3.5 Optimization of Distributed Queries

A calculus expressed on global relations can be mapped into a query on relation fragments by decomposition and data localization. This mapping uses the global and fragment schemas. During this process, the application of transformation rules permits the simplification of the query by eliminating common subexpressions and useless expressions. This type of optimization is independent of fragment characteristics such as cardinalities. The query resulting from decomposition and localization can be executed in that form simply by adding communication primitives in a systematic way. However, the permutation of the ordering of operations within the query can provide many equivalent strategies to execute it. Finding an optimal ordering operation for a given query is the main role of the query optimization layer, or optimizer in short.

The selection of the optimal strategy generally requires the prediction of execution costs of the alternative candidate orderings prior to actually executing the query. The execution cost is expressed as weighted combination of I/O, CPU, and communication cost. A typical simplification of the earlier distributed query optimizers was to ignore local processing cost (I/O and CPU costs) by assuming that the communication cost is dominant.

Query optimization refers to the process of producing a query execution plan (QEP) which represents an execution strategy for the query. This QEP minimizes an objective cost function. A query optimizer, the software module that performs query optimization, is usually seen as consisting of the three components: a search space, a cost model, and a search strategy (see Figure 8.1).

3.5.1 Search Space

The search space is the set of alternative execution plans that represent the input query. These plans are equivalent, in the sense that they yield the same result, but they differ in the execution order of operations and the way these operations are implemented, and therefore in their performance. The search space is obtained by applying transformation rules, such as those for relational algebra described previously.

Query execution plans are typically abstracted by means of operator trees, which define the order in which the operations are executed. They are enriched with additional information, such as the best algorithm chosen for each operation. For a given query, the search space can thus be defined as the set of equivalent operator trees that can be produced using transformation rules. To characterize query optimizers, it is useful to concentrate on join trees, which are operator trees whose operators are join or Cartesian product. This is because permutations of the join order have the most important effect on performance of relational queries.

3.5.2 Search Strategy

The most popular search strategy used by query optimizers is dynamic programming which is deterministic. Deterministic strategies proceed by building

plans, starting from base relations, joining one more relation at each step until complete plans are obtained. Dynamic programming builds all possible plans, breadth-first, before it choose the "best" plan. To reduce optimization cost, partial plans that are not likely to lead to the optimal plan are pruned (i.e. discarded) as soon as possible. By contrast, another deterministic strategy, the greedy algorithm, builds only one plan depth-first.

Dynamic programming is almost exhaustive and assures that the "best" of all plans is found. It incurs an acceptable optimization cost (in terms of time and space) when the number of relations in the query is small. However, this approach becomes too expensive when the number of relations is greater than 5 or 6. For more complex queries, randomized strategies have been proposed, which reduce the optimization complexity but do not guarantee the best of all plans. Unlike deterministic strategies, randomized strategies allow optimizer to trade optimization time for execution time.

3.5.3 Distributed Cost Model

An optimizer's cost model includes cost functions to predict the cost of operators, statistics and base data, and formulas to evaluate the sizes of intermediate results. The cost in terms of execution time, so a cost function that represents the execution time of a query. The cost of a distributed execution strategy can be expressed with respect to either the total time or the response time. The total time is the sum of all time (also referred to as cost) components, while the response time is the elapsed time from the initiation to the completion query.