# Reminiscences on Influential Papers

*Richard Snodgrass, editor*

This column celebrates the process of scientific inquiry by examining, in an anecdotal fashion, how ideas spread and evolve. I've asked a few well-known and respected people in the database community to identify a single paper that had a major influence on their research, and to describe what they liked about that paper and the impact it had on them. These contributions emphasize that research has a strong emotional component, involving joy, beauty, amazement, and friendship, sometimes all at the same time.

---

**Serge Abiteboul**, INRIA, `Serge.Abiteboul@inria.fr`

[A. K. Chandra and D. Harel, "Computable Queries for Relational Data Bases," *Journal of Computer and System Sciences* 21(2):156–178, 1980]

Chandra and Harel's papers in JCSS in 1980 and 1982 have been very influential. At that time, we had a very limited vision of query languages. This was essentially coming from Codd's original papers. Codd implicitly put some limits to query languages, "(Codd) query completeness". Since his work was mathematically founded, there was some reluctance to question it. The idea that one could define classes of queries independently of any specific language was somewhat new. By pushing this, Chandra and Harel opened the ground for a large body of works on expressivity and complexity. Some of the techniques they introduced turned out to be applicable to a large variety of contexts.

---

**Sophie Cluet**, INRIA, `Sophie.Cluet@inria.fr`

[B. P. Jenq, D. Woelk, W. Kim and W-L. Lee, "Query Processing in Distributed ORION," in *Proceedings of the Conference on Extending Data Base Technology*, pp. 169–187, Venice, Italia, 1990]

Some people say that the ODMG model does not support a declarative query language (but it does: OQL) whereas the object-relational one provides such a support (in the future it will: SQL3). Another rumor is that object-oriented queries cannot be optimized. This landmark paper (published before the ODMG was even created) proves that this statement is about as true as the one, often heard in the seventies, claiming that SQL could not be efficiently implemented.

Jenq, Woelk, Kim and Lee were the first to state an important and often ignored fact: navigational and join queries are equivalent. In other words, relational optimization techniques can be applied to OQL. As a matter of fact, the support of both associative and navigational accesses is a source of more optimizations. Objects can be stored on disks in many different ways. If some objects are clustered with their components, navigational queries are much more efficient than any joins one could perform on two relations storing the same information. Conversely, if they are stored on different pages, one can rely on joins to obtain performance similar to that found in relational systems. Hence, you can have the best of both worlds. Isn't that beautiful?

**Michael Franklin**, University of Maryland, franklin@cs.umd.edu

[G. Copeland and D. Maier, "Making Smalltalk a Database System," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 316–325, 1984]

I first saw this paper in 1985 or 1986 while I was a student at the Wang Institute. I was spending much of my time on two excellent courses: The first was a class on transaction processing taught by Phil Bernstein, based on the book he had just finished with Hadzilacos and Goodman. The second was a seminar on Object-Oriented Programming Languages. At the time, these two subjects seemed completely unrelated—the database course focused on bulletproofing COBOLish-looking applications that seemed most naturally written in ALL CAPS, while our Smalltalk system had a nice GUI and a natural way of modeling data but could barely run toy programs. When I first saw the Copeland and Maier paper, I was amazed to find that they were attempting to merge these two worlds. The paper, however, made a convincing argument that this was a natural and important thing to do. Upon graduation, I jumped at the chance to work on extending these ideas with George and the Bubba group at MCC.

This paper is one of those classics that caused a lot of people to re-evaluate their basic assumptions. In addition to identifying and proposing to eradicate the "impedance mismatch" between procedural languages and database systems, this paper also described language constructs and an architecture for doing so. The resulting system included user-defined type extensions with methods, type hierarchies, historical data access, and a uniform language for database queries, navigation, and general programming. While there had been previous work on enhancing the relational model and on adding persistence to programming languages, this paper was ground-breaking in its attempt to seamlessly combine ideas from databases, programming languages, and operating systems in a way that could support a wide range of applications. Today's database systems have not generally gone the full distance towards reducing the impedance mismatch, but the ideas in this paper have clearly influenced the direction of the field and the industry. The paper serves as a reminder that in order to make progress, we must be constantly looking beyond the limitations imposed by current technology and ways of thinking.

