# Lecture #10: Query Processing

**15-445/645 Database Systems (Fall 2018)**
https://15445.courses.cs.cmu.edu/fall2018/
Carnegie Mellon University
Prof. Andy Pavlo

## 1  Query Plan

The DBMS converts a SQL statement into a query plan. Operators are arranged in a tree. Data flows from the leaves towards the root. The output of the root node in the tree is the result of the query. Typically operators are binary (1–2 children). The same query plan can be executed in multiple ways. Most DBMSs will want to use an index scan as much as possible.

## 2  Processing Models

A DBMS **processing model** defines how the system executes a query plan. There are different models that have various trade-offs for different workloads.

### Iterator Model

This is the most common processing model and is used by almost every DBMS.

- Top-Down Processing
- Every query plan operator implements a `next` function:
    - On each call to `next`, the operator returns either a single tuple or a null marker if there are no more tuples.
    - The operator implements a loop that calls `next` on its children to retrieve their tuples and then process them (i.e., calling `next` on a parent calls `next` on their children).
- Allows for *pipelining* where the DBMS can process a tuple through as many operators as possible before having to retrieve the next tuple.
- Some operators will block until children emit all of their tuples (joins, subqueries, order by).
- Output control works easily with this approach (`LIMIT`).

### Materialization Model

Each operator processes its input all at once and then emits its output all at once. The operator "materializes" its output as a single result.

- Bottom-Up processing.
- This approach is better for OLTP workloads because queries typically only access a small number of tuples at a time. Thus, there are fewer function calls to retrieve tuples.
- Not good for OLAP queries with large intermediate results.

### Vectorization Model

Like the iterator model where each operator implements a `next` function. But each operator emits a *batch* (i.e., vector) of data instead of a single tuple.

- Top-Down Processing
- Ideal for OLAP queries that have to scan a large number of tuples because there are fewer invocations of the next function.

## 3    Access Methods

An access method is the how the DBMS accesses the data stored in a table. These will be the bottom operators in a query plan that "feed" data into the operators above it in the tree. There is no corresponding operator in relational algebra.

**Sequential Scan**

For each page in table, iterate over each page and retrieve it from the buffer pool. For each page, iterate over all the tuples and evaluate the predicate to decide whether to include tuple or not.

Optimizations:

- **Prefetching:** Fetches next few pages in advance so that the DBMS does not have to block when accessing each page.
- **Parallelization:** Execute the scan using multiple threads/processes in parallel.
- **Buffer Pool Bypass:** The scan operator stores pages that it fetches from disk in its local memory instead of the buffer pool. This avoids the sequential flooding problem.
- **Zone Map:** Pre-compute aggregations for each tuple attribute in a page. The DBMS can then check whether it needs to access a page by checking its Zone Map first. The Zone Maps for each page are stored in separate pages and there are typically multiple entries in each Zone Map page. Thus, it is possible to reduce the total number of pages examined in a sequential scan.
- **Late Materialization:** Each operator passes the minimal amount of information needed to by the next operator (e.g., record id). This is only useful in column-store systems (i.e., DSM).
- **Heap Clustering:** Tuples are stored in the heap pages using an order specified by a clustering index.

**Index Scan**

The DBMS picks an index (or indexes) to find the tuples that the query needs.

When using multiple indexes, the DBMS executes the search on each index and generates the set of matching record ids. One can implement this record id using bitmaps, hash tables, or Bloom filters. The DBMS combines these sets based on the query's predicates (union vs. intersect). It then retrieve the records and apply any remaining terms. This is called a *Bitmap Scan* in Postgres.

Retrieving tuples in the order that they appear in an unclustered index is inefficient. The DBMS can first figure out all the tuples that it needs and then sort them based on their page id.

## 4    Expression Evaluation

The DBMS represents a query plan as a tree. Inside of the operators will be an expression tree. For example, the WHERE clause for a filter operator.

The nodes in the tree represent different expression types:

- Comparisons (=, <, >, !=)
- Conjunction (AND), Disjunction (OR)
- Arithmetic Operators (+, -, *, /, %)
- Constant and Parameter Values

- Tuple Attribute References

To evaluate an expression tree at runtime, the DBMS maintains a context handle that contains metadata for the execution, such as the current tuple, the parameters, and the table schema. The DBMS then walks the tree to evaluate its operators and produce a result.