Introduction to Query Execution and Query Processing

Query Processing in a nutshell

Query processing in DBMS starts with reading and parsing the user query and ends with providing the results back to the user. The query processor consists of two layers of functionality:

- **Query Compiler**, *parses* the incoming query, constructs *the logical query plan*, and selected *the physical query plan*.
- **Query Execution Module** provides implementations of atomic operations used in the physical query plan.

The operations implemented in the query execution module come from *relational algebra*. They are:

- Selection (σ);
- Projection (π);
- Cartesian Product ($\times$);
- Join(s) ($\bowtie$);
- Union, Intersection, Difference ($\cup, \cap, -$);
- Duplicate elimination (δ);
- Grouping and Aggregation (γ);
- Sorting (τ).

Other operations, not reflected in query algebra include

- table scan;

- index scan;
- sort-scan;
- iterators;

Each operation can have multiple implementations in the DBMS.

Logical query plans differ from each other in the order in which operations are executed.

Physical query plans differ from each other in the choice of specific implementations of operations.

The following affects the implementations of relational algebra operations:

- Basic strategy:
  - scanning;
  - hashing;
  - sorting;
  - indexing.
- amount of available memory;
- existence of indexes;

Cost of an operation

The **cost** of an operation is *the number of* ***disk I/Os*** that takes place during its execution.

Assumption: the arguments of each operation are found on disk, the results stay in main memory.

Measuring Costs

M: number of main-memory buffers available to the DBMS (buffer size).

B: given a relation R, $B(R)$ is the number of blocks that store records from R on disk.

T: given a relation R, $T(R)$ is the number of tuples in R.

V: given a relation R and its attribute A, $V(R, A)$ is the number of distinct values of attribute A currently stored in R.

Scans

A scan is the most basic operation in DBMS. It also serves as a good example of standard approaches to query execution.

A scan is the operation of *reading the entire contents of a relation.* The following types of scans are typically found in the DBMS:

- table-scan. The relation R is stored in a single data file in secondary memory. The blocks containing the tuples of R are *retrieved one-by-one*.

  Cost: $B(R)$.

- index-scan. The relation R is stored in a single data file, or as part of a data file. An index I_R exists for R. The index-scan is performed, but first scanning the index I_R, and then retrieving the blocks containing the tuples of R.

  Cost: $B(I_R) + B(R)$

  Note: an index-scan is less efficient than a table-scan but, it is more flexible, as it may allow for faster retrieval of *portions* of the relation (and a scan is going to be an integral part of a selection operation, as well as of such operations as union, intersection and difference).

- sort-scan. This operation scans a given relation and sorts it *on the fly* based on values of a specified key. sort-scan can be implemented in a number of ways:

  1. **B+-tree.** If a B+-tree index I_R on the given key exists, sort-scan can be converted into an index-scan.
     Cost: $B(I_R) + B(R)$.
  2. **Memory sort.** If R is small enough to fit main memory, perform a table-scan and then use main-memory sorting algorithm (e.g. quicksort, adapted for blocks) to sort it.
     Cost: $B(R)$.
  3. **External sort.** If R is large, external sort methods, such as *merge-sort* must be used.
     Cost: $3B(R)$ for a *multiway merge-sort*[1].

Iterators

An Iterator is a group of three functions, which *allows the passage of information one tuple at a time.* The three operations are:

- Open. Starts the process of getting tuples but does not retrieve tuples.
- GetNext. Retrieves next tuple from the relation/input stream. Returns a special "message" if end of tuple stream is reached.
- Close. Finishes the iterator.

[1]Assuming that $B(R)/M < M$.