Chapter 19

Query Optimization

It is an activity conducted by the query optimizer to select the best available strategy for executing the query.

1. **Query Trees and Heuristics for Query Optimization**

- Apply heuristic rules to modify the internal representation of the query
- The scanner and parser of an SQL query first generates a data structure that corresponds to an initial query representation
- It is optimized according to some heuristics rules
- This leads an optimized query representation, which corresponds to the query execution strategy
- One of the heuristic rule is to apply selection and projection before join to reduce data space
- Query tree is used to represent relational algebra, query graph is used to represent relational calculus

Notation for Query Tree and Query Graphs

Q2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address and birth date.

SELECT Pnumber, Dnum, Lname, Address, Bdate

FROM PROJECT, DEPARTMENT, EMPLOYEE

WHERE Dnum = Dnumber AND Mgr_ssn = Ssn AND Plocation='Stafford';

Figure 19.1 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.

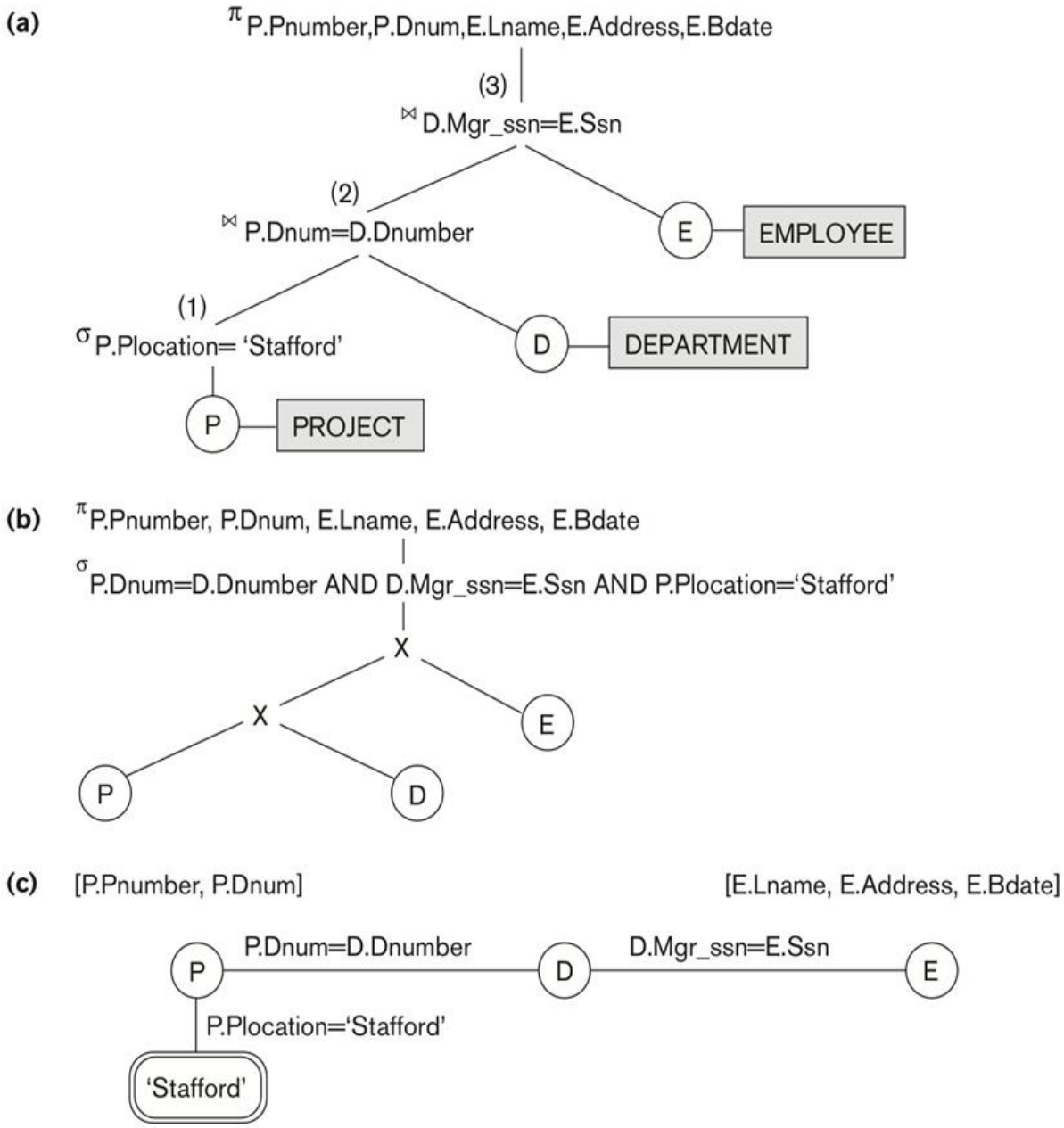


Figure 5.6 One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Figure 5.6 (continued) One possible database state for the COMPANY relational database schema.

