7. HEURISTIC-BASED QUERY OPTIMIZATION

In general, many different relational algebra expressions—and hence many different query trees can be **equivalent**; that is, they can represent the *same query*.

The query parser will typically generate a standard **initial query tree** to correspond to an SQL query, without doing any optimization.

For example, for a SELECT-PROJECT-JOIN query, such as Q2, the initial tree is shown in Figure. The CARTESIAN PRODUCT of the relations specified in the FROM clause is first applied; then the selection and join conditions of the WHERE clause are applied, followed by the projection on the SELECT clause attributes.

Such a canonical query tree represents a relational algebra expression that is *very inefficient if executed directly,* because of the CARTESIAN PRODUCT (×) operations.

The heuristic query optimizer will transform this initial query tree into an equivalent **final query tree** that is efficient to execute.

The optimizer must include rules for *equivalence among relational algebra expressions* that can be applied to transform the initial tree into the final, optimized query tree. First we discuss informally how a query tree is transformed by using heuristics, and then we discuss general transformation rules and show how they can be used in an algebraic heuristic optimizer.

1. Break up SELECT operations with conjunctive conditions into a cascade of SELECT operations
2. Using 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
3. Using commutativity and associativity of binary operations, rearrange the leaf nodes of the tree
4. Combine a CARTESIAN PRODUCT operation with a subsequent SELECT operation in the tree into a JOIN operation, if the condition represents a join condition
5. Using 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

6. Identify sub-trees that represent groups of operations that can be executed by a single algorithm

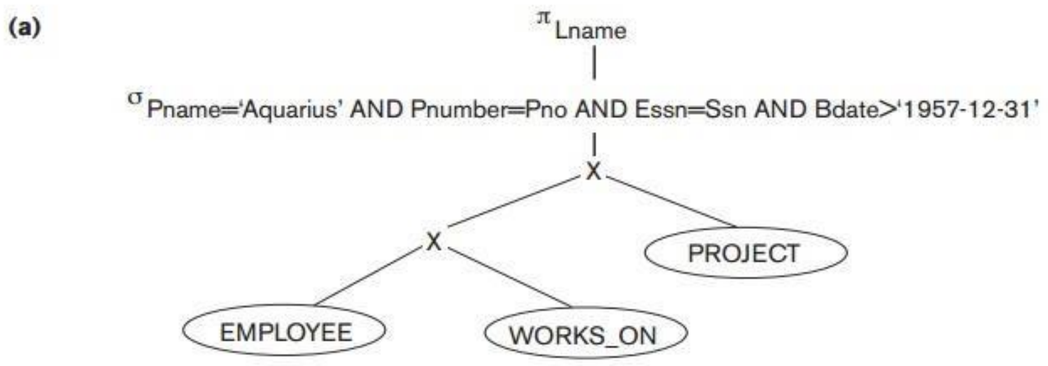
   Query "Find the last names of employees born after 1957 who work on a project named _Aquarius‘."

