the entire R, then $\sigma_c(R) = R$.

Using Heuristics in Query Optimization

Outline of a Heuristic Algebraic Optimization Algorithm:

- Using rule 1, break up any select operations with conjunctive conditions into a cascade of select operations.
- Using rules 2, 4, 6, and 10 concerning the commutativity of select with other operations, move each select operation as far down the query tree as is permitted by the attributes involved in the select condition.
- Using rule 9 concerning associativity of binary operations, rearrange the leaf nodes of the tree so that the leaf node relations with the most restrictive select operations are executed first in the query tree representation.
- Using Rule 12, combine a Cartesian product operation with a subsequent select operation in the tree into a join operation.
- Using rules 3, 4, 7, and 11 concerning the cascading of project and the commuting of project with other operations, break down and move lists of projection attributes down the tree as far as possible by creating new project operations as needed.
- Identify subtrees that represent groups of operations that can be executed by a single algorithm.

Using Heuristics in Query Optimization

Summary of Heuristics for Algebraic Optimization:

- The main heuristic is to apply first the operations that reduce the size of intermediate results.

- 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.)

- 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 Heuristics in Query Optimization

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.
- **Pipe-lined evaluation**: as the result of an operator is produced, it is forwarded to the next operator in sequence.

Cost Components for Query Execution :

1. Access cost to Secondary Storage
2. Disk Storage Cost
3. Computation Cost
4. Memory Usage Cost
5. Communication Cost