Technical Perspective: Query Answers – Fewer is Faster

Leonid Libkin
The University of Edinburgh & RelationalAI
Edinburgh & Paris
l@libk.in

We often write queries using `LIMIT` k, indicating that only k answers are to be returned. This feature is present in most query languages, for different data models: SQL, SPARQL, Cypher etc. For example, in a repository of about 250M SPARQL queries, about 15M queries are of this form. Not surprisingly of course, the database research community studied such queries extensively. The dominant setting is this: there is an ordering on tuples that can be returned by a query. Then the answer is limited to the first k tuples in this ordering.

But how realistic is the ordering assumption? And even if an ordering on tuples can be defined somehow, how important is it for query answering?

To use an ordering explicitly to select the top k answers, one would combine the `LIMIT` clause with an `ORDER BY` clause. But is this combination really the prevalent one in real-life queries? It turns that there are important cases (again, based on a detailed analysis of the above mentioned SPARQL query log) in which the answer is negative. In fact, fewer than 0.1% of queries that use `LIMIT` also use `ORDER BY`. Furthermore, in queries that use `LIMIT` k, the number of returned tuples is usually small: k is typically under 100.

There is therefore a tantalizing question: if we do not really care about the order in which the first k answers to a query are returned, can this be used to our advantage? In particular, can this be used to evaluate queries faster? This is precisely the question that the VLDB 2022 paper *"Threshold queries in theory and in the wild"* set to answer.

At first, it explained a variety of scenarios where such queries appear. These include data exploration (e.g., show me some random 10 tuples from the query answer), cardinality constraint queries (find violations of cardinality restrictions, e.g., limits on sizes of groups), or data monitoring queries (say, connecting a group to a number of related features). Having provided examples of those queries written in SQL, SPARQL, and Cypher, the paper then went ahead with a combination of theoretical and experimental studies.

Theoretical study.

The key contribution here is a clean theoretical model that encompasses the above types of queries and is at the same time amenable to complexity-theoretic and algorithmic analyses. The essence of the theoretical model is a combination of a usual conjunctive (or join) query with a device from logic known as a counting quantifier. Specifically, the paper looked at queries of the form $q(\bar{x}) \wedge \exists^{k,m} \bar{y}\ q'(\bar{x}, \bar{y})$, which find tuples $\bar{x}$ in the answer to q so that the number of tuples $\bar{y}$ such that $(\bar{x}, \bar{y})$ is in the answer is between k and m. Here $m = \infty$ is an option, in which case the number of answers must be at least k.

With this abstraction, one next translates queries into an extended version of relational algebra, that has two additional operators. One mimics the counting quantifier, and the other allows a nondeterministic selection of tuples from a relation, subject to certain conditions. At this point a clean theoretical framework is established, and one can provide a detailed algorithmic analysis and show how to evaluate queries in a way that beats the naive way: just compute the join and then select k rows.

Experimental study.

A theoretical algorithm, no matter how good on paper, would not be very useful without a strategy to implement it in a DBMS. And the paper found such a strategy, using SQL window functions, which are a generalization of the standard grouping and aggregation in SQL. They showed how to encode their queries using window functions and then tested them on a real data set (the `imdb.com` database) and syntetic graphs designed to model social networks. Their queries are essentially path queries where the counting threshold is used to limit the number of n-hop neighbors in a graph. They compared these windowed version, inspired by the theoretical analysis, with the standard evaluation using `LIMIT`, and showed a significant order-of-magnitude improvement.

Why you should read this paper.

One can think of many reasons. First, it addresses a very real problem that applies to millions of queries found in various query logs. We all write queries of that kind, teach students how to write such queries, etc. Second, it finds a very real and very large gap in our understanding of how common queries run, simply by noticing that `LIMIT` and `ORDER BY` do not usually go hand-in-hand. And third, it provides a beautiful combination of theory and practice in one piece of work, by finding a mathematical model, analyzing it, and showing that its theoretical properties can be supported by an implementation in a RDBMS.