QUERY PROCESSING

DATABASE MANAGEMENT SYSTEM

SUJAN TAMRAKAR

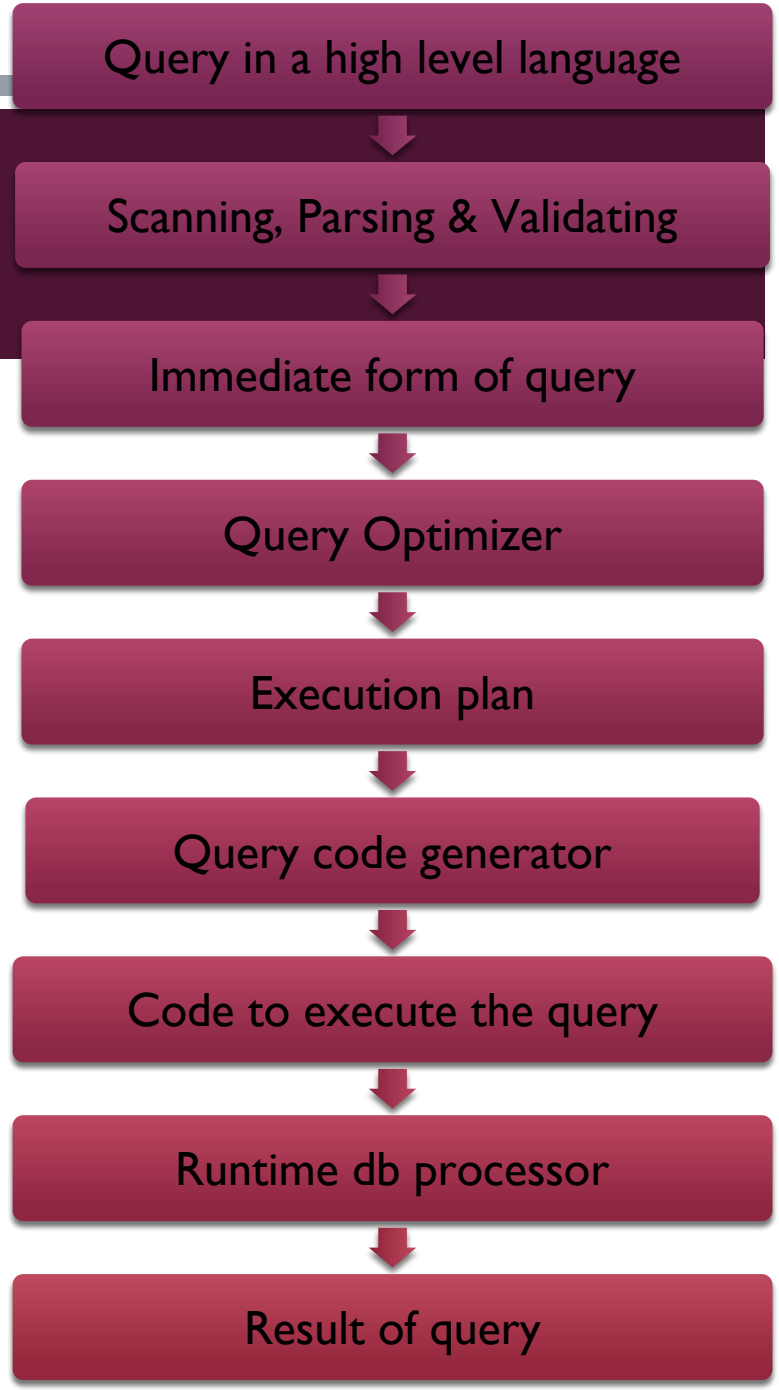
QUERY PROCESSING

- It refers to the range of activities involved in extracting data from a db.
- Activities include
  - translation of queries in high level db languages into expressions that can be used at the physical level of file system,
  - a variety of query optimizing transformations &
  - actual evaluation of queries.
- Steps involved in processing a query:
  1. Parsing & Translation
  2. Optimization
  3. Evaluation

QUERY PROCESSING

- Typical steps shown in the diagram:

- Scanner: identifies language tokens (keywords)
- Parser: Checks query syntax to determine if rules are being followed.
- Validator: Checks all attributes, relation name are valid & exists.
- Optimizer: has task of producing an execution plan.
- Generator: has task of running query whether in compiled or interpreted mode to produce query result.



