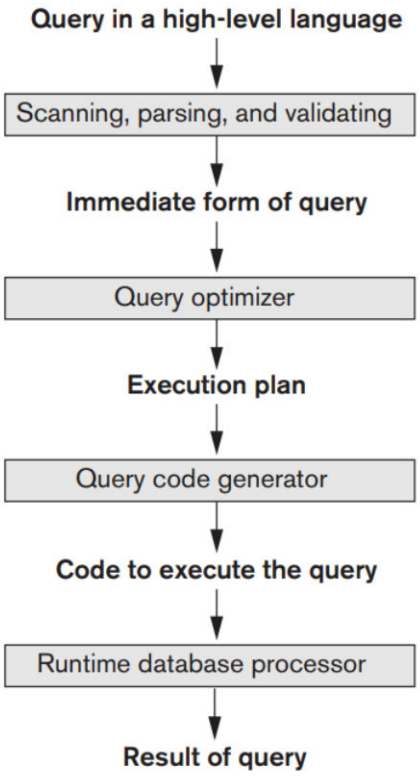
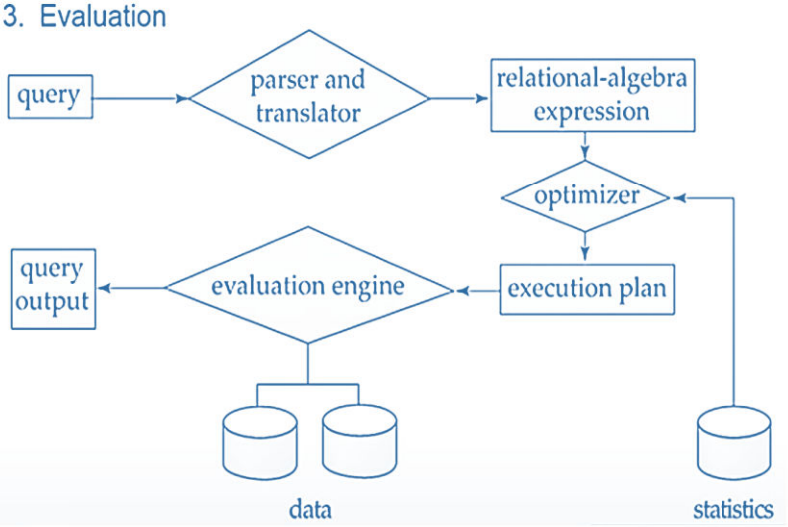
INTRODUCTION TO QUERY PROCESSING AND OPTIMIZATION

All database systems must be able to respond to requests for information from the user—i.e., process queries. Obtaining the desired information from a database system in a predictable and reliable fashion is the scientific art of Query Processing. Getting these results back in a timely manner deals with the technique of Query Optimization.



Typical steps when processing a high-level query

1. Parsing and translation
2. Optimization
3. Evaluation



There are three phases that a query passes through during the DBMS's processing of that query:

1. Parsing and translation
2. Optimization
3. Evaluation

- Parsing and translation
  - translate the query into its internal form. This is then translated into relational algebra.
  - Parser checks syntax, verifies relations
- Evaluation
  - The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.

- A relational algebra expression may have many equivalent expressions
  - E.g., $\sigma_{salary<75000}(\Pi_{salary}(instructor))$ is equivalent to $\Pi_{salary}(\sigma_{salary<75000}(instructor))$
- Each relational algebra operation can be evaluated using one of several different algorithms
  - Correspondingly, a relational-algebra expression can be evaluated in many ways.
- Annotated expression specifying detailed evaluation strategy is called an **evaluation-plan**. E.g.:
  - Use an index on *salary* to find instructors with salary < 75000,
  - Or perform complete relation scan and discard instructors with salary ≥ 75000

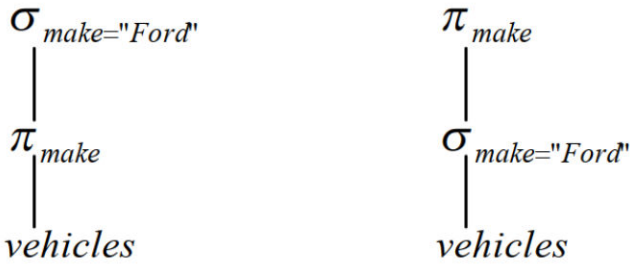
THE QUERY PROCESSOR

select *make* **from** *vehicles*
where *make* = “Ford”

