Reminiscences on Influential Papers

Richard Snodgrass, editor

This new column celebrates the process of scientific inquiry by examining, in an anecdotal fashion, how ideas spread and evolve.

Fundamentally, we are in this business to embrace the novel concepts that intrigue and delight us. These insights are generated or discovered by one or a few, then somehow propagated through the fabric of the intellectual community, via publication, presentation, and informal conversation. However, the transference is often more accidental than intentional. The common model is of a researcher having a question and searching for the paper that answers that question. Seemingly just as prevalent is a haphazard event. The researcher comes across a paper that touches something deep inside, triggering a radical restructuring of their mental model and allowing them to see things in a new light. Timing is important: often that paper or interaction would not have had such a profound effect if encountered earlier or later.

I've asked a few well-known and respected people in the database community to identify a single paper (often a challenging task!) that had a major influence on their research, and to describe what they liked about that paper and the impact it had on them. Their responses make fascinating reading, and illuminate the sometimes unpredictable journey of good ideas.

Elisa Bertino, University of Milan, `bertino@dsi.unimi.it`

[P. P. Griffiths and B. Wade, "An Authorization Mechanism for a Relational Database System," *ACM Transactions on Database Systems*, 1(3):242–255, 1976]

It has been very easy for me to indicate a paper that has greatly influenced my research. This paper is the one by Pat Griffiths and Bob Wade on the System R authorization model. The reason why I liked this paper, when I first read it, is that it addresses a real, important problem—the problem of access control in database systems—and at the same time it provides a nice, theoretical foundation for reasoning about access control models and mechanisms. The approach described in that paper has been the basis of access control mechanisms of several commercial DBMS. When, years later, I decided to do some research work on database access control mechanisms, I used the approach proposed by Pat and Bob in that paper as the starting point of my research. I and some colleagues of mine have now developed several extensions to the access control model originally proposed for System R, including non-recursive revoke operations, negative authorizations, temporal authorizations. Last but not least, the paper also provides a nice guideline for teaching students about access control mechanisms in relational DBMS.

Mike Carey, IBM Almaden Research Center, carey@almaden.ibm.com

[C. Zaniolo, "The Database Language GEM," in *Proceedings of the 1983 ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1983, D. DeWitt and G. Gardarin, eds., pp. 207–218.]

This is one of my all-time favorite database papers—and I'd recommend it highly to anyone working on object extensions to relational database systems. In a nutshell, this paper extended the relational model and a relational query language (Quel) to provide for more strongly typed tables, generalization, references, path expressions, and set-valued attributes. What makes this paper such a gem (pun intended!) is that its extensions are remarkably clean, simple, and natural. While some "relational bigots" will argue even today that relations and objects are like oil and water, my view is that Zaniolo in many ways proved them wrong through his GEM research almost 15 years ago. Some of the highlights of this paper are its introduction of the dot-notation for path expressions, its clear treatment of the interplay between null values and path expressions, its unique type system, and its treatment of sets. This paper strongly influenced the object-oriented data model and query language work that we did in the context of the EXODUS project at Wisconsin; it continues to influence my view of the world in my current work on object-relational extensions for DB2 UDB and SQL3.

Jim Gray, Microsoft Research, gray@MICROSOFT.com

[D. Bitton, D. J. DeWitt C. Turbyfill, "Benchmarking Database Systems: A Systematic Approach," in *Proceedings of the International Conference on Very Large Databases*, October, 1983, Florence, Italy, M. Schkolnick and C. Thanos, eds., pp. 8–19.]

During the early 1970's there was great enthusiasm for database-machines, special hardware that would somehow solve the performance problems that plague database systems to this day. The theory was that if the brilliant database researchers could just bypass the file system and operating system, if they could just get to the bare metal, things would be ten times faster. Being an OS guy at heart, I was puzzled that these folks thought they could do IO better than we could—they did not seem to understand what interrupts were. Not just that, they seemed not to appreciate all the subtleties of error handling, multi-programming, multi-processing, security, and so on. One day I had a epiphany: I read the Bitton-DeWitt-Turbyfill paper. Suddenly, I realized what was going on: I had been working on what would now be called online-transaction-processing problems, while all my database machine friends were working on what would now be called data mining. They were doing joins, they were scanning the whole 4 MB database, while I was doing tiny transactions on giant (500MB) at the time databases. I was worried about security, concurrency, recovery, manageability, while they were worried about sequential scans, aggregates, and especially joins.

To crystallize this difference I wrote a puff-piece: "A Measure of Transaction Processing" that described an OLTP transaction that eventually gave rise to the Transaction Processing Performance Council and the TPC-A benchmark. It included a user-level mini-batch transaction (copy 1,000 records) and a batch transaction (sort a million records). Unfortunately, it left out a backup-restore job. This paper circulated among a large crowd. About 25 people added something. To avoid the legal hassles of getting the ATT and DEC and Xerox and IBM and Tandem lawyers to agree, we published it under the pseudonym Anon Et Al. The paper appeared in April Fools Day issue of Datamation in 1985. To this day, we award the SIGMOD Sort trophies on that date (see

http://www.research.microsoft.com/barc/SortBenchmark/DEFAULT.html). Once in a while I got letters to Dr. Anon (the Tandem mail room knew and enjoyed the joke).

The irony of this story is that the "Wisconsin" benchmark eventually spawned the TPC-D benchmark. TPC-D