Database Management System

Query Processing and Optimization

Dr. Balasundaram A

VIT-Chennai

CSE2004 Syllabus

Module-4

- Translating SQL Queries into Relational Algebra
- Heuristic query optimization
- Introduction to Transaction Processing

Text Books and References

Text Books

- R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7 th Edition, 2015
- Raghu Ramakrishnan,Database Management Systems,Mcgraw-Hill,4 th edition,2015

References

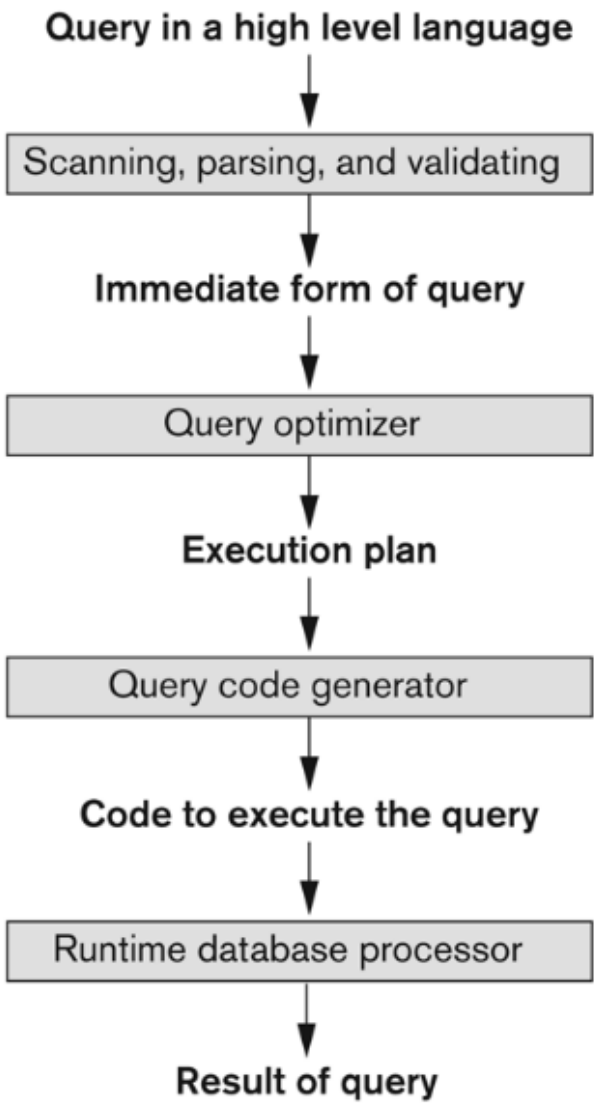
- A. Silberschatz, H. F. Korth & S. Sudershan, Database System Concepts, McGraw Hill, 6 th Edition 2010
- Thomas Connolly, Carolyn Begg, Database Systems : A Practical Approach to Design, Implementation and Management,6 th Edition,2012
- Pramod J. Sadalage and Marin Fowler, NoSQL Distilled: A brief guide to merging world of Polyglot persistence, Addison Wesley, 2012.
- Shashank Tiwari, Professional NoSql, Wiley, 2011.

VIT Vellore Institute of Technology

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 Validation involves ensuring and checking whether 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



Code can be:

