Reminiscences on Influential Papers

Kenneth A. Ross, editor

I continue to invite unsolicited contributions. See `http://www.acm.org/sigmod/record/author.html` for submission guidelines.

Zach Ives, University of Pennsylvania, `zives@cis.upenn.edu`.

[Navin Kabra and David DeWitt. Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans. ACM SIGMOD 1998, pages 106–117.]

As a young PhD student looking for a research area, this is one of the first papers that I read in the database field. It hooked me on a topic I had previously imagined was solved — query processing — which actually had a wealth of open research questions. Traditional System-R-style query optimization is based on a simplified model of query execution; it fares quite poorly under volatile conditions or when given limited statistics. In a world that is increasingly difficult to model in an optimizer — due to missing statistics, remote data sources, XML and object-relational data, and user-defined functions — there is a very interesting question of how to do better.

Kabra and DeWitt focus on an important case of this problem, improving the performance of relational or object-relational database engines that have incomplete statistics. Prior work had tried to handle this issue by precompiling "contingent" plans that would be employed under different runtime conditions. This paper (and Urhan and Franklin's query scrambling work, which was published concurrently) was among the first to actually choose new query plans in the midst of execution: the DBMS could re-invoke the optimizer if warranted by real-world execution conditions. While this initially seemed like it might be expensive, the paper presented strong evidence that runtime re-optimization could be done in a way that paid off. That revelation greatly influenced my PhD thesis work in an area with similar optimization challenges, query processing for data integration. The work of this paper also helped define the new subfield of adaptive query processing — a topic that continues to intrigue me.

Bertram Ludäscher, San Diego Supercomputer Center, U.C. San Diego, `ludaesch@sdsc.edu`.

[Allen Van Gelder. The Alternating Fixpoint of Logic Programs with Negation. ACM Symposium on Principles of Database Systems (PODS), pages 1–10, 1989.]

The well-founded semantics (WFS) of logic programs with negation by Van Gelder, Ross, and Schlipf maps "non-controversial" logic atoms to either *true* or *false*, leaving *undefined* those atoms whose truth cannot be determined using only a "well-founded argumentation". WFS became widely accepted as the declarative semantics of logic programs involving cyclic dependencies with negation. However, it was the simple and elegant procedural computation in the form of an alternating sequence of *increasing underestimates* and *decreasing overestimates* that shaped my thinking about the WFS and its applicability for resolving "disputes" between two opposing rational players: Each player can argue and attack the opponent's viewpoint by invoking the "rules of the game" as encoded in the logic program. (Contrast this with some televised debates in which opponents try to attack each other while avoiding logic and well-founded arguments ;-)

The understanding of WFS as alternating rounds of "best moves" in a logic argumentation game led me to think of other arguably ambiguous problems as games encoded in logic. For example, conflicts between database triggers to maintain referential integrity after updates can be handled in this way (joint work with W. May, PODS'97, EDBT'98, TODS'02). In another line of work, using a result from M. Kubierschky, we could answer an open problem on the WFS from Abiteboul, Hull, and Vianu's textbook, i.e., that all FO+LFP queries can be expressed as an argumentation game with a single recursive rule $win(\bar{x}) \leftarrow move(\bar{x}, \bar{y}), \neg win(\bar{y})$ in which cyclic arguments (drawn positions in the game) are detected as such and "thrown out", resulting in a model with no undefined atoms (joint work with J. Flum and M. Kubierschky, ICDT'97, TCS'00).

Van Gelder's alternating fixpoint computation provides a simple algorithm and intuitive understanding of the well-founded semantics. The latter is a useful tool when it comes to capturing the semantics of possibly conflicting rules as argumentation games. Current trends such as the Semantic Web and deep, semantic modeling of scientific data might trigger a renaissance of logic and rules – let the games begin!

Ioana Manolescu, INRIA Futurs, France, `ioana.manolescu@inria.fr`.

[Goetz Graefe, William J. McKenna. The Volcano Optimizer Generator: Extensibility and Efficient Search. ICDE 1993: 209-218.]

Throughout my research activity so far, I was always involved in some work related to query (or, more generally, data-intensive computation) optimization. I find this to be a fascinating field, albeit one could argue it is immature: few well-established, general models and abstractions are agreed upon. The field is also (wrongly, I think) disdained by some as low-level engineering.

If I had to name one work that breaks this discouraging, dreary image, it would be the ICDE 1993 paper by Graefe and McKenna. This is the most general, and genuinely beautiful, query optimization framework I was able to find.

I found that work relatively late (in 2002, after finishing my PhD). Understanding it enabled me to organize the disparate things that I had learned while building a query optimizer for the LeSelect data integration system. Many optimization algorithms and frameworks I had known (in particular, those of the Disco, Garlic, and Starbust projects) could be seen as instances of the Volcano general framework. In our work, our minds must simultaneously hold a large amount of abstract knowledge. I had learned in a pedagogy class that for a new knowledge item to be retained in our brains, it needs to "stick" to what the brain already holds: it has to be reachable from the root of the mental tree in which we organize, and make sense of, our knowledge. I believe a first great contribution of the Volcano paper lies in its framework, very general, and yet easy to understand and to explain.

The Volcano framework, furthermore, enables one to devise new optimizers for new problems. In the field of XML query processing, for instance, optimization is currently a remote goal, not yet a "field". The XML query language is complex, thus many optimizers assume away various language features. XQuery optimization also raises (stringent) questions that have not been fully, completely addressed even for SQL up to very recent times, such as the efficient handling of ordering and grouping in intermediary query plans; thus, it is not easy to choose an optimization model, since it is not clear how all the nuts and bolts will turn out. When designing an optimizer for our recent XQueC native XML prototype, I relied again on the principles of this work.