EXAMPLE

- SELECT Ename FROM Employee WHERE Salary > 5000;

- **Translated into Relational Algebra Expression**

  σ Salary > 5000 (π Ename (Employee))

  OR

  π Ename (σ Salary > 5000 (Employee))

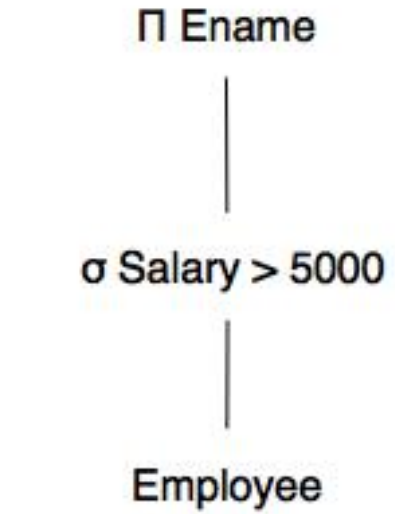


Fig. Query Execution Plan

QUERY PROCESSING : EVALUATION OF EXPRESSIONS

- The obvious way to evaluate an expression is simply to evaluate one operation at a time in an appropriate order.
- The result of each evaluation is materialized (created) in a temporary relation for subsequent use. This is the disadvantage of this approach that a separate temporary relation must be created in storage.
- Alternative approach is to evaluate several operations simultaneously in a pipeline with the results of one operation passed on to the next, without the need to store a temporary relation.
- Based on the query, both the approach seems feasible at one way or the other.

QUERY PROCESSING : EVALUATION OF EXPRESSIONS

Materialization:

- We start from lowest level operations in expression. (ex: Nested query)
- We execute the operations & store the results in temporary relation.
- We can use those temporary relations to execute the operations at next level up in the tree, where inputs are now either temporary relations or relations stored in db.
- By repeating the process, we will eventually evaluate the operation at the root of the tree, giving the final result of the expression.
- This type of evaluation is called Materialized evaluation.

QUERY PROCESSING : EVALUATION OF EXPRESSIONS

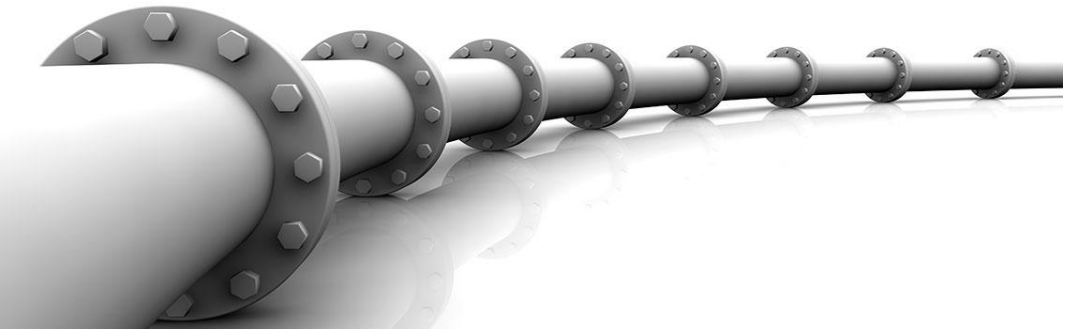
Materialization:

- The cost of a materialized evaluation is not simply the sum of the costs of the operations involved.
- To compute the cost of evaluating an expression, we have to:
  - Add the costs of all the operations
  - As well as the cost of writing the intermediate results to disk (working on disk is comparatively slower)

QUERY PROCESSING : EVALUATION OF EXPRESSIONS

Pipelining: (Alternative to materialization; saving is not done)

- We combine several relational operations into a pipeline of operations in which the results of one operation are passed along to the next operation in the pipeline.
- Such type of evaluation is called Pipelined evaluation.
- Combining operations into a pipeline eliminates the cost of reading & writing temporary relations.



QUERY PROCESSING : EVALUATION OF EXPRESSIONS

Pipelining: (Alternative to materialization; saving is not done)

Implementation:

- By constructing a single, complex operation that combines the operations that constitutes the pipeline.
- It is desirable to reuse the code for individual operations in the construction of a pipeline.
- Thus, each operation in pipeline is modelled as a separate process within the system that
  - takes a stream of tuples from its pipelined inputs &
  - generates a stream of tuples for its output.