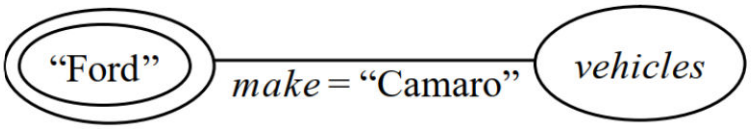
This can be translated into either of the following relational algebra expressions:

$$\sigma_{make="Ford"}(\pi_{make}(vehicles))$$
$$\pi_{make}(\sigma_{make="Ford"}(vehicles))$$

Which can also be represented as either of the following query trees:



And represented as a query graph:



After parsing and translation into a relational algebra expression, the query is then transformed into a form, usually a query tree or graph, that can be handled by the optimization engine. The optimization engine then performs various analyses on the query data, generating a number of valid evaluation plans. From there, it determines the most appropriate evaluation plan to execute. After the evaluation plan has been selected, it is passed into the DMBS' query-execution engine (also referred to as the runtime database processor), where the plan is executed and the results are returned.

Parsing and Translating the Query

- The primary job of the parser is to extract the tokens from the raw string of characters and translate them into the corresponding internal data elements (i.e. relational algebra operations and operands) and structures (i.e. query tree, query graph).

- The last job of the parser is to verify the validity and syntax of the original query string.

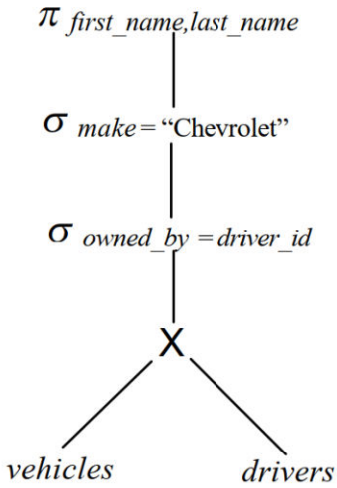
Expression and Tree Transformations

select *first_name, last_name* **from** *drivers, vehicles*
where *make* = "Chevrolet" and *owned_by* = *driver_id*

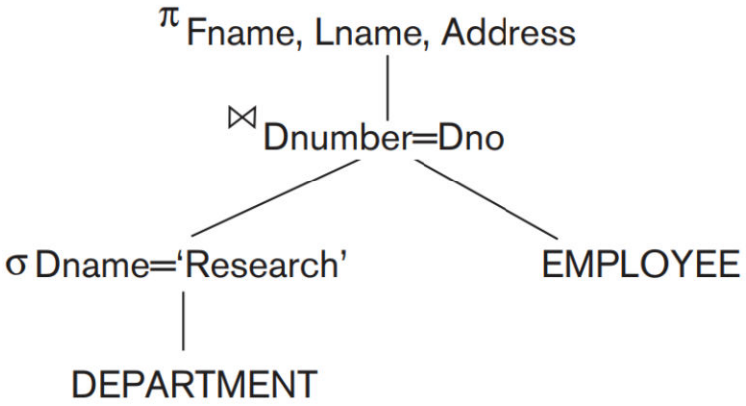
A corresponding relational algebra expression is:

$\pi_{first\_name,last\_name}((\sigma_{make="Chevrolet"}(\sigma_{owned\_by=driver\_id}(vehicles \times drivers)))$

And the corresponding canonical query tree for the relational algebra expression:



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



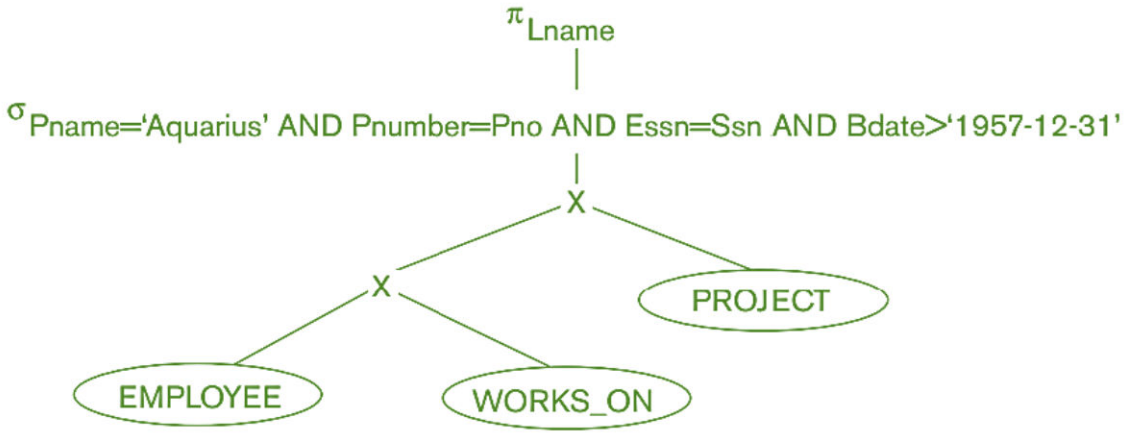
Query Trees and Heuristics for Query Optimization