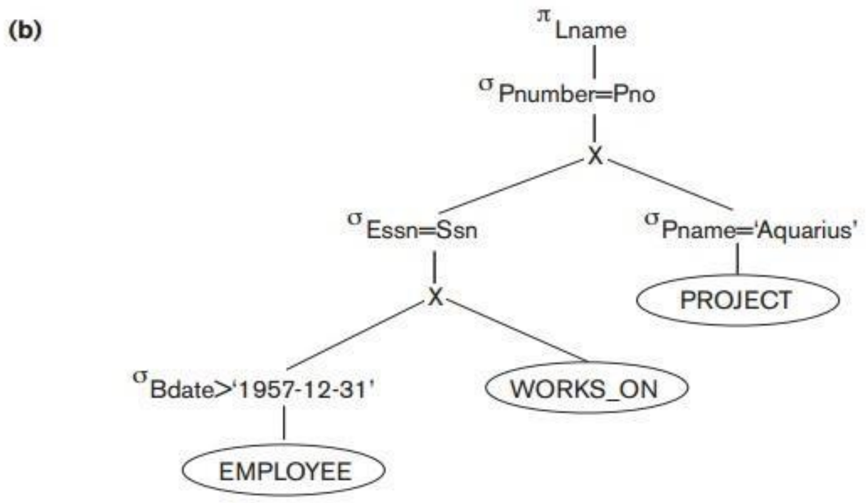
SQL

SELECT LNAME

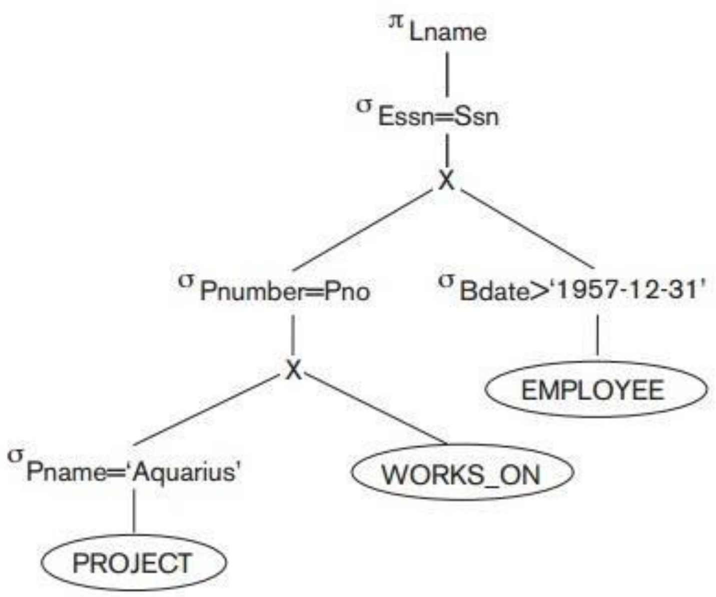
FROM EMPLOYEE, WORKS_ON, PROJECT

WHERE PNAME=_Aquarius‘AND PNUMBER=PNO AND ESSN=SSN AND BDATE._1957-12-31‘;

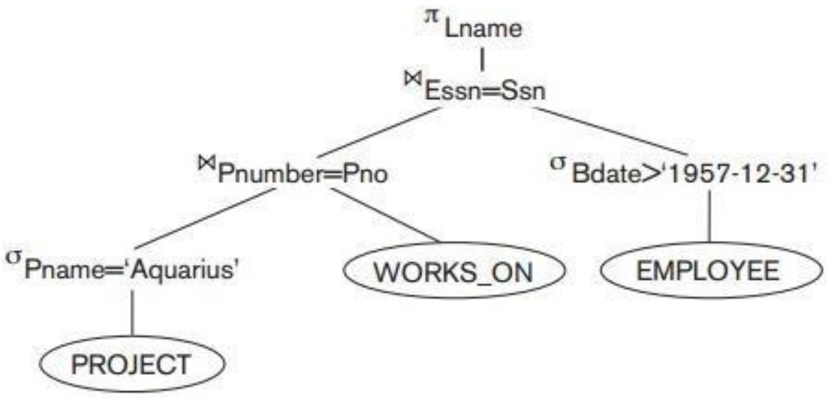

(a)
π Lname
σ Pname='Aquarius' AND Pnumber=Pno AND Essn=Ssn AND Bdate>'1957-12-31'
X
X
PROJECT
EMPLOYEE
WORKS_ON



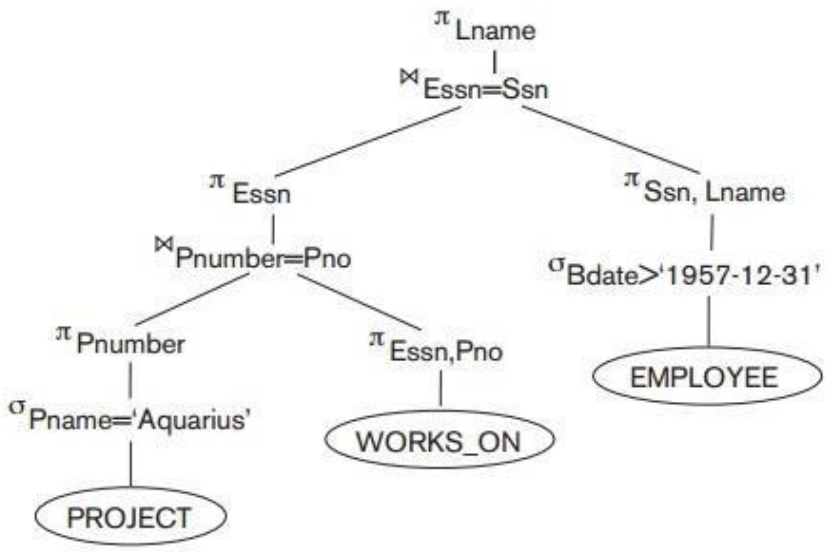
(c)


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

(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

Cost Components of Query Execution

The cost of executing the query includes the following components:

- Access cost to secondary storage.
- Storage cost.
- Computation cost.
- Memory uses cost.
- Communication cost.

Importance of Access cost

Out of the above five cost components, the most important is the secondary storage access cost.

- The emphasis of the cost minimization depends on the size and type of database applications.
- For example in smaller database the emphasis is on the minimizing computing cost as because most of the data in the files involve in the query can be completely store in the main memory.
- For large database, the main emphasis is on minimizing the access cost to secondary device.
- For distributed database, the communication cost is minimized as because many sites are involved for the data transfer.

– [nBlocks(R)/2], if the record is found.

– [nBlocks(R)], if no record satisfied the condition.

Binary Search :

[log2(nBlocks(R))], if equality condition is on key attribute, because SCA(R) = 1 in this case.

[log2(nBlocks(R))] + [SCA(R)/bFactor(R)] – 1, otherwise.

Equity condition on Primary key

– [nLevelA(I) + 1]

– [nLevelA(I) + 1] + [nBlocks(R)/2]

Cost functions for JOIN Operation

Join operation is the most time consuming operation to process.

- An estimate for the size (number of tuples) of the file that results after the JOIN operation is required to develop reasonably accurate cost functions for JOIN operations.
- The JOIN operations define the relation containing tuples that satisfy a specific predicate F from the Cartesian product of two relations R and S.