QUERY PROCESSING : EVALUATION OF EXPRESSIONS

Pipelining: (Alternative to materialization; saving is not done)

Pipeline can be executed in either of two ways:

1. **Demand driven**: System makes repeated requests for tuples from the operation at the top of the pipeline. Each time that an operation receives a request for tuples, it computes the next tuple to be returned & then returns that tuple.
2. **Producer driven**: Operations do not wait for requests to produce tuples, but instead generate the tuples eagerly. Each operation at the bottom of a pipeline continually generates output tuples and puts them in its output buffer, until buffer is full. An operation at any other level of a pipeline generates output tuples when it gets input tuples from lower down in the pipeline, until its output buffer is full. Once operation uses a tuple from pipelined input, it removes tuple from its buffer.

QUERY PROCESSING : EVALUATION OF EXPRESSIONS

Pipelining: (Alternative to materialization; saving is not done)

- Using Producer-driven pipelining can be thought of as pushing data up an operation tree from below, whereas using Demand-driven pipelining can be thought of as pulling data up an operation tree from top.
- Whereas tuples are generated eagerly in Producer-driven pipelining, they are generated lazily on demand in Demand-driven pipelining.

QUERY PROCESSING : MEASURES OF QUERY COST

- The cost of query evaluation can be measured in terms of a number of different resources, including
  - disk accesses,
  - CPU time to execute a query,
  - the cost of communication in a distributed or parallel db system,
- The response time for a query evaluation plan (clock time required to execute the plan) assuming no other activity is going on in the computer, would account for all these costs & could be used as a good measure of the cost of the plan.
- In large db system, however the cost to access data from disk is usually the most important cost, since disk accesses are slow compared to in-memory operations.
- It is likely that the time spent in disk (mechanical) activity will continue to dominate the total time to execute a query.

QUERY PROCESSING : MEASURES OF QUERY COST

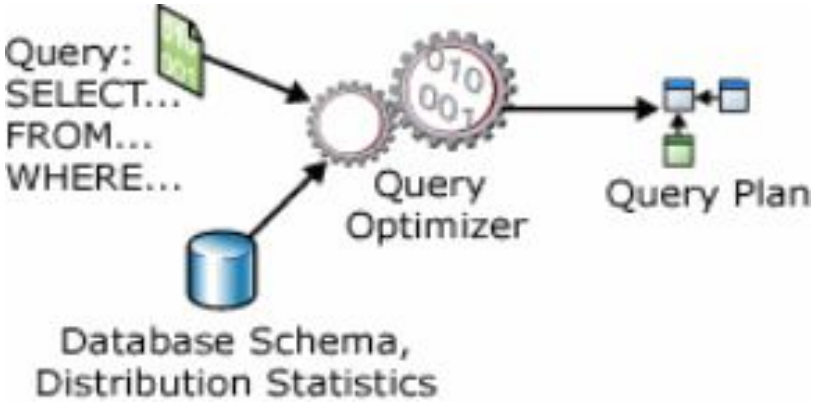
- We use the ‘number of block transfers’ from disk & the ‘number of disk seeks’ to measure the cost of accessing data from disk.
- We can refine our cost estimates further by distinguishing block reads from block writes, since block writes are typically about twice as expensive as reads (read to verify for successful/unsuccessful write).
- The costs of all the algorithms that we consider depend on the size of the buffer in main memory.
- In best case, all data can be read into the buffer & doesn’t need to be accessed again.
- In worst case, we assume that buffer can hold only a few blocks of data, so more access to disk for reading & hence slow.
- We generally assume the worst case, so that final resulting cost will be less than estimated.

QUERY PROCESSING : QUERY OPTIMIZATION

- Given a query, there are generally a variety of methods (m/p) for computing the answer. It is the responsibility of the system to transfer the query as entered by the user into an equivalent query that can be computed more efficiently.
- The process of finding a good strategy for processing a query is called QUERY OPTIMIZATION.
- We do not expect users to write their queries so that they can be processed efficiently.
- One aspect of optimization occurs at the relational algebra level where the system attempts to find an expression that is equivalent to the given expression, but more efficient to execute.

QUERY PROCESSING : QUERY OPTIMIZATION

- Another aspect is selecting a detailed strategy for processing the query, such as choosing the algorithm to use for executing an operation, choosing the specific indexes to use & so on.
- The difference in cost (in terms of evaluation time) between a good strategy & a bad strategy is often important. Hence, it is worthwhile for the system to spend a large amount of time on the selection of a good strategy for processing a query, even if the query is executed only once.



THANK YOU.

HTTPS://WWW.TUTORIALCUP.COM/DBMS/QUERY-PROCESSING.HTM