Query optimization is the process of selecting the most efficient query-evaluation plan from among the many strategies usually possible for processing a given query, especially if the query is complex. We do not expect users to write their queries so that they can be processed efficiently. Rather, we expect the system to construct a query-evaluation plan that minimizes the cost of query evaluation. This is where query optimization comes into play.

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. 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 indices to use, and so on.

The difference in cost (in terms of evaluation time) between a good strategy and a bad strategy is often substantial and may be several orders of magnitude. Hence, it is worthwhile for the system to spend a substantial amount of time on the selection of a good strategy for processing a query, even if the query is executed only once.

Bibliographical Notes

The seminal work of [Selinger et al. (1979)] describes access-path selection in the System R optimizer, which was one of the earliest relational-query optimizers. Query processing in Starburst, described in [Haas et al. (1989)], forms the basis for query optimization in IBM DB2.

[Graefe and McKenna (1993)] describe Volcano, an equivalence-rule–based query optimizer that, along with its successor Cascades ([Graefe (1995)]), forms the basis of query optimization in Microsoft SQL Server.

Estimation of statistics of query results, such as result size, is addressed by [Ioannidis and Poosala (1995)], [Poosala et al. (1996)], and [Ganguly et al. (1996)], among others. Nonuniform distributions of values cause problems for estimation of query size and cost. Cost-estimation techniques that use histograms of value distributions have been proposed to tackle the problem. [Ioannidis and Christodoulakis (1993)], [Ioan-
nidis and Poosala (1995)], and [Poosala et al. (1996)] present results in this area. The use of random sampling for constructing histograms is well known in statistics, but issues in histogram construction in the context of databases is discussed in [Chaudhuri et al. (1998)].


[Blakeley et al. (1986)] describe techniques for maintenance of materialized views. Optimization of materialized view maintenance plans is described by [Vista (1998)] and [Mistry et al. (2001)]. Query optimization in the presence of materialized views is addressed by [Chaudhuri et al. (1995)]. Index selection and materialized view selection are addressed by [Ross et al. (1996)], and [Chaudhuri and Narasayya (1997)].

Optimization of top-K queries is addressed in [Carey and Kossmann (1998)] and [Bruno et al. (2002)]. A collection of techniques for join minimization has been grouped under the name tableau optimization. The notion of a tableau was introduced by [Aho et al. (1979b)] and [Aho et al. (1979a)], and was further extended by [Sagiv and Yannakakis (1981)].

Parametric query-optimization algorithms have been proposed by [Ioannidis et al. (1992)], [Ganguly (1998)] and [Hulgeri and Sudarshan (2003)]. [Sellis (1988)] was an early work on multiquery optimization, while [Roy et al. (2000)] showed how to integrate multi-query optimization into a Volcano-based query optimizer.

[Galindo-Legaria et al. (2004)] describes query processing and optimization for database updates, including optimization of index maintenance, materialized view maintenance plans, and integrity constraint checking, along with techniques to handle the Halloween problem.

Bibliography


Chapter 16 Query Optimization


Credits

The photo of the sailboats in the beginning of the chapter is due to ©Pavel Nesvadba/Shutterstock.