The scanner and parser of an SQL query first generate a data structure that corresponds to an *initial query representation,* which is then optimized according to heuristic rules. This leads to an *optimized query representation*, which corresponds to the query execution strategy. Following that, a query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

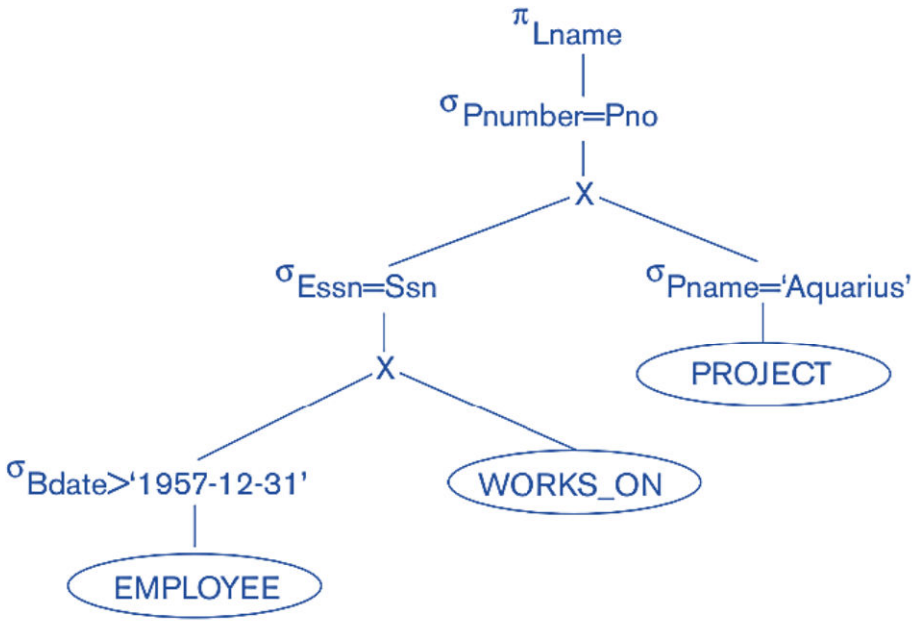
One of the main heuristic rules is to apply SELECT and PROJECT operations before applying the JOIN or other binary operations, because the size of the file resulting from a binary operation—such as JOIN—is usually a multiplicative function of the sizes of the input files. The SELECT and PROJECT operations reduce the size of a file and hence should be applied before a join or other binary operation.

Steps in converting a query tree during heuristic optimization

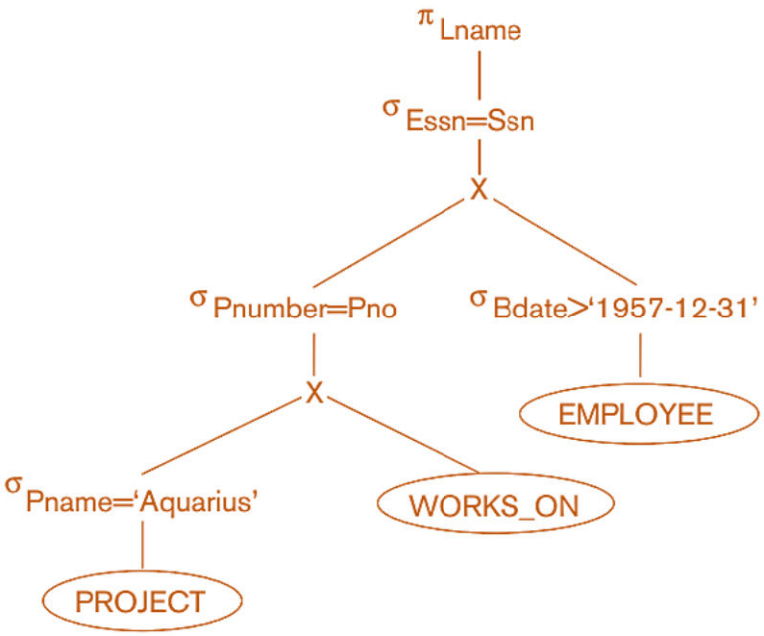
SELECT E.Lname FROM EMPLOYEE E, WORKS_ON W, PROJECT P
WHERE P.Pname='Aquarius'
AND P.Pnumber=W.Pno
AND E.Essn=W.Ssn
AND E.Bdate > '1957-12-31';