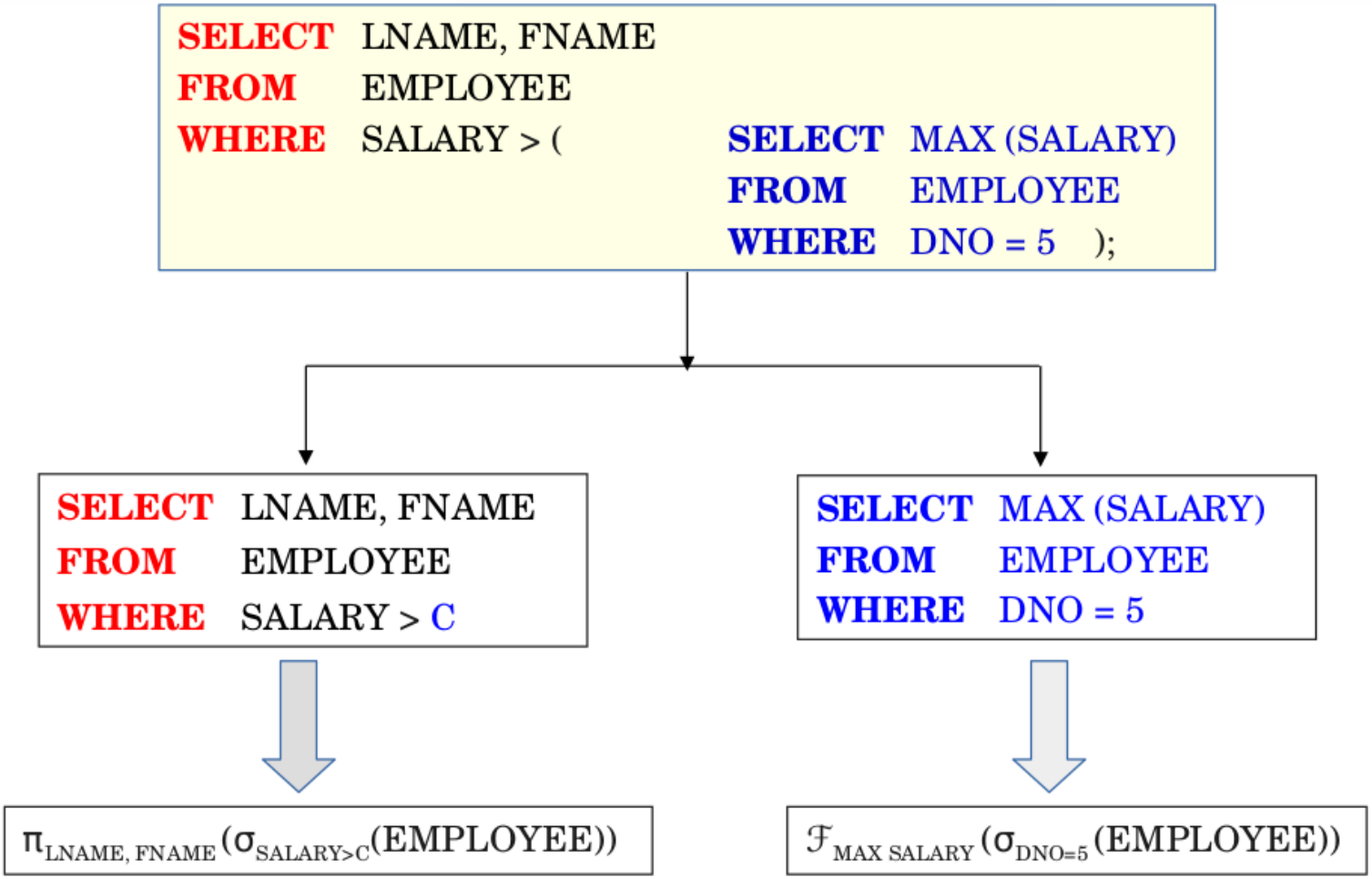
Executed directly (interpreted mode)

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 : are within a query are identified as separate query blocks.
- Aggregate operators : are in SQL must be included in the extended algebra

Translating SQL Queries into Relational Algebra



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
  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.

Query Trees and Heuristics for Query Optimization

Query tree: is a data structure that corresponds to a Reln. Algebra expression, which 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.

Example-1

For Eg :

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

SQL Query :