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

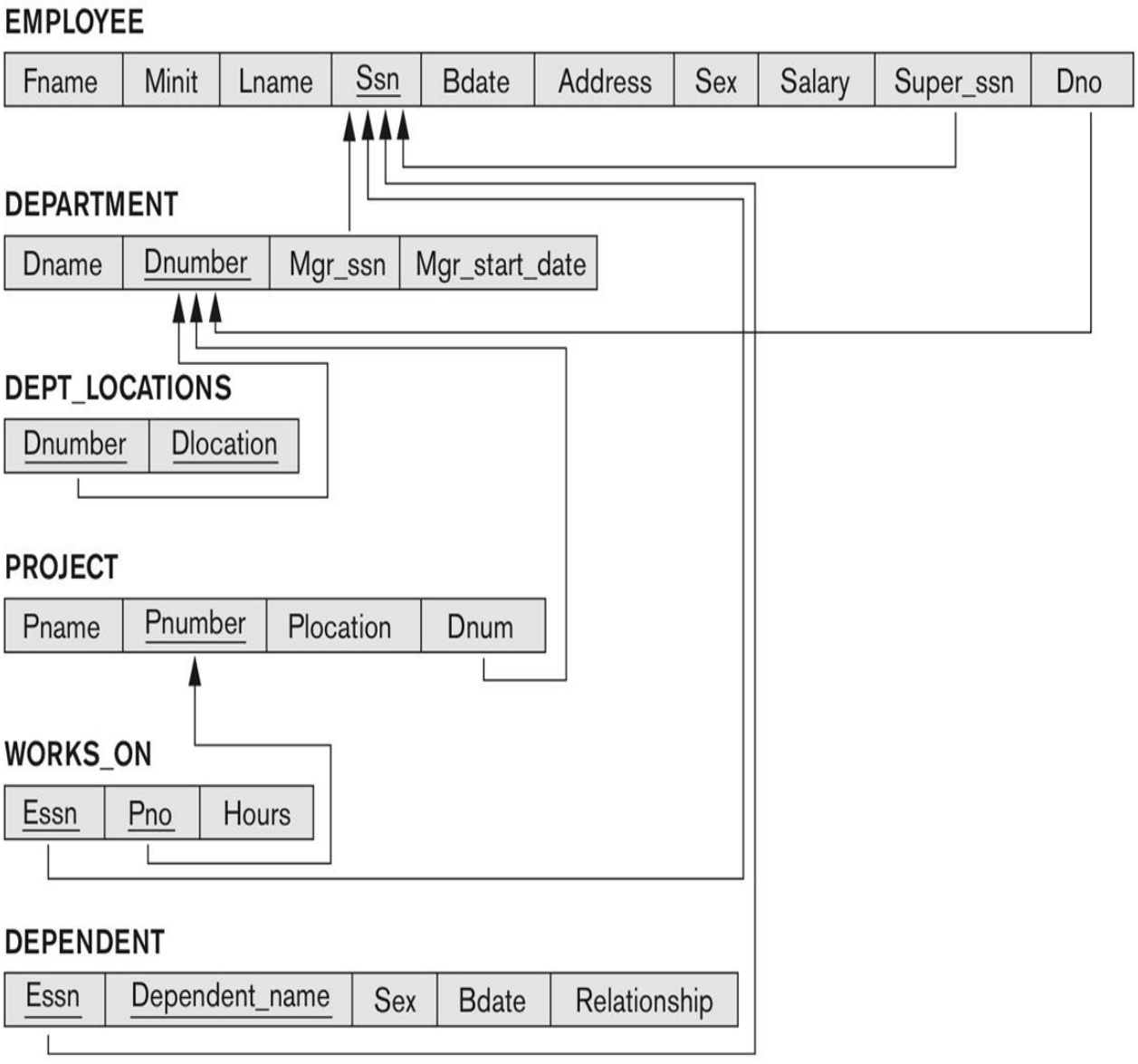
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Figure 5.7 Referential integrity constraints displayed on the COMPANY relational database schema.



- Query tree: relations at the leaf, operators at the nodes
- Perform operations until the root node is reached
- Node 1, 2, 3 must be executed in sequence
- Query tree represents a specific order of execution

Heuristic Optimization of Query Trees

- There are many query trees possible for a given relational algebraic expression
- Query parser will generate a standard query tree without doing any optimization

Example:

Find the last names of employees born after 1957 who work on a project name ‘Aquarius’.

SELECT E.Lname

FROM EMPLOYEE E, WORKS_ON W, PROJECT P

WHERE P.Pname = ‘Aquarius’ AND P.Pnumber = W.Pno AND E.Ssn=W.Ssn AND E.Bdate = ‘1957-12-31’;

Fig. 19.2(a) is the Query tree, not optimized

Fig. 19.2(b) is the Query tree with improvement

Fig. 19.2 (c) more improvement

Fig. 19.2 (d) more improvement

A query can be transformed step by step into an equivalent query that is more efficient to execute (need some rules to do this)

Figure 19.2a Steps in converting a query tree during heuristic optimization. Initial (canonical) query tree for SQL query Q.

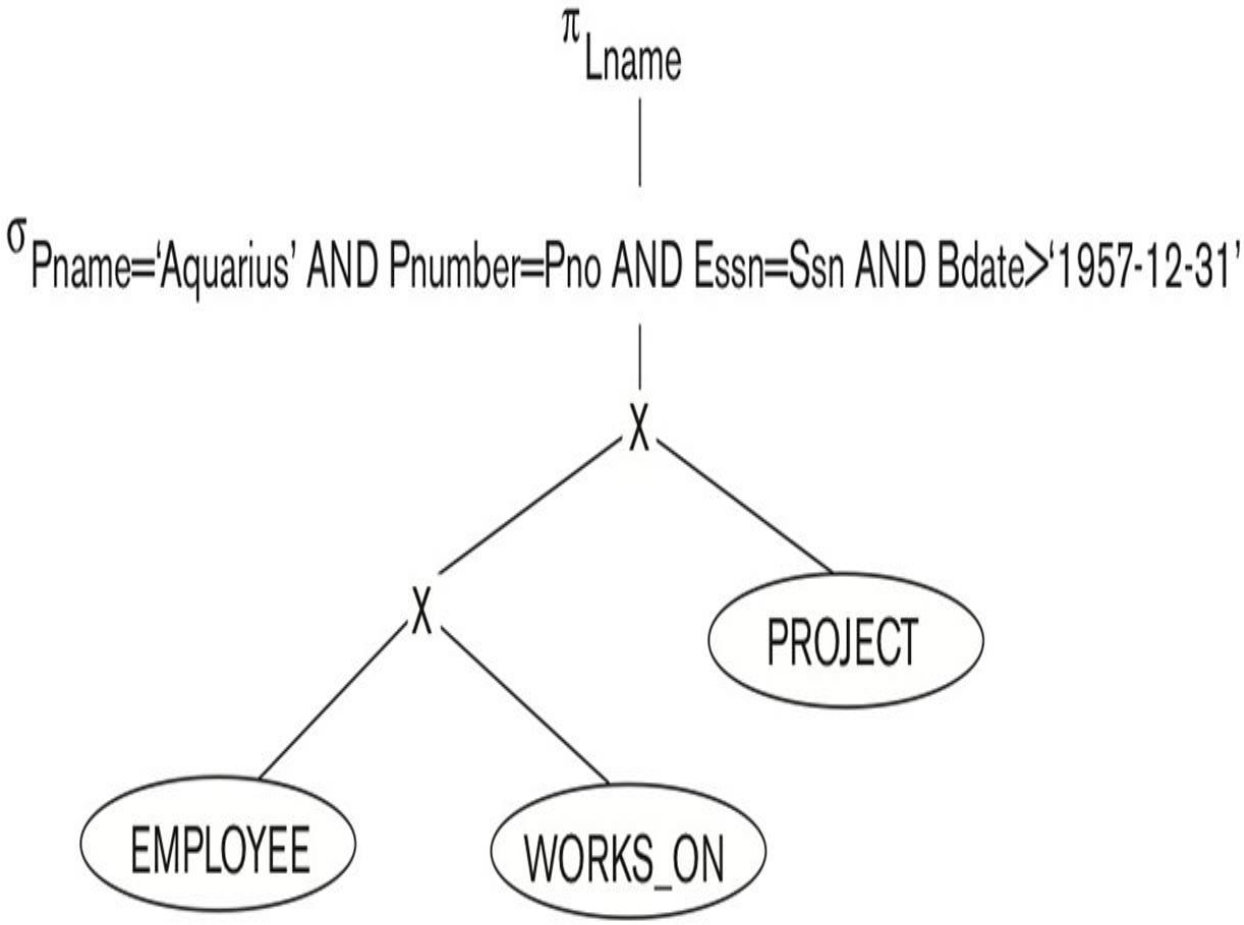


Figure 19.2b Steps in converting a query tree during heuristic optimization. Moving SELECT operations down the query tree.

