Research Highlights: Bridging Theory and Practice with Query Log Analysis

Leonid Libkin
School of Informatics, University of Edinburgh
libkin@inf.ed.ac.uk

Take a database conference paper and search for *"in the real world"* in it; chances are high you will find it. Of course what is real depends on one's perspective: for a pure theory paper it could be what one saw in a systems paper, for a systems paper it could be an issue that implementors of DBMSs had to deal with, and for the latter it may be what the customers need. But to sharpen our research tools, it helps tremendously to understand that the real "real world" is, and adjust our (sometimes very elaborate) techniques to address problems that actually occur.

A nice example of this is analyzing the computational complexity of database queries. Database theory has developed an arsenal of tools for this. We know that for many classes of queries, the complexity is roughly $\|D\|^{O(\|Q\|)}$ for a database D and a query Q, where $\|\ \|$ means size. Thus, much research went into the detailed analysis of the structure of queries that removes $\|Q\|$ from the exponent and replaces it by a small fixed constant. We have a very good theoretical understanding of such classes, but we know much less about their relationship with queries that real-world users write.

Sometimes the situation is even more dramatic and the complexity of best known algorithms is exponential, e.g., $c^{\|D\|}$ for a constant c. This looks like an non-starter, and theoretical research tends to dismiss queries of such a complexity. But it shouldn't. To start with, we have many examples of problems working well in the real "real world" and yet having bad theoretical behavior. For example, satisfiability is the canonical NP-complete problem, and yet SAT solvers do very well these days. There are programming languages whose type inference algorithms require exponential time. And closer to databases, the current leader among graph databases, Neo4j, uses an NP-hard query evaluation algorithm and yet it works well, as is evidenced by their position in the graph database market.

The reason all these work well in practice is that the complexity analysis considers the worst case, and the worst is not necessarily what occurs in life. Yes, one can choke a SAT solver, one can write a simple program whose type will fill pages, and one can write a graph database query that will take forever even on a relatively small graph, but these are not the typical cases. The question then arises: what are the typical cases that one deals with in that real world?

This is the question that the paper by Wim Martens and Tina Trautner answers for path queries over graphs. To understand what the real world looks like, one has to observe it, rather than just make assumptions about it. This is what they do, by analyzing very large repositories of available SPARQL queries, and seeing what types of path queries they use. It is important to look at large query logs, and also at different ones, as different logs may well have very different characteristics and one shouldn't be making assumptions about the entire world out there based on a partial view of it, even if this view has many data points (this point is probably relevant even beyond analyzing query logs...).

The paper does not stop at analyzing query logs. It takes the analysis further to answer the following question: why some graph database queries, while behaving so badly in theory, actually do well in practice? The type of restrictions in path queries on graphs one deals with most often is in the mode in which paths are traversed: either there are no repeated nodes in paths (i.e., we have simple paths), or there are no repeated edges (such paths are called trails; this is the approach that Neo4j takes). Both have long been known to be NP-hard, in the worst case. From practice, we know that the trail semantics of Neo4j works well. Why?

The paper answers this question. By analyzing queries found in the logs, it establishes conditions that simultaneously cover a vast majority of real life queries, and at the same time admit efficient evaluation algorithms. Thus, the worst case might occur, but it is not that common, and this tells us that the high theoretical complexity is not seen in the real world.

This is a very nice *combination* of applied and theoretical research. One does not stop at simply analyzing the logs; instead the authors turn the result of the analysis into a nontrivial theoretical result that says when query evaluation is efficient.

I very much hope we shall see more papers of this kind. Very often the gap between theory and systems is too large in our community: theoreticians produce results motivated by their theoretical value, and systems research often finds such results too far fetched and goes ahead disregarding theoretical developments, hoping for the best essentially (Neo4j's NP-hard algorithm is an example: their approach would have been dismissed by theoreticians, but they went ahead, and it worked). To close the gap, we need to establish this back-and-forth between theory and systems, and the paper you are about to read is one of relatively few but very prominent examples of it. I hope more will follow.