---

**Guy Lohman**, IBM Almaden Research Center, lohman@almaden.ibm.com

[P. G. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price, "Access Path Selection in a Relational Database Management System," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 23–34, Boston, 1979]

This paper summarized how System R optimized SQL queries, and it is now probably the most oft-quoted fundamental paper in relational query optimization. So it shouldn't be surprising that it had a very significant impact on my career, even though I had neither joined IBM nor started working in query optimization at the time. On a personal level, Pat's presentation of the paper impressed me immensely, prompting a dialogue that ultimately led to Pat's becoming my manager when I first joined her R* project at IBM in 1982. Our collaboration, respect, and friendship has grown ever since.

But the importance of this paper to the field far overshadows its impact on me personally. My copy of it is worn and covered with marginal comments from careful study. After numerous re-readings, I continue to be amazed by how packed this paper is with insight and well-reasoned decisions, and by how many ideas first articulated in this paper have withstood the test of time and competition. Every major DBMS vendor now has a query optimizer that exploits most of the fundamental concepts introduced in this paper: using database statistics and a model of execution costs—rather than simplistic rules—to prune dominated access strategies; avoiding Cartesian products and exploiting dynamic programming and re-use of subplans to reduce the exponential search space of alternative join orders; estimating the cardinality of intermediate results using probabilistic "filter factors"; distinguishing plans having different "interesting" orders; identifying how predicates can be most efficiently applied (and coining the memorable term SARG, for "Search ARGument"); pipelining intermediate results to avoid materialization; the evaluation of nested subqueries; and many more details too numerous to list here. While each of these innovations have been significantly refined and improved upon over the last 20 years, the staying power of the original ideas is, to me, quite remarkable. It is a truly seminal work.

---

**David Lomet**, Microsoft Research, `lomet@microsoft.com`

[J. Gray, "Notes on Database Operating Systems," IBM Technical Report RJ2188, 1978. Also published in Operating systems: An Advanced Course. Springer-Verlag Lecture Notes in Computer Science. Vol. 60. New York.]

By the late 1970's and early 80's, there were a handful of successful transactional database systems that had been built, IMS, System R, Ingres, Oracle. However, the magic by which these systems implemented transactions was, with the exception of one paper, hidden deep in the depths of the system implementations. That exception was the "Notes" paper written by Jim Gray. Here, for the first time, were brought together in one place commit protocols, concurrency control, and recovery. Indeed, before this paper, there was excruciatingly little on recovery in the literature. In this one paper it was possible to learn about write-ahead logging, the DO-REDO-UNDO log record protocol, recovery checkpointing, restart, several flavors of two phase commit, multi-granularity locking, strict two phase locking, etc. In addition, one could learn about recoverable messages, record management, and a variety of systems issues. The paper not only described solutions to hard problems, it provided insights into what was important and why.

I avidly read the "Notes" paper when it came out (and I was surely not alone) and returned to it again and again over the years. The paper is a bit dated now, superseded partially by the System R recovery paper, the Bernstein, Hadzilacos, and Goodman text book, and fully by the monumental book authored by Jim himself (together with Andreas Reuter). But the "Notes" paper, which did not appear in either a conference or a journal, was the single most useful paper in the database literature for an entire generation of transaction system researchers and implementors, myself among them. Indeed, my interest in recovery dates from the reading of the "Notes" paper, so it continues to have an influence on my research directions to this day.

---

**Gultekin Özsoyoğlu**, Case Western Reserve University, `tekin@ces.cwru.edu`

[J. M. Smith, and D. C. P. Smith, "Database Abstractions: Aggregation and Generalization," *ACM Transactions on Database Systems* 2(2):105–133, June 1977]

I read Smith and Smith's paper while looking for a PhD thesis topic in the summer of 1977. The paper was extremely impressive with its depth, maturity and original thoughts in introducing "aggregation and generalization hierarchy abstractions for objects" in data modeling. While it applied these hierarchies to the relational model and introduced "relational invariants," the paper was really about semantic data modeling with objects, i.e., object-oriented databases! I remember reading it several times that summer. In the following years, I routinely came back to this paper, and made it a must-read for my students.

Smith and Smith's paper had a significant effect on my research during my PhD and the following years. Eventually I ended up working on statistical database security as my PhD thesis. At the time, due to the need for formal inference control analysis, most statistical database security research used simple data models for statistical databases. One research direction that I took was using richer data models in general, and incorporating the generic hierarchies of Smith and Smith's paper in particular, into statistical databases, and still providing inference control guarantees. This started a long line of statistical database security research for me that continued well into mid-80's.