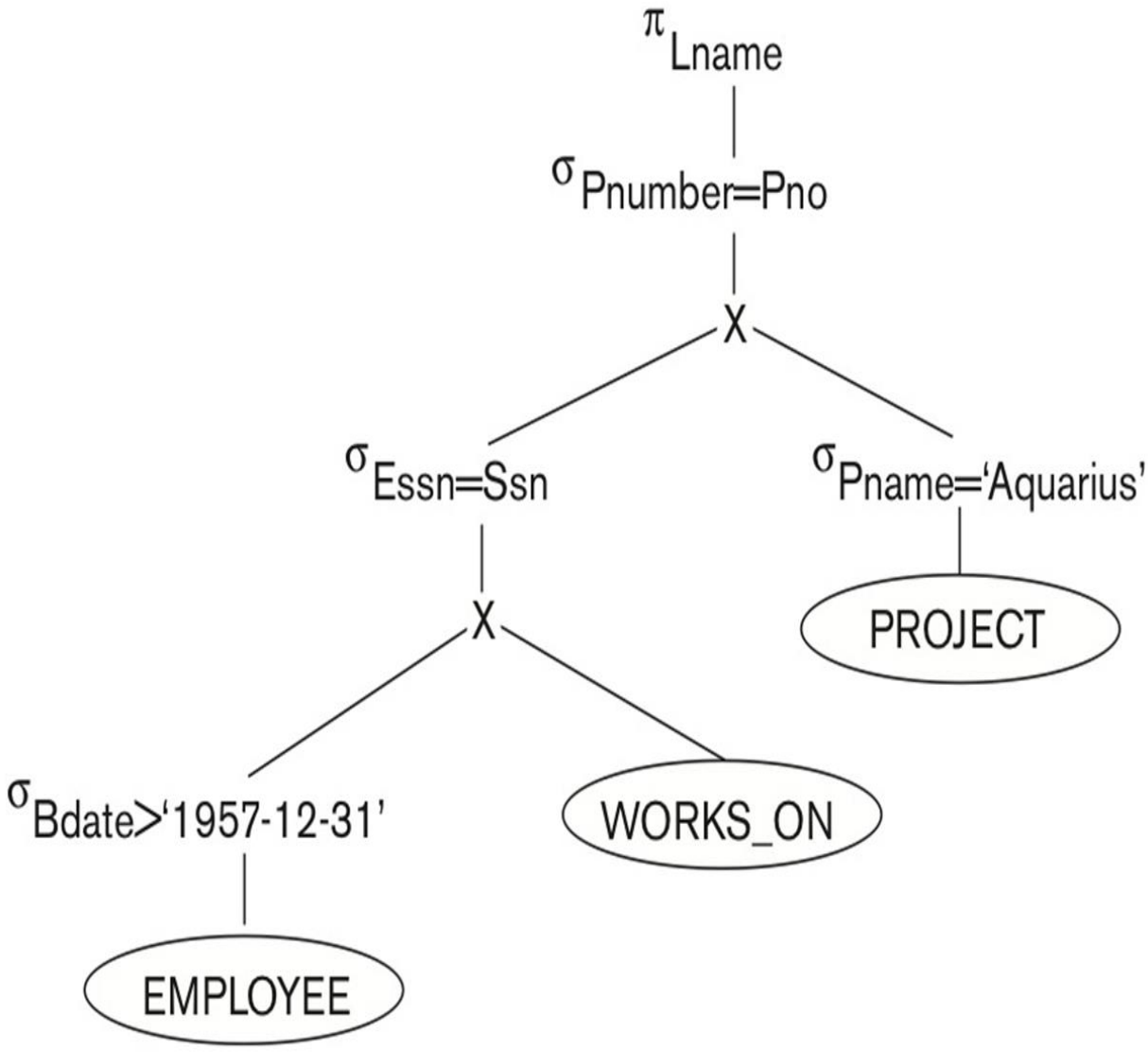


Figure 19.2c Steps in converting a query tree during heuristic optimization. Applying the more restrictive SELECT operation first.

