Technical Perspective: Accurate Summary-based Cardinality Estimation Through the Lens of Cardinality Estimation Graphs

Dan Suciu
Universtity of Washington
suciu@cs.washington.edu

Query engines are really good at choosing an efficient query plan. Users don't need to worry about how they write their query, since the optimizer makes all the right choices for executing the query, while taking into account all aspects of data, such as its size, the characteristics of the storage device, the distribution pattern, the availability of indexes, and so on. The query optimizer always makes the best choice, no matter how complex the query is, or how contrived it was written. Or, this is what we expect today from a modern query optimizer. Unfortunately, reality is not as nice.

The problem is that query optimizers have an Achilles' heel: cardinality estimation. In order to choose an optimal query execution plan, they need to search over many alternative plans and evaluate their cost, and in order to evaluate their cost the optimizer needs to estimate the number of tuples that will result for each subexpression of the query. If they estimate incorrectly the number of tuples, then their cost estimate is wrong, and they end up choosing an inefficient query plan. When query optimizers fail to choose the best execution plan, it is almost always because they were informed incorrectly by the cardinality estimator.

The cardinality estimation problem is easy to state. Given a query, estimate the cardinality of its answer *without* evaluating the query. Typically, the query is restricted to selection and join operators, since other relational operators (group by, duplicate elimination, union, etc) are usually pushed to the top of the query plan. In order to perform cardinality estimation, the system computes offline some statistics on each base table, then uses them at optimization time to estimate the cardinality of various query expressions. There are two major challenges. First, there is a very limited space budget for what statistics we can precompute and store, because all statistics must be kept in main memory during query optimization. Production databases often have hundreds of tables, some with hundreds of attributes, hence we can afford to store only very little information about each table and each attribute. Second, cardinality estimation must be very fast, because it is in the inner loop of the query optimizer, which needs to invoke the estimator thousands of times during optimization. Disk accesses, or solving complex optimization problems are ruled out: cardinality estimation must be simple and efficient.

The following paper (*Accurate Summary-based Cardinality Estimation*) by Chen et al. presents a unified framework that captures the most common approaches to cardinality estimation: those that are based on *summary statistics* (as opposed to samples) and those that are based on *average or maximum degree* (which includes all estimators that I'm aware of). There are two separate tasks in cardinality estimation, which are often studied separately: selectivity estimation of predicates on the base tables, and the cardinality estimation of a multi-join query. The paper studies exclusively the second task.

A cardinality estimate of a multi-join query relies on average degrees. Consider estimating the size of a join, $R(A,B) \bowtie S(B,C)$. A reasonable estimate is the cardinality of R times the average degree of the attribute $S.B$; for example, if each value of B occurs 5 times in S, then our estimate is 5 times $|R|$. But we can also switch the roles of R and S and use as estimate $|S|$ times the average degree of $R.B$. This is a problem, which estimate should we use? Database systems choose the smallest of these two, and this leads to the classic textbook formula, that we like to teach in undergraduate classes: $|R \bowtie S| = |R| \cdot |S| / \max(\text{dom}(R.B), \text{dom}(S.B))$. It becomes clear that, for an n-way join query, the number of choices becomes very large, and it is unclear which one to use as estimate. The paper introduces a simple abstraction, called the *cardinality estimation graph*, or CEG, where each path represents one possible choice for the cardinality estimator. Using a CEG, the paper can explain and compare the various cardinality estimators described in the literature, opening up the possibility for designing new estimators.

A recently proposed alternative to computing a cardinality *estimate* is to compute an *upper bound*. This approach is called a *pessimistic cardinality estimate* (with some abuse, since it is not an estimate, but a guaranteed upper bound). The definitive mathematical formula for the upper bound of a multi-join query is known, but impractical, and the few implementations of pessimistic estimates use relaxations of this formula. The paper shows that the same CEG abstraction can also be used to explain these pessimistic cardinality estimates, by simply replacing the average degree with a maximum degree.

Readers with little knowledge of cardinality estimation will hopefully find the CEGs a convenient, quick introduction to various cardinality estimation methods. Experts will find the CEG a convenient abstraction for comparing various estimators. And systems builders may find the experimental section a useful guide in making choices for their own cardinality estimator.