---

**Raghu Ramakrishnan**, University of Wisconsin, raghu@cs.wisc.edu

[F. Bancilhon, D. Maier, Y. Sagiv, J. D. Ullman, "Magic Sets and Other Strange Ways to Implement Logic Programs," in *Proceedings of the ACM Principles of Database Systems Symposium*, pp. 1–16, 1986]

I joined the LDL group at MCC Austin as a part-time member in 1985, while a graduate student at UT-Austin. I was looking for a thesis topic, of course, and had done some work in both databases and logic programming. So I was ripe for the ideas in this elegant paper. The paper showed how the binding propagation achieved by Prolog could be captured through a simple program transformation for a certain class of programs, and suggested that database implementation techniques (e.g., efficient join methods) could be brought to bear on recursive queries.

Over the years, I've returned time and again to this issue, trying to generalize the approach to larger classes of programs, and to understand its implications in the broader context of guiding search strategies. This paper influenced a number of other researchers as well, and today many commercial database systems utilize variants of the Magic Sets idea. For me, it will always remain special as the paper that taught me research could be fun, and simple ideas could cut deep.

---

**Ken Ross**, Columbia University, kar@cs.columbia.edu

[C. Nyberg, T. Barclay, Z. Cvetanovic, J. Gray, D. Lomet, "AlphaSort: A Cache Sensitive Parallel External Sort," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 603–627, 1994, and later in the *VLDB Journal* 4(4):603–627 (1995)]

The impact of the AlphaSort paper on me was primarily the realization that cache behavior was particularly important for the performance of data intensive operations. I was particularly impressed by the clear way that cache behavior was controlled, and by the high cache hit ratios obtained. I could foresee a time when for many reasonable applications, the entire database could fit into main memory. In that context, cache behavior would be the critical performance factor, since the gap between processor speed and main memory speeds was widening (about 2 orders of magnitude over the last 12 years). This observation would hold not just for sorting, but for all database operations. Cache performance issues are now a central theme in my new main-memory database project at Columbia.

**Timos Sellis**, National Technical University of Athens, timos@dblab.ece.ntua.gr

[A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 47–57, 1984]

It is clear that all of us have been influenced by a lot of papers throughout our academic careers. Since Rick asked me to identify only one of them, I could not pick any other paper but the one that influenced the largest part of my research, namely my work on spatial access methods. I was very lucky to meet Antonin Guttman during my PhD studies at Berkeley and had a good opportunity to discuss with him the properties of R-Trees. The reason why I liked this paper, when I first read it, is that it addressed a very important and difficult problem—indexing multi-dimensional spaces where no obvious sort-order can be defined—and it suggested a nice, clean and elegant method to solve it. In my opinion, this work opened up a whole new area of research that led to many well-known good data structures and algorithms, which have even been adopted by commercial systems. One can see 15 years later, papers still being published around improvements and application-specific variations of R-Trees (in image, spatial and, temporal DBs, as well as in more "exotic" areas like data warehouses), continuing a very interesting line of research that Antonin opened up with his paper in 1984.

---

**Patrick Valduriez**, INRIA, Patrick.Valduriez@inria.fr

[M. M. Zloof, "Query-By-Example: Operations on the Transitive Closure," IBM Research Report RC 5526, Yorktown heights, New York, October, 1976, and in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 47–57, 1984]

Like many French people, I was brought up to appreciate elegance and simplicity, as can be found in an arrangement of exquisite cheese, bread and wine. I guess this helped me in my research career when designing data structures and algorithms which could be easily implemented. Zloof's paper is a good example of scientific elegance and simplicity. I read it in 1985 while at MCC, trying to figure out the practical aspects of deductive databases, then a hot topic. At first, deductive databases had appeared to me as complex and impractical. Written many years before the fashion of recursive queries, Zloof's paper showed that a powerful, yet simple, form of recursion could be naturally added to a relational, domain-calculus based, language. Then one could extend relational algebra with transitive closure and design data structures and algorithms for it. This has strongly influenced my work on transitive closure algorithms, join indices and the Fad database programming language. As wider evidence of its practical impact, transitive closure has been recently added to SQL as a standard construct.