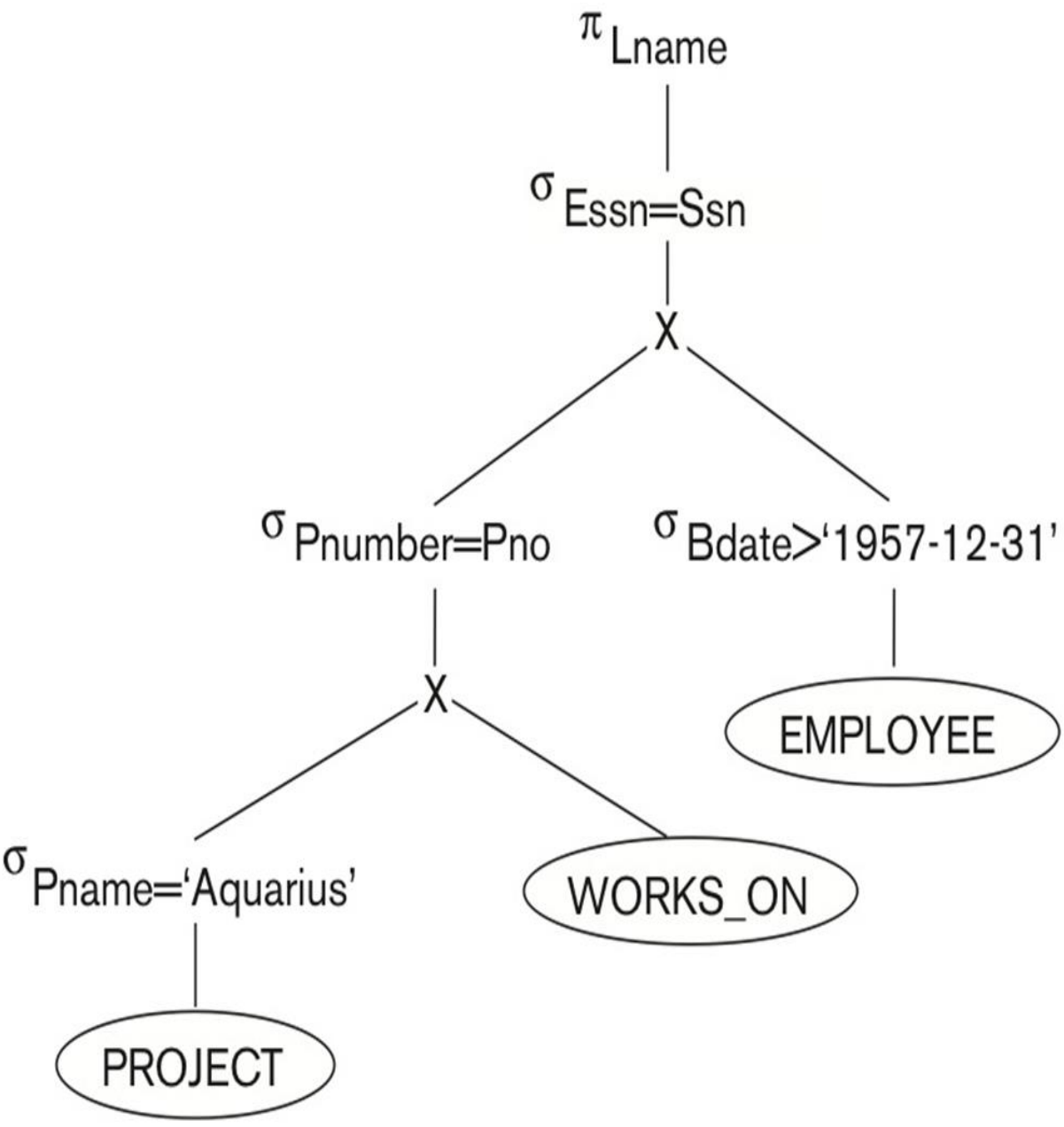


Figure 19.2d Steps in converting a query tree during heuristic optimization. Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.

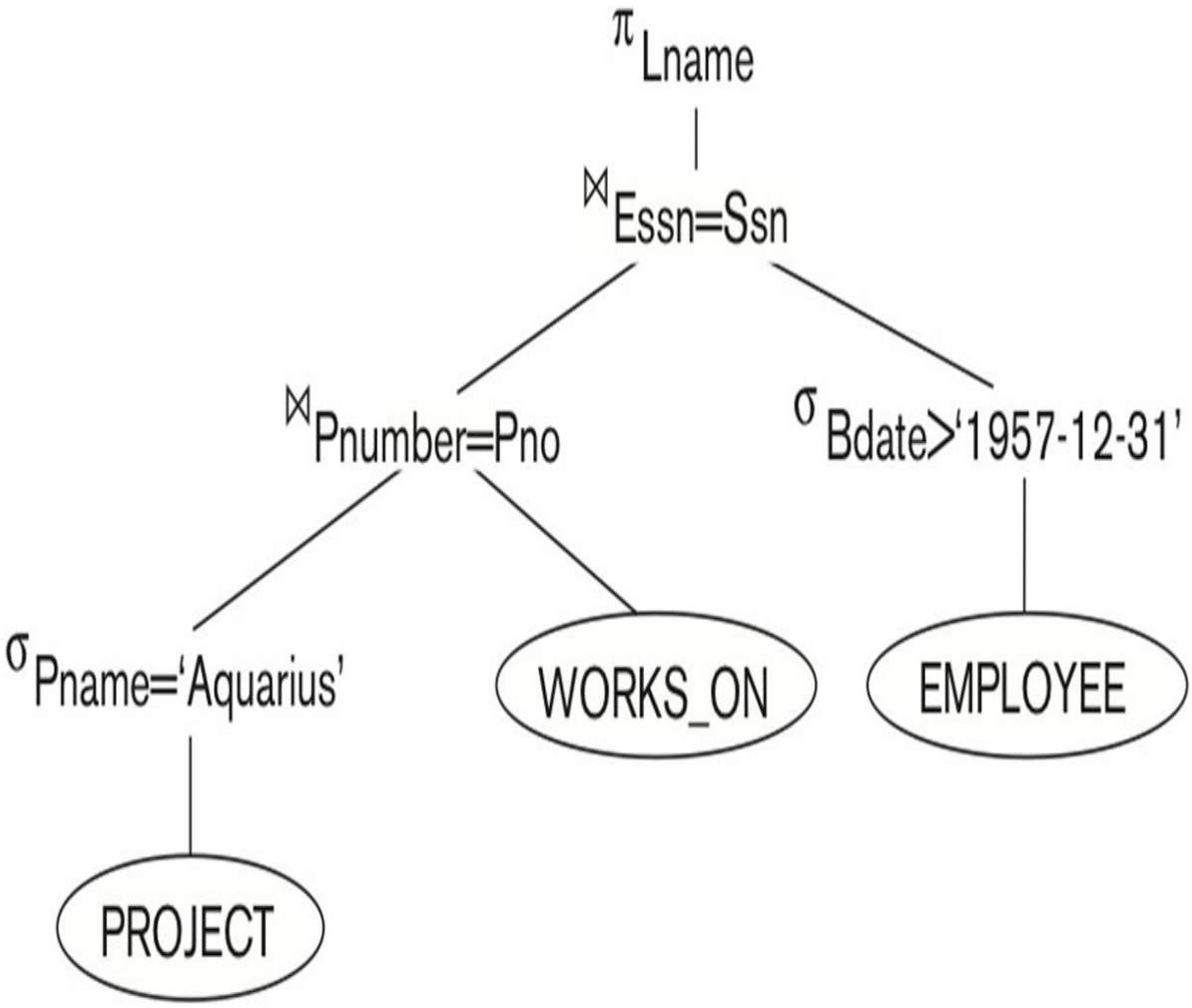
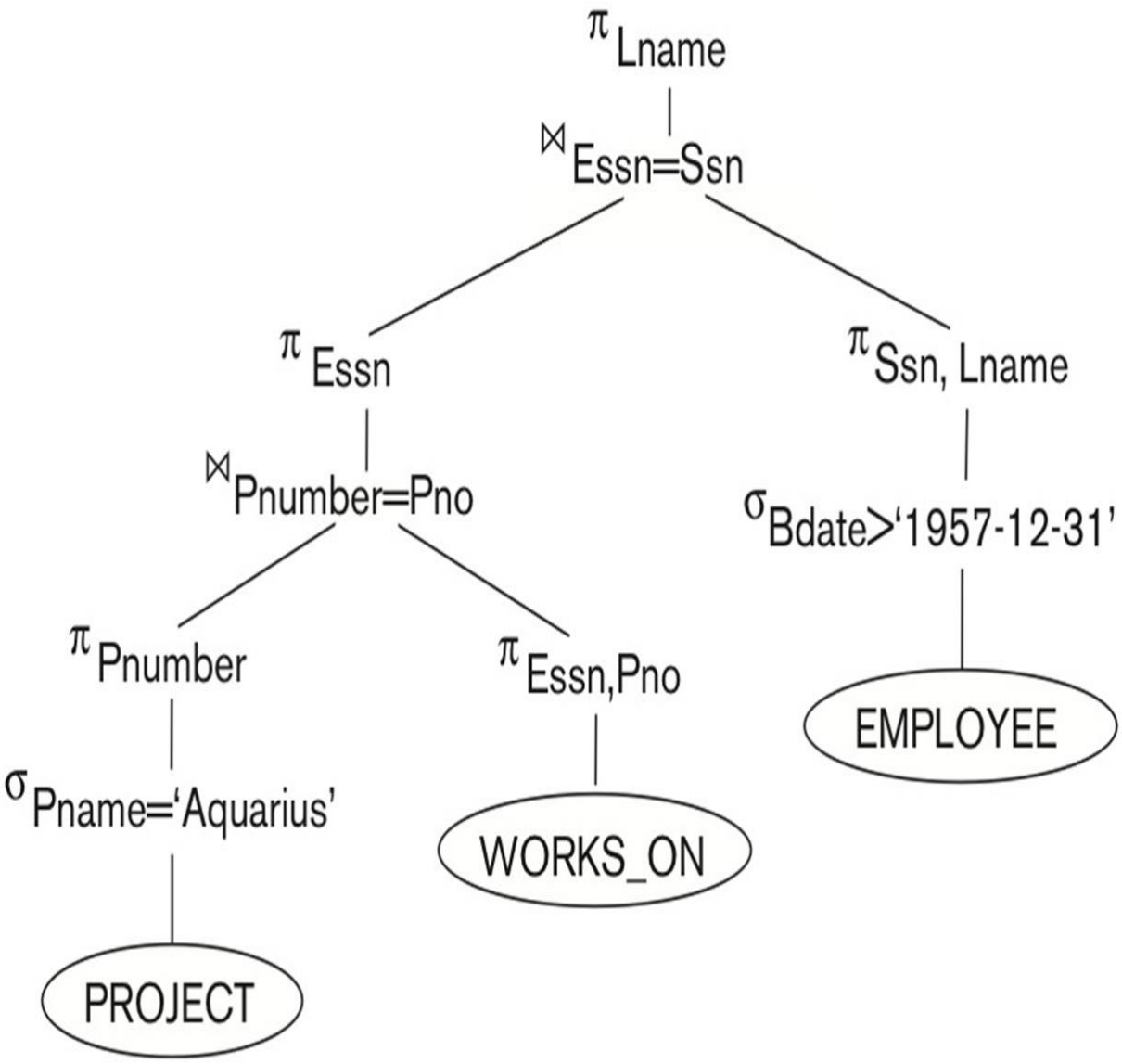