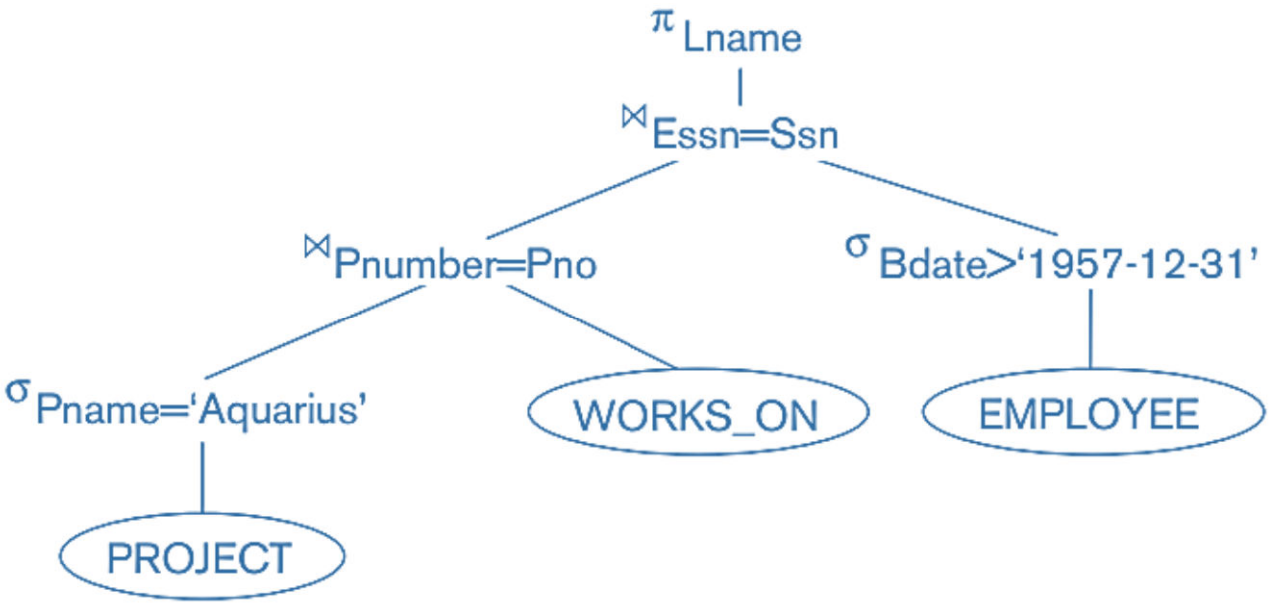
Initial (canonical) query tree for SQL query



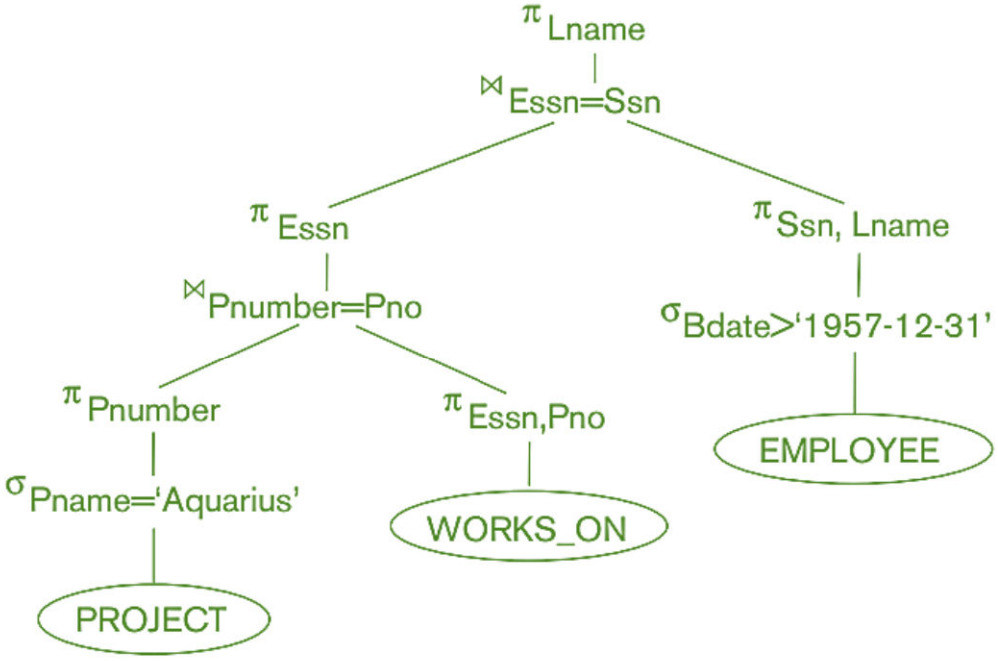
Moving SELECT operations down the query tree.



Applying the more restrictive SELECT operation first



Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.



Moving PROJECT operations down the query tree