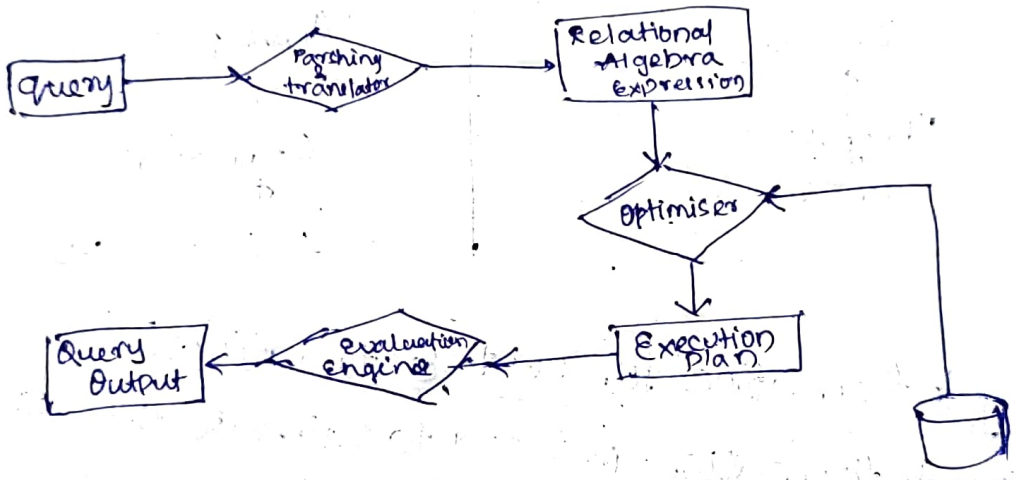
Query processing:

→ Query is the vehicle for instructing a DBMS to update or retrieve specific data to/from the physical stored medium.

→ Obtaining the desired information from a database system in a predictable and reliable fashion is scientific art of Query processing.

It has 3 steps:
(i) Parsing and Translation
(ii) Optimization
(iii) Evaluation



(*) Parsing and Translating the Query:

→ The first step is to convert the query into a low-level language

→ The main job of a parsher is to extract the information and convert to internal data elements and also verify the validity and syntax of original query string.

(*) The relational algebra expression is passed to the optimiser for optimising the query.

(*) Optimising the Query:

→ With the help of catlog statistic meta data the best query is selected and send for execution and evaluation process.

⊛ Evaluating the query:

→ On reaching to the execution plan it is converted into query tree.

→ The final step is processing the query.

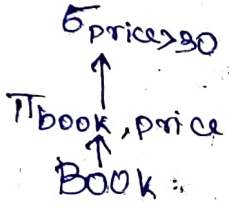
→ The best evaluation plan candidate generated by the optimisation engine is selected and then executed. (based on the cost)

Query tree
- I/P as leaf nodes
- Operators as internal nodes

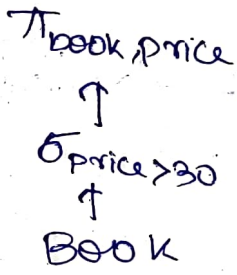
For ex:

Select book, price from Book where price > 30.

$\sigma_{price>30}(\pi_{book,price}(Book))$



$\pi_{book,price}(\sigma_{price>30}(Book))$



⊛ Catlog Information for cost Estimation:

→ Catlog information is used to estimate the cost of executing queries in a database system.

→ It helps to determine selectivity factors for operations.

→ Used by query optimizers to evaluate query plans and choose the most efficient one.

⊛ Measures of Query Cost

The execution time of a query depends on:

(i) Disk accesses
(ii) CPU cycles
(iii) RAM
(iv) Parallel and distributed systems

Buffer: Temporary storage area in main memory

⊙ Disk I/O Costs:

→ Most critical and resource intensive part of query execution.
→ Reads pages from disk into memory
→ Writing pages back to disk
→ No. of pages read/written directly relates to the query plan.

⊙ CPU costs:

→ Smaller than disk I/O but becomes significant for large datasets or complex queries

⊙ Network Costs (for distributed systems):-

→ Span multiple nodes, additional costs arise.

✱ Selection Operation

→ Retrieves rows from a relation that satisfy a specified condition
→ Represented by $\sigma_{condition}(relation)$

Factors affecting it:

① Access method:
- Full table scan
- Index Scan

② Condition type.
- Equality cond^n (=)
- Range cond^n
- Complex cond^n

③ Selectivity factor
- fraction of tuples satisfying the condition
- Higher selectivity reduce I/O costs.

Implementation methods:

(i) Linear Search : $O(n)$
(ii) Binary Search : $O(\log n)$
(iii) Index lookup : $O(\log_b n)$

✱ Join Operations:

→ Combines tuples from two relations based on a condition

$R \bowtie_{condn} S$

Types: Theta, Equi, Natural, Outer

✱ Equivalence rules in Relational Algebra:

→ Allows transformations of relational algebra expressions to generate efficient query execution plans.

Key Equivalence rules:

① Commutativity:
- Selection :- $\sigma_{cond1}\,\sigma_{cond2}(R) = \sigma_{cond2}\,\sigma_{cond1}(R)$
- Join :- $R \bowtie S = S \bowtie R$

② Associativity:-
- Join :- $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

③ Distributive:
- Selection over join

$$\sigma_{cond}(R \bowtie S) = (\sigma_{condn}(R)) \bowtie (\sigma_{condn}(S))$$

④ Union/intersection commutativity:

$R \cup S = S \cup R$

$R \cap S = S \cap R$

⑤ Union/Intersection Associativity:

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

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

⑥ Selection pushdown:

$\sigma_{cond1\ AND\ cond2}(R) = \sigma_{cond1}(\sigma_{cond2}(R))$

Benefits:-

① Query Optimisation:

→ Reduce computation cost by transforming queries into equivalent, efficient forms.

② Parallel Execution:

→ Helps identify operations that can be performed independently.

③ Reusability:

→ Identifies reusable subqueries or operation.

✱ Query Optimization Process

→ Process of choosing the most efficient strategy to execute a database query by considering multiple execution plans.

→ The goal is to minimize resource usage while maximising performance.

⊙ Query Optimizing phases:

ⓐ Query Parsing:

→ User written SQL query
→ Process SQL into parse tree.
→ Check syntax errors and semantic correctness.

ⓑ Logic Query Optimization:

→ Parse tree as input
→ Applies relational algebra equivalence rules to transform query into logically equiv. forms.
→ Reduces intermediate results.

ⓒ Physical Query Optimization:

→ logic query plan
→ Chooses the best execution plan
→ Considers cost estimation

ⓓ The database engine executes the chosen plan, retrieves data and presents results.

① Query Optimization Techniques:

② Logical Optimization techniques:

→ Transform the query to reduce computational & I/O overhead.

ⓐ Selection pushdown:

→ Moves selection process as close as possible to base relations.

ⓑ Projection pushdown

→ Pushes projection operations down to avoid carrying unnecessary attributes through query steps.

ⓒ Join Reordering

→ Reorders join operations to minimize the size of intermediate results.

ⓓ Physical Optimization Technique:

- Index Usage
- Join algorithm
- Query caching
- Bitmap indexing
- Parallel query execution

Benefits:

① Reduced execution time
② Resource efficiency
③ Scalability
④ Improved user experience.