Figure 19.2e Steps in converting a query tree during heuristic optimization. Moving PROJECT operations down the query tree.



General Transformation Rules for Relational Algebra Operations

1. A conjunctive selection condition can be broken up into a cascade of individual σ operations.

$$\sigma_{c1 \text{ AND } c2 \text{ AND } c3}(R) = \sigma_{c1}(\sigma_{c2}(\sigma_{c3}(R)))$$

2. Commutative of σ : The σ operator is commutative,

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

3. Cascade of $\prod$

$$\prod_{List1}(\prod_{List2} \ldots\ldots(\prod_{Listn}(R))) = \prod_{List1}(R)$$

4. Commuting σ with $\prod$ If the selection condition c involves only those attributes A1, A2, …An in the projection list, the two operators can be commuted

$$\prod_{<A1, A2, \ldots An>}(\sigma(R)) = \sigma(\prod_{<A1, A2, \ldots An>}(R))$$

5. Commutativity of $\bowtie$ and X

$$R \bowtie S = S \bowtie R$$

$$R \times S = S \times R$$

6. Commuting σ_c with $\bowtie$

If the attributes in the selection condition c involve only the attributes of the one of the relation being joined then

$$\sigma_c(R \bowtie S) = (\sigma_c(R)) \bowtie S$$

if the selection condition c is c1 AND c2, c1 involves only attributes in R and c2 involves only attributes in S then

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

7. Commuting $\prod$ with $\bowtie$

Suppose L = (A1, A2, …,An, B1, B2, ..Bn) where
(A1, A2, …, An) are attributes of R
(B1, B2, …, Bn) are attributes of S,
If the join condition only involves attributes in L, then

$$\prod_L (R \bowtie_c S) = (\prod_{<A1, A2, \ldots An>} (R)) \bowtie (\prod_{<B1, B2, \ldots Bn>} (S))$$

8. Commutativity of set operations. The set operations union and intersection is commutative but difference is not

9. Associativity of JOIN, X, Union and Intersection: These are individually associative

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

10. Commuting selection with set operations: the σ operation commutes with union and intersection

    $\sigma (R \phi S) = (\sigma (R)) \phi (\sigma (S))$

11. The $\prod$ commutes with union

$\prod_L (R \cup S) = (\prod_L (R)) \cup (\prod_L (S))$

12. Converting a σ and X sequence into a JOIN

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

13. Pushing σ in conjunction with set difference

$\sigma_C (R - S) = \sigma_C (R) - \sigma_C (S)$

However, selection may be applied to only one relation.