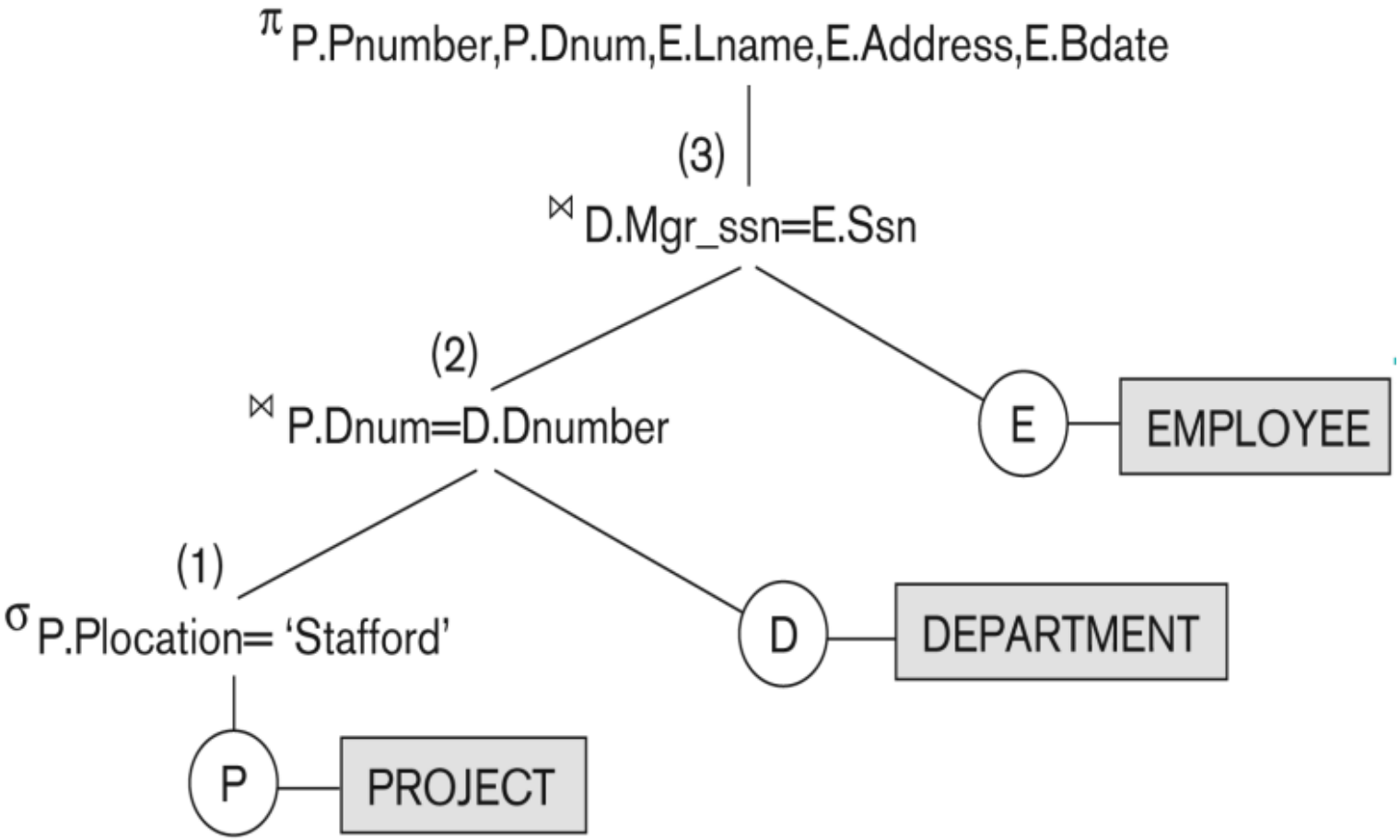
- SELECT P.NUMBER,P.DNUM,E.LNAME, E.ADDRESS, E.BDATE
  FROM PROJECT AS P, DEPARTMENT AS D, EMPLOYEE AS E
  WHERE P.DNUM=D.DNUMBER AND D.MGRSSN=E.SSN AND
  P.PLOCATION=‘STAFFORD’;

Relation Algebra :

- $\pi_{Pnumber,Dnum,Lname,Address,Bdate}(((\sigma_{Plocation='Stafford'}(\textbf{PROJECT})) \bowtie_{Dnum=Dnumber}(\textbf{DEPARTMET})) \bowtie_{Mgr_ssn=Ssn}(\textbf{EMPLOYEE}))$

