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.

James Hamilton, Microsoft, `jamesrh@microsoft.com`.

[C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwartz. ARIES: A transaction Recovery Method Supporting Fine-Grained Locking with Partial Rollbacks Using Write-Ahead logging. ACM TODS 17(1), pages 94–162, 1992.]

Rarely, a paper comes along that summarizes the current literature and brings it all together in a single place, concisely written, and clearly argued and these papers form the basis point for both future product development and further research. The ARIES paper featured here achieved that goal with unusual clarity yet goes a step further by not just summarizing the current research but in also documenting in detail how leading commercial relational databases had implemented logging, concurrency control, buffer management, and logging at a time when there was almost no information available describing the internals of these systems. The paper is massive in scope covering logging and recovery methods, providing a taxonomy of different approaches to buffer management, concurrency control and lock hierarchies with intent locking, fined grained (row) locking techniques and the impact on recovery, write ahead logging, and recovery techniques include the value of an log analysis phase and some of the different optimizations possible. Some of the concepts discussed where brought together from the existing literature, some of the techniques where from commercial systems where the implementation hadn't been publicaly discussed prior to the publication of this work, and some of approaches were inventions of the authors first presented in this paper.

Subsequent to having read this paper, I was lucky enough to work with the authors on various commercial database projects and I've learned a great deal from them over the years. More than a decade later, I still find this paper on the desks of database implementers, scalable mail system builders, and transaction manager developers throughout the industry. It's amazing how relevant that paper has remained.

Laks V.S. Lakshmanan, University of British Columbia, `laks@cs.ubc.ca`.

[Catriel Beeri, Ronald Fagin, David Maier, Mihalis Yannakakis: On the Desirability of Acyclic Database Schemes. JACM 30(3): 479-513(1983). (Preliminary version: Beeri, C., Fagin, R., Maier, D., Mendelzon, A.O., Ullman, J.D., and Yannakakis, M.: Properties of acyclic database schemes. Proc. ACM STOC (Milwaukee, Wisc., May 11-13, 1981), ACM, New York, 1981, pp. 355-362.)]

As a doctoral student at the Indian Institute of science, Bangalore, I began looking for an interesting thesis topic. A little known article in the American Mathematical Monthly on the foundations behind database normal forms had convinced me, it *had* to be on databases. It was then that this article about acyclic hypergraphs and their applications for database schemes and dependency theory caught my attention. (The speed of communication at the time made it almost impossible for me, sitting in India, to get hold of the STOC paper before the JACM version!) I was delighted as I read this paper. Its main contribution was to show the equivalence between acylcicity (of which there are various degrees as expounded elsewhere) of hypergraphs and each of some 11 desirable properties of database schemes. Let me give two examples. (1) Testing (global) consistency of a database instance, i.e., whether it is admits a universal relation, is in general NP-complete. For a pair of relations, this can be trivially done in polynomial time. Such pairwise consistency implies global consistency exactly for databases over acyclic schemes. (2) A join dependency (JD) is equivalent to a (conflict-free) set of multi-valued dependencies (MVDs), a very nicely behaved class in their own right, exactly when the JD is acyclic.

As it turned out, this paper made a deep impact on my thinking about relational databases and dependency theory at the time. On reading it, I was revved up for a whole series of works building on this paper and knew exactly what I wanted to do. In particular, my papers on efficient recognition and generation of alpha/beta-acyclic database schemes, on conflict-free MVDs, and on the tight relationship between beta-acyclic hypergraphs and strongly chordal graphs were profoundly influenced by this paper.

It is a testimony to the fundamental nature of hypergraph acyclicity as a concept that it has shown up, in addition to in classical database problems, in such diverse contexts as information integration (PODS'96), constraint satisfaction in AI (AI Journal, 1994), polynomial time solutions for NP-complete problems when restricted to acyclic instances (Acta Informatica 1983), meta-querying and tree decomposition of queries (PODS'99 and '00), to mention a few.

This is one of the best papers establishing a compelling connection between graph theory and database theory.

Limsoon Wong, Kent Ridge Digital Laboratories, `limsoon@krdl.org.sg`.

[R. M. Burstall and J. Darlington. A Transformation System for Developing Recursive Programs. JACM, 24(1), pages 44–67, January 1977.]

In this paper, Burstall and Darlington showed how one program could be transformed into an equivalent program using a set of clear and simple laws. I was still an undergraduate student when I read this paper. It showed me certain concepts of computer science in general and programming languages in particular that I had not encountered at that time: that computer programs had logical structure; that programming languages had invariants; that the logical structure and invariants could be made explicit, could be studied, and could be exploited to solve problems; and that programming could be made a rigorous process.

The splendour of these concepts and the elegance of the Burstall-Darlington approach have asserted a great influence in my own development. I fell in love with the logic and semantics of programs and with functional programming because of this paper. I even entered graduate school so that I could pursue this love affair! Mike Smyth recommended me to UPenn. That was very fortunate. At UPenn, Peter Buneman and members of his lab were developing many interesting ideas in database query languages based on exactly these things that I loved. The rest was history—the flavours of functional programming, logics of programs, invariant properties, and so forth permeated many of my query language papers.