$\sigma_C (R - S) = \sigma_C (R) - S$

14. Pushing σ to only one argument in η

If in the condition σ_C all attributes are from relation R, then

$\sigma_C (R \eta S) = \sigma_C (R) \eta S$

15. If S is empty, RUS = R

Outline of a heuristic optimization algorithm

1. Using rule 1, break up the select operation, this will allow moving selection down the tree at different branches
2. Using rules 2, 4, 6, 10, 13, 14 move each select operation as far down the query tree as is permitted by the attributes
3. Using rule 5 and 9, rearrange the leaf nodes
4. Using rule 12, combine X with a selection and JOIN
5. Using rule 3, 4, 7, 11 cascading of projection, breakdown and move down the tree
6. Reduce the size of intermediate results; perform selection as early as possible
7. Use projection attributes as early as possible to reduce number of attributes
8. Execute select and join operations that are more restrictive or result in less tuples

2. Choice of Query Execution Plans

Alternatives for Query Evaluation:

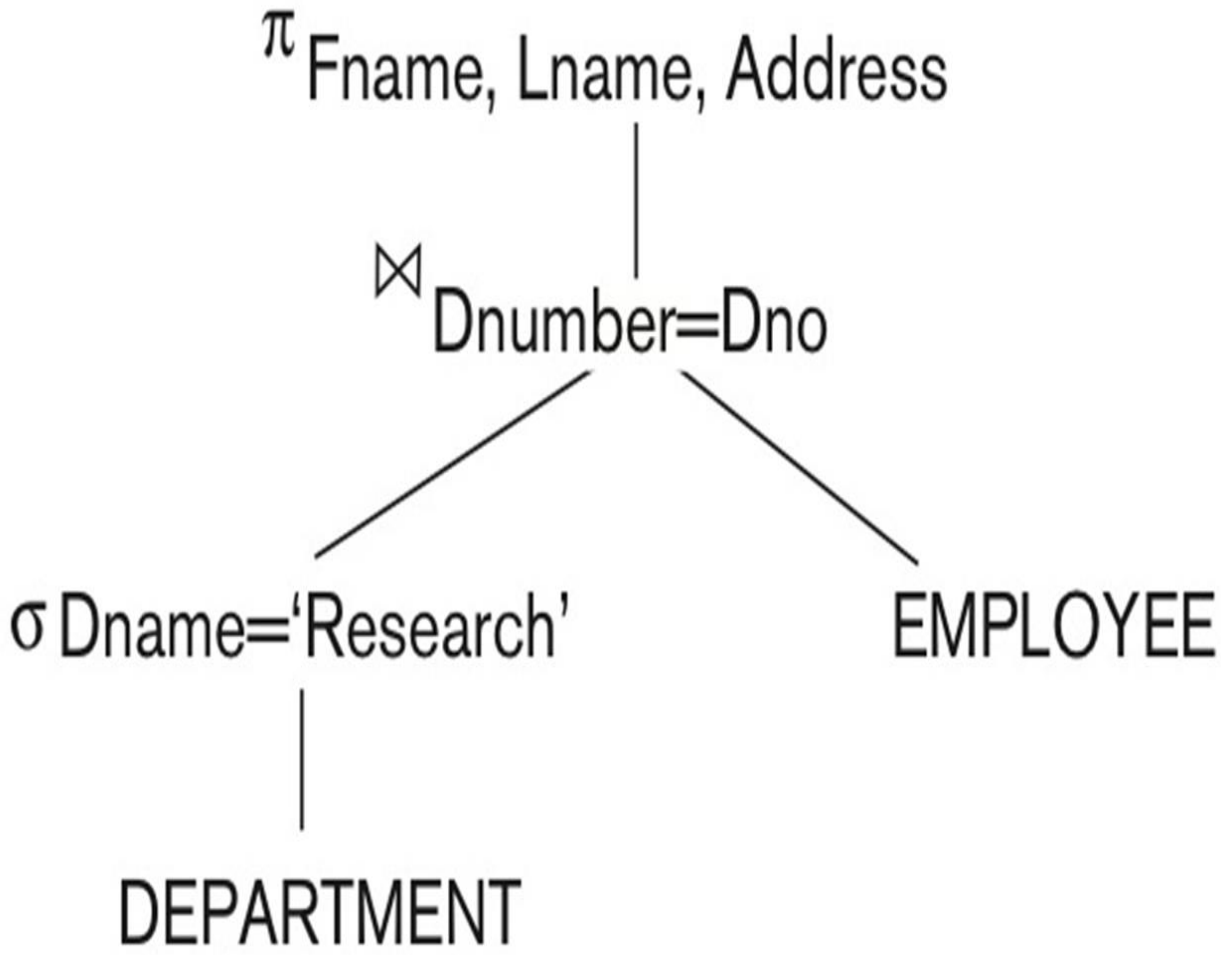
$\prod_{\text{Fname, Lname, Address}}$
$(\sigma_{\text{Dname='Reseach'}}(\text{DEPARTMENT}) \bowtie_{\text{Dnumber = Dno}} (\text{EMPLOYEE})$

Query tree is in Fig. 19.3

- Query tree includes the information about the access methods for each relation and algorithms to execute the tree
- To execute this query plan, optimizer might choose an index for SELECT operation on DEPARTMENT; assume also that an index

exits for Dno attribute of EMPLOYEE. Optimizer may also use a pipelined operation.

Figure 19.3 A query tree for query Q1.



For materialized evaluation, the result is stored as a temporary (intermediate) relation.
For example, Join operation can produce intermediate results and then apply projection. In a pipelined operation, one operation results are fed to the next operation as they arrive.

Nested Subquery Optimization
Consider the query:

```
SELECT E1.Fname, E1.Lname
FROM EMPLOYEE E1
WHERE E1.Salary = (SELECT MAX(Salary)
                   FROM EMPLOYEE E2);
```

Inner qury evaluates and get E.Salary = M (max salary in E2)
Then the outer query runs for each employee in E1.

The inner query is evaluated once and the outer query uses this value. This is a non-correlated query.

Consider the query:

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE E
WHERE EXISTS (SELECT *
              FROM DEPARTMENT D
              WHERE D.Dnumber = E.Dno AND
D.Zipcode=30332)
```

The inner query returns true or false depending on whether the employee working in a department lives in 30332 zip code area or not.

The inner subquery has to be evaluated for every outer query tuple, which is inefficient.

The above nested query can be written as a block query (unnesting) using a Join operation.

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.Dnumber=E.Dno AND D.Zipcode=30332
```

Temporary intermediate results can be used to perform join operation. Usually, nested queries can be converted in to block queries.

SKIP (Views and Merging)

3. Use of Selectivities in Cost-based Optimization

A query optimizer estimates and compares costs of executing queries using different strategies. It then uses the strategy with lowest cost. Number of strategies have to be limited to save compute time.

Queries are either interpreted or compiled. The cost functions are estimated. This approach is applicable to compiled queries as it consumes more time at run time.

The query cost components are stored in the catalog to speed up the process.