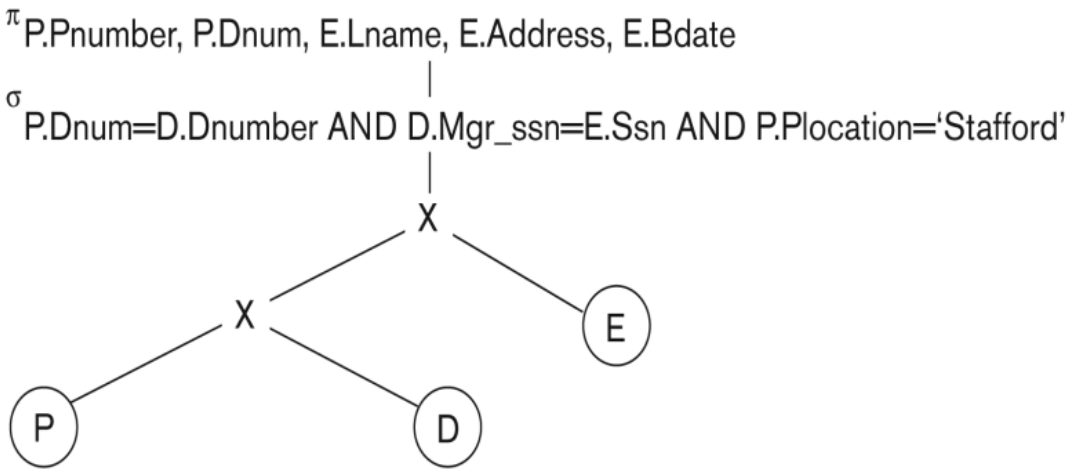
Example-1

Query Tree -1

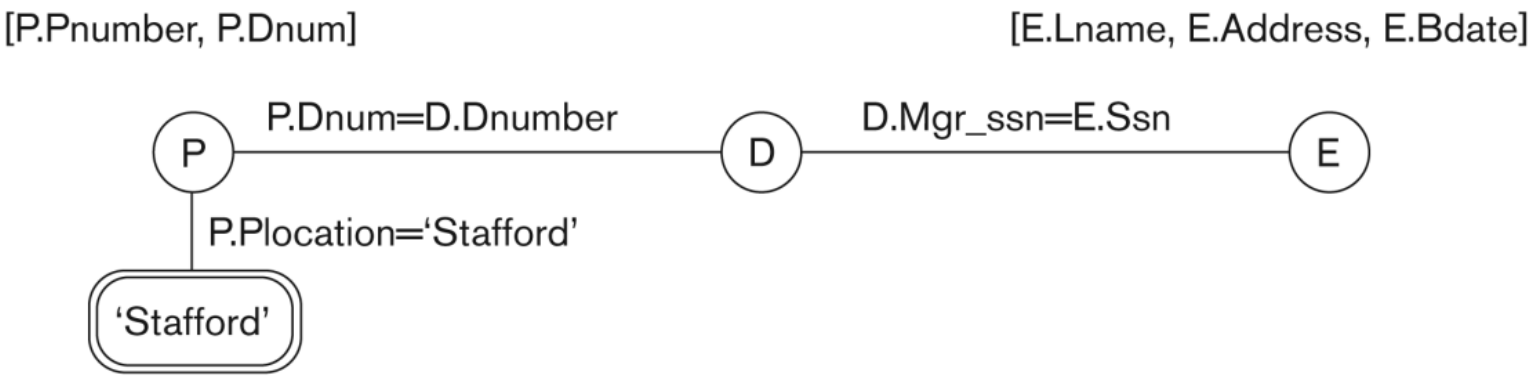


Example-1

Query Tree -2

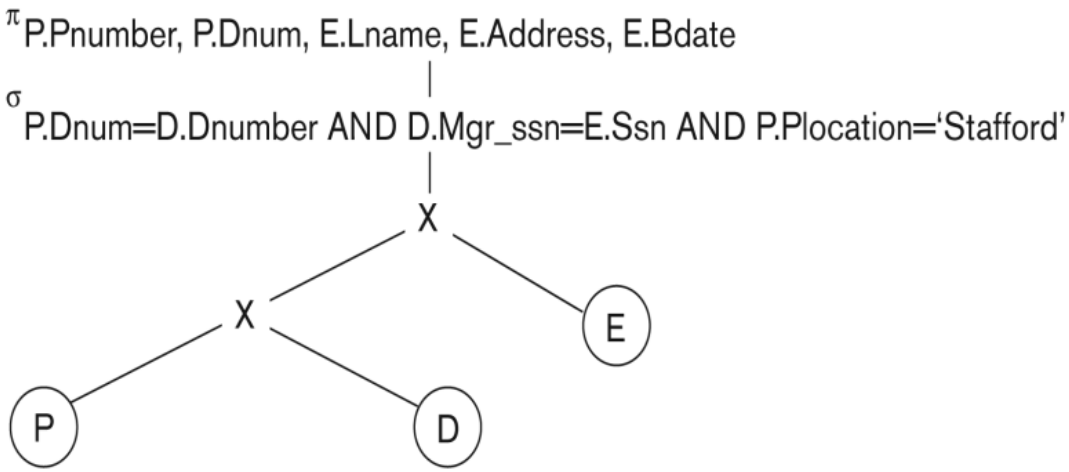


Query Tree -3

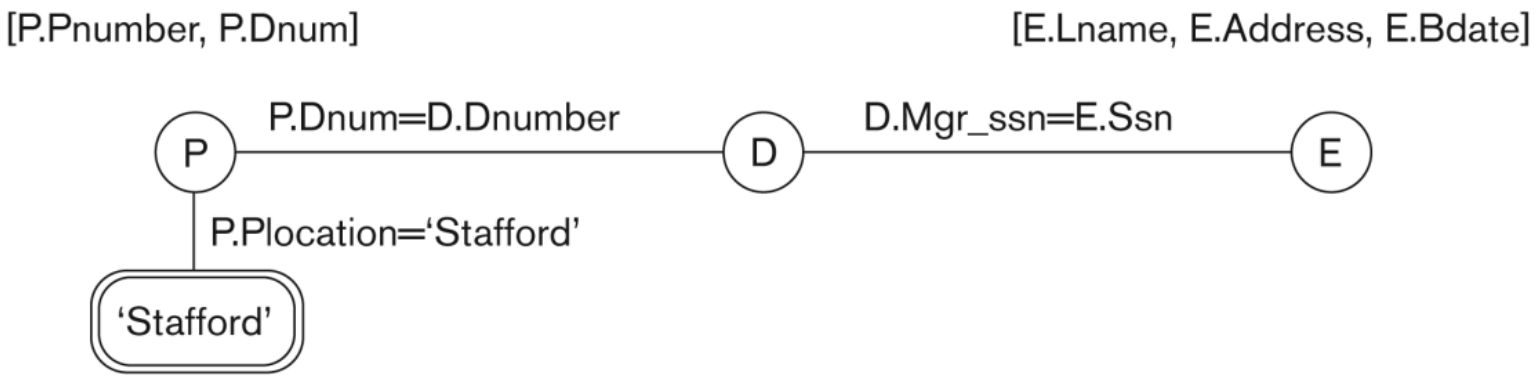


Example-1

Query Tree -2



Query Tree -3



