Technical Perspective: Broadening and Deepening Query Optimization Yet Still Making Progress

Jeffrey F. Naughton
University of Wisconsin–Madison
Madison WI, U.S.A.

Query optimization is a fundamental problem in data management. Simply put, most database query languages are declarative rather than imperative – that is, they specify properties that the answer should satisfy, rather than give an algorithm to compute the answer. The best known and most widely used database query language, SQL, is a prime example of a language for which optimization is essential.

By "essential" I mean that database optimization is not a matter of shaving 10% or even a factor of 2X from a query's execution time. In database query evaluation, the difference between a good plan and a bad or even average plan can be multiple orders of magnitude – so successful query optimization makes the difference between a plan that runs quickly and one that never finishes at all. Accordingly, since the seminal papers in the 1970s, query optimization has received and continues to receive a great deal of attention from both the industrial and research database communities.

Early work on optimization focused on a scenario in which the query was fully specified, and the optimization goal was query evaluation time. That is, the problem was this: what is the fastest way to evaluate this query? While this problem was (and is!) challenging, it is not broad enough to capture the optimization problem faced by modern systems. As an important example, many times the query is not fully specified in advance (as a simple example, it may contain variables, or "parameters" that are only discovered at run time). This generalization gives rise to *parametric query optimization*, where the problem is as follows:

> Given a partially specified query, find a set of good evaluation plans, one of which will be chosen at run time when the parameter is instantiated.

Yet another necessary generalization has to do with the optimization goal. Sometimes execution time is not the only criterion by which plans should be selected. As a prominent and current example, if the query is being run in the cloud, the system may of course want to find fast evaluation plans, but may also desire inexpensive ones. That is, now we have two objectives: running time and cost. This gives rise to *multi-objective query optimization*, where the problem is as follows:

> Given a query and a set of objectives, find a set of plans that are Pareto-optimal for these objectives (a plan is "Pareto-optimal" if it is not dominated in all objectives by other plans.)

Both parametric and multi-objective query optimization have been studied in the past, but the following paper, by Trummer and Koch, is a remarkable tour de force exploration of the combination of the two. Here, the problem is roughly the following. Given a partially specified query, and multiple objectives for the resulting plan, find a set of Pareto-optimal plans that can be chosen at run time by filling in all parameters.

Since the original query optimization problem and its variants are already very difficult, one might despair that simultaneously treating two substantial extensions would yield a hopelessly intractable problem. Thus the current paper is surprising in its elegance and effectiveness. The paper embeds the problem in an insightful and expressive formal framework, and specifies a solution that combines aspects of piecewise linear functions, dynamic programming with pruning based upon Pareto polytope analyses, and linear programming. A thorough set of experiments with an implementation of their algorithm completes the paper, and indicates that all of this actually works.