Overall cost-based query optimization

1. For a given query expression, multiple equivalent rules may exist; there is no definitive convergence; it is difficult to do this in a limited time
2. It is necessary to use some quantitative measures for evaluating alternatives; reduce them to common metric called cost
3. Keep cheaper ones and prune the expensive ones
4. Scope is a query block; various index access paths, join permutations, join methods, group by methods, etc…
5. In global query optimization, multiple query blocks used

Cost components for query execution

1. Access cost to disk (disk I/O, searching for a record, access structures, hashing, indexes, file allocation methods, etc..
2. Disk storage cost: any intermediate file storage cost
3. Computation cost: CPU time (processing time), searching, merging, performing computations on fields, etc.
4. Memory usage cost: no of buffers
5. Communication cost: cost of shipping data from the source of query location to the server

Minimize access cost to disk…for large databases

For small database, it is stored in memory

Catalog information used in cost functions

Most of the information required is stored in the catalog so that the query optimizer can access it.

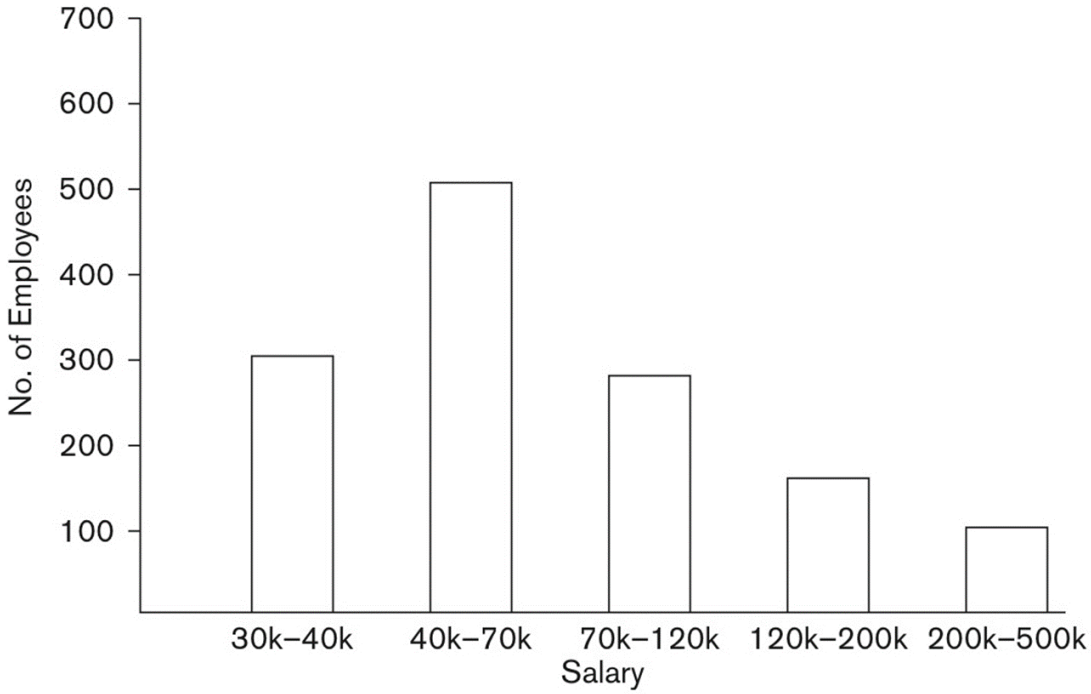
- Number of records (tupes) (r)
- Average record size R
- The number of file blocks (b)

- The blocking factor (bfr)
- Primary file organization
- Indexes
- The number of levels of multiindex x
- The number of first level index blocks (bl1)
- The number of distinct values of an attribute (d)
- Selectivity of an attribute (sl)
- Avg number of records that satisfy an equality condition on an attribute (selection cardinality s=sl*r; selectivity and no of records)

Histograms

- Tables or data structures maintained by DBMS to record distribution of data
- Without histograms, it is uniform distribution

Figure 19.4 Histogram of salary in the relation EMPLOYEE.



Examples of cost functions for SELECT

S1: Linear search (brute force) approach

- $C_{s1a} = b$
- For an equality condition on a key (on the average it is found that) $C_{s1a} = b/2$ if found otherwise $C_{s1a} = b$

S2: Binary search:

- $C_{s2} = \log_2 b + \lceil (s/bfr) \rceil - 1$
- For an equality search on a unique key attribute $C_{s2} = \log_2 b$

S3: Using a primary index or a hash key to retrieve a single record:

- $C_{s3a} = x + 1$
- $C_{s3b} = 1$ for static or linear hashing
- $C_{s3b} = 2$ for extendible hashing

Examples of cost functions for JOIN

- JOIN selectivity (js)
- R ⋈ S (A=B R.A = S.B) join condition C
- js = | R ⋈ S | / |R X 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 after join operation
- | R ⋈ S | = js * |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

- If A is a key of R (A=B condition)

  $|R \bowtie S| <= |S|$; $js = 1/|R|$

- If B is a key of S (A=B condition)

  $|R \bowtie S| <= |R|$; $js = 1/|S|$

J1: Nested Loop Join:

- $C_{j1} = b_R + (b_R * b_S) +$
  $((jS * |R| * |S|)/bfr_{RS})$ (writing results to disk)
  (R for outer loop, RS for resulting file)