Using Heuristics in Query Optimization

Heuristic Query Optimization:

- Oracle calls this Rule Based optimization.
- A query can be represented as a tree data structure. Operations are at the interior nodes and data items (tables, columns) are at the leaves.
- The query is evaluated in a depth-first pattern.

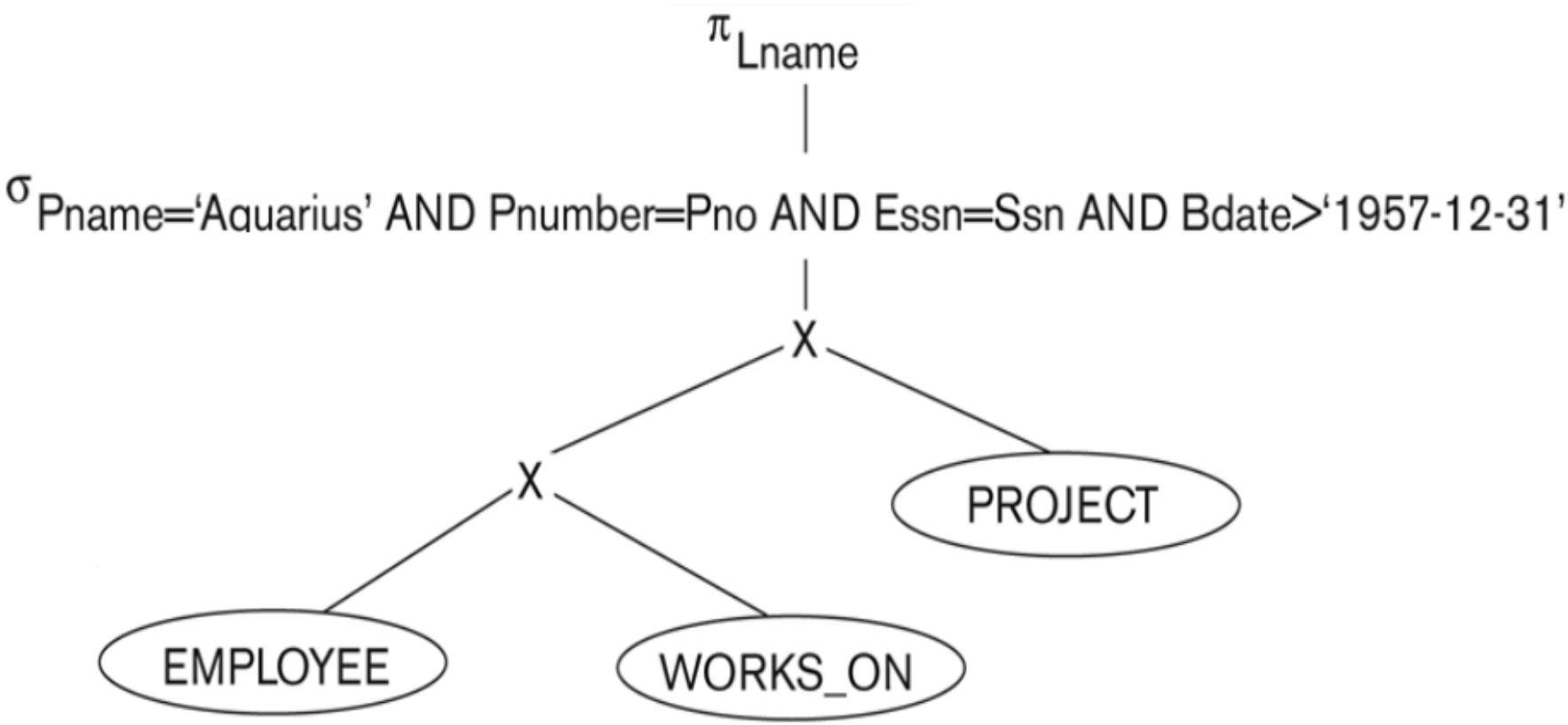
Heuristic Optimization of Query Trees: The same query could correspond to many different relational algebra expressions and hence many different query trees. The task of heuristic optimization of query trees is to find a final query tree that is efficient to execute.

Example:

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

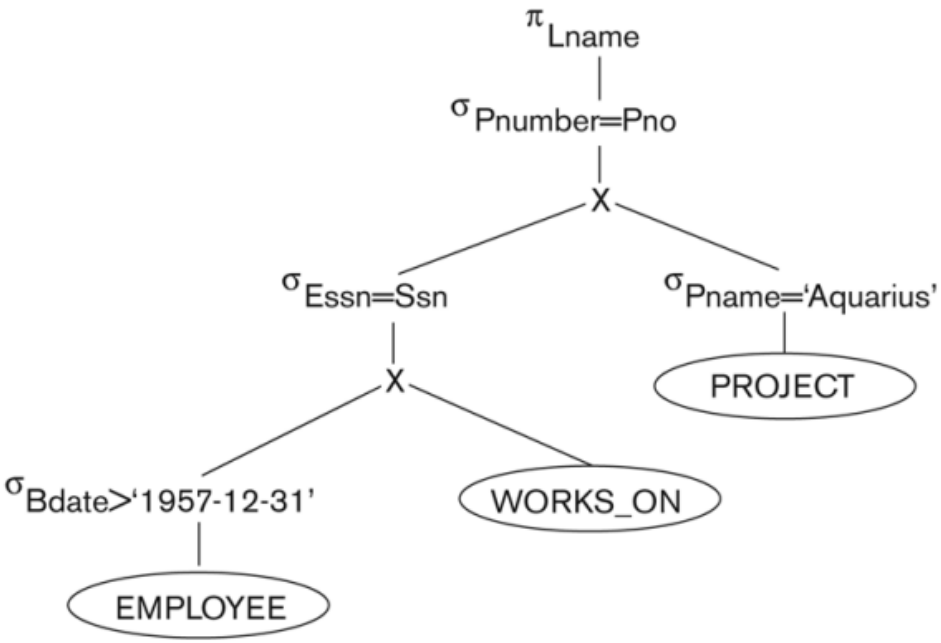
Using Heuristics in Query Optimization

Query Tree-1

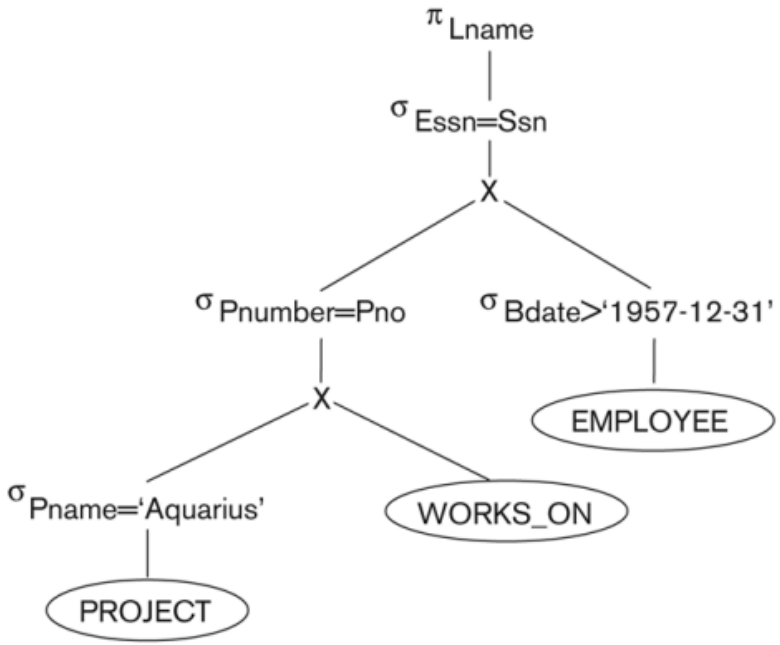


Using Heuristics in Query Optimization

Query Tree-2

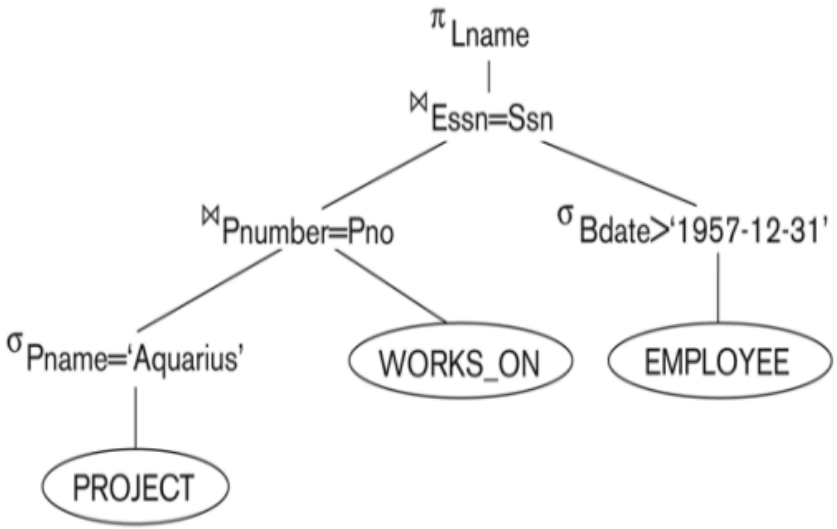


Query Tree-3

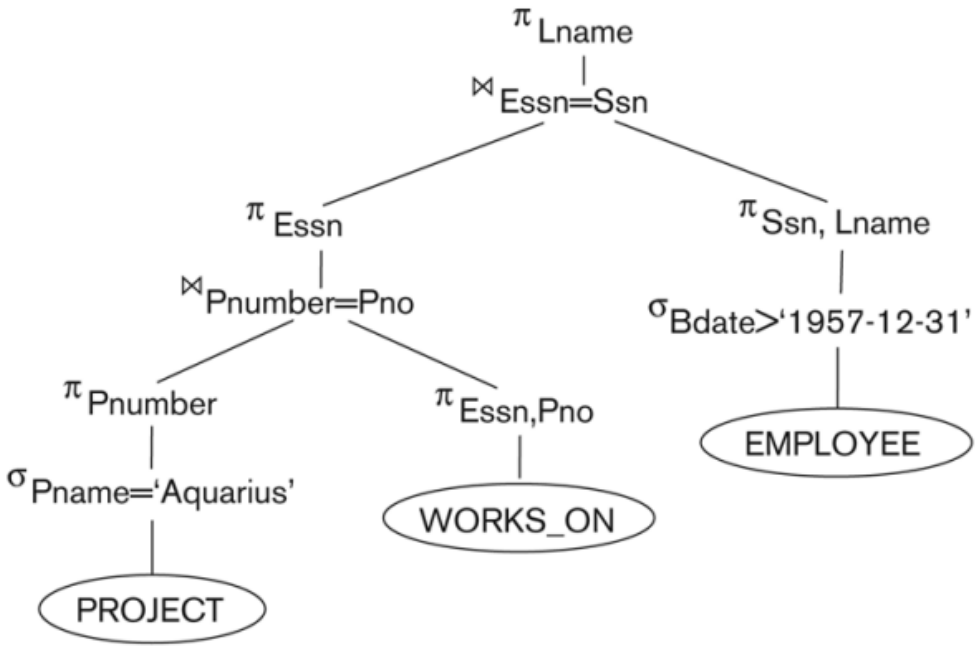


Using Heuristics in Query Optimization

Query Tree-4



Query Tree-5



Steps to Optimize Query

Query Optimization Steps in converting a Query Tree

- Initial (Canonical) Query Tree for SQL Query 'Q'.
- Moving SELECT Operations down the Query Tree
- Applying the More Restrictive SELECT Operation First
- Replacing the CARTESIAN PRODUCT and SELECT Operations with JOIN Operations
- Moving PROJECTION Operations down the Query Tree 'Q'

Transformation Rules for Relational Algebra

An overall rule for heuristic query optimization is to perform as many select and project operations as possible before doing any joins.

1. Cascade of σ : A conjunctive selection condition can be broken up into a cascade (that is, a sequence) of individual σ operations:
$$\sigma_{c_1,c_2,c_3....,c_n}(R) = \sigma_{c_1}(\sigma_{c_2}(\sigma_{c_3}....., (\sigma_{c_n}(R))))$$
2. Commutative of (σ) : The 'σ' operation is commutative:
$$\sigma_{c_1}(\sigma_{c_2}(R)) = \sigma_{c_2}(\sigma_{c_1}(R))$$
3. Cascade of π : In a cascade (sequence) of π operations, all but the last one can be ignored:
$$\pi_{List_1}\ (\pi_{List_2}\ (...., \pi_{List_n}(R)\ )\ ) = \pi_{List_1}(R)$$
4. Commuting σ with π : If the selection condition 'c' involves only those attributes A_1 , ... , A_n in the projection list, the two operations can be commuted:
$$\pi_{A_1,A_2,A_3...,A_n}(\sigma_c(R)) = \sigma_c(\pi_{A_1,A_2,A_3...,A_n}((R)))$$

Transformation Rules for Relational Algebra

5. Commutativity of $\bowtie$ and X : The Join Operation is Commutative, as is 'X' Operation:
   1. $S \bowtie_c R = R \bowtie_c S$
   2. $R \; X \; S = S \; X \; R$

   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 (X or $\bowtie$): 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 (c_1 and c_2), where condition c_1 involves only the attributes of R and condition c_2 involves only the attributes of S, then the operation commute as follows :

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

Transformation Rules for Relational Algebra

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

8. 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 :
$$(R\ \theta\ S)\ \theta\ T = R\ \theta\ (\ S\ \theta\ T\ )$$

9. Commuting π with ($\bowtie$ 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:
$$\pi_L(R \bowtie_c S\ ) = \pi_{A_1,A_2,A_3,....A_n}(R) \bowtie_c \pi_{B_1,B_2,B_3,....B_n}(S)$$

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

Transformation Rules for Relational Algebra

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

12. Converting (σ, X) into $\bowtie$ = If the condition 'c' of a σ that follows a 'X' corresponds to a join condition, convert the (σ, X) sequence into a $\bowtie$ as follows:
$$\sigma_c\ (R\ X\ S) = R \bowtie_c S$$

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 $\cap$: 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 $R \cup S = R$ 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 :

- Access cost to Secondary Storage
- Disk Storage Cost
- Computation Cost
- Memory Usage Cost
- Communication Cost

Query Optimization

- Query optimization is a difficult part of the query processing.
- It determines the efficient way to execute a query with different possible query plans.
- It cannot be accessed directly by users once the queries are submitted to the database server or parsed by the parser.
- A query is passed to the query optimizer where optimization occurs.
- Main aim of Query Optimization is to minimize the cost function, I/O Cost + CPU Cost + Communication Cost
- It defines how an RDBMS can improve the performance of the query by re-ordering the operations.
- It is the process of selecting the most efficient query evaluation plan from among various strategies if the query is complex.
- It computes the same result as per the given expression, but it is a least costly way of generating result.

Importance of Query Optimization

- Query optimization provides faster query processing.
- It requires less cost per query.
- It gives less stress to the database.
- It provides high performance of the system.
- It consumes less memory.

Thanks