Technical Perspective: Supporting Linear Algebra Operations in SQL

Yannis Papakonstantinou
Computer Science and Engineering Department
University of California, San Diego
yannis@cs.ucsd.edu

Linear algebra operations are at the core of Machine Learning. Multiple specialized systems have emerged for the scalable, distributed execution of matrix and vector operations. The relationship of such computations to data management and databases however brings frictions. It is well known that a great deal of human time and machine time is being spent nowadays on fetching data out of the database and performing a computation on a specialized system. One answer to the issue is that we truly need a new kind of non-SQL database that is tuned to these computations.

The creators of SimSQL opted for the decidedly incremental approach. Can we make a very small set of changes to the relational model and RDBMS software to render them suitable for executing linear algebra in the database?

We have come across the "brand new system" versus "incremental to relational" question many times in the database field. E.g., do we need brand new query languages and query processors for data cubes? Or do we need to have our query processors pay attention to specific cases that are especially common in data analytics queries over stars and snowflakes? Do semistructured query languages need to depart from SQL or it is enough to be incremental to SQL? Same for query processors. Repeat the questions to graph data and RDF data. In many cases, new custom systems emerged only to figure out later that we could/should have tackled the problem incrementally. That's the trap that the authors of this paper avoid.

This is not to say that radical changes and extensions should be forbidden. Rather it says that we should closely scrutinize the necessity of the changes, do them when needed and keep them minimal. The authors identify the right opportunities. Here is a non-exhaustive list:

(a) Writing matrix and vector operations as a join over the index can be syntactically tedious. They solve the problem by introducing special syntactic features.

(b) They notice a connection between signatures and size estimation and exploit it.

(c) They allow their query user to move across different denormalizations to find the one that makes sense from expressiveness and performance point of view. The point where types relate to performance is whether the right level of granularity for distribution in a shared-nothing architecture is specified.

Overall, the extensions of the paper follow a thoughtful and minimal approach that is worth studying in the particular field of linear algebra operations, as well as generally in the design of systems for analytics.