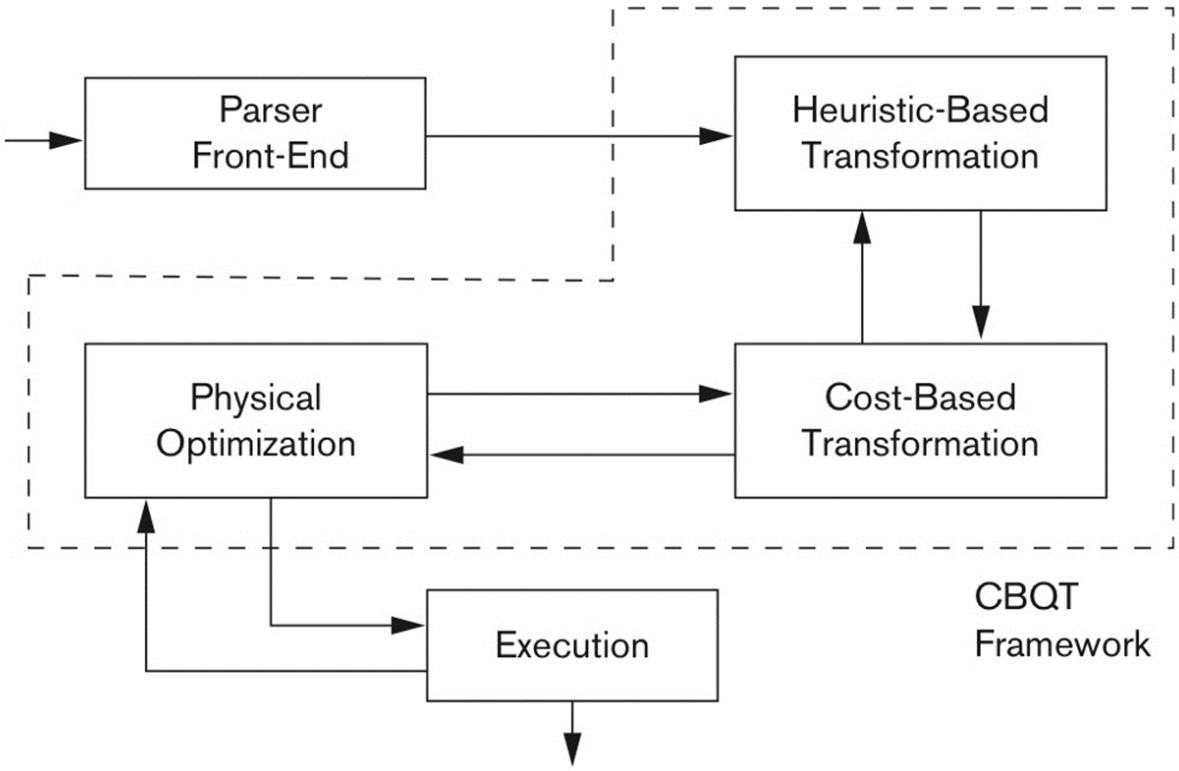
4. Overview of Query Optimization in Oracle

- Cost based, introduced in Oracle 7.1
- It examines alternate table and index access paths, operator algorithms, join ordering, join methods, parallel execution distributed methods and so on
- Chooses the execution plan with the lowest estimated cost
- Estimated cost is a relative number proportional to the expected elapsed time needed to execute the query with a given plan

Optimizer calculates cost based on:

- Object statistics
  - Table cardinalities
  - Number of distinct values in columns
  - Column high and low values
  - Data distribution of column values
  - Estimated usage of resources (CPU and I/O time)
  - Memory needed
- Estimated cost
  - Internal metric
  - Corresponds to the run time and required resources
  - Find the best combination of lowest run time and least resource utilization

Figure 19.7 Cost-based query transformation framework (based on Ahmed et al., 2006).



Global Query Optimizer

- Query optimization consists of logical and physical phases
- In Oracle, logical transformation and physical optimization are integrated to generate optimal execution plan (Fig. 19.7)
- Transformation can be heuristic based on cost based
- Cost based query transformation (CBQT) introduced in 10g.
- Applies one or more transformations
- An SQL statement may consist of multiple blocks, which are transformed by physical optimizer

- This process is repeated several times, each time applying a different transformation and its cost is computed
- At the end one or more transformations are applied to the original SQL statement if they result in optimal execution plan
- To avoid combinatorial explosion, it provides efficient search strategies for searching the state space of various transformations
- Major transformations include: group-by, distinct, subquery merging, subquery unnesting, predicate move aroung, common subexpression elimination, join predicate push down, OR expansion, subquery coalescing, join factorization, subquery removal through window function, start transformation, group-by placement, and bushy join trees

Adaptive Optimization

- Oracle's physical optimizer is adaptive
- Uses feedback loop from execution level to improve on its previous decisions (backtrack)
- Optimal execution plan for a given SQL statement (uses object statistics, system statistics)
- Optimality depends on accuracy of the statistics fed to the model and the sophistication of the model itself
- Execution engine and physical optimizer has the feedback loop (Fig. 19.7)
- Based on the estimated value of the table cardinality, optimizer may choose index based nested loop join method; during execution, the actual cardinality may be different from the estimated value; during the execution, this may trigger the physical optimizer to change the decision to use hash join method instead of index join.

Array Processing

- Oracle lacks N-dimensional array based computation
- Extensions are made for OLAP features
- Improves performance in complex modeling and analysis
- Computation clause allows a table to be treated as a multi-dimensional array and specify a set of formulas over it, the formulas replace multiple joins and union operations

Hints

- Application developer can provide hints to query optimizer (query annotations or directives)
- Hints are embedded in the text of query statement
- Hints are used to address infrequent cases to help optimizer
- Occasionally, application developer can override the optimizer in case of suboptimal plans chosen by the optimizer
- E.g EMPLOYEE record 'Sex' attributes may assume half male and half female; it is possible in the database all are male; then application developer can specify that to optimize the column index
- Some types of hints:
    - The access path for a given table
    - The join order for a query block
    - A particular join method for a join between tables
    - Enabling or disabling of tranformations

Outlines

- Outlines are used to preserve execution plans of SQL statements or queries
- They are implemented as a collection of hints

- Outlines are used for plan stability, what-if analysis, and performance experiments

SQL Plan Management

- Execution plans have a significant impact on overall performance of a database management system
- SQL Plan Management (SPM) was introduced in Oracle 11g
- This option can be enabled for all execution plans or for a specific SQL statements
- Execution plans may become obsolete due to a variety of reasons; new statistics, configuration parameter changes, software updates, new optimization techniques
- SMP will use optimal plans and avoid semi-optimal ones, create new plans and add to the system as needed

5. Semantic Query Optimization

- Along with other optimization techniques, semantic query optimization uses constraints specified on the database schema
- These constraints are used to generate more efficient queries to execute
- Consider:

```
SELECT E.Lname, M.Lname
FROM EMPLOYEE AS E, EMPLOYEE AS M
WHERE E.Super_ssn=M.Ssn AND E.Salary>M.Salary;
```

(Retrieve the names of employees who earn more than their supervisors)

If there is a constraint indicating that employees can't earn more than their supervisors, then there is no need to execute the able query.

- Consider another example:

```
SELECT Lname, Salary
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.Dno=DEPARTMENT.Dnumber AND
  EMPLOYEE.Salary > 100000;
```

It can be rewritten:

```
SELECT Lname, Salary
FROM EMPLOYEE
WHERE EMPLOYEE.Dno IS NOT NULL AND
     EMPLOYEE.Salary > 100000;
```

(The referential integrity constraint that EMPLOYEE Dno is a foreign key that refers to DEPARTMENT Dnumber primary key. All the attributes referenced in the query are from EMPLOYEE. Thus, there is no need for DEPARTMENT and it can be eliminated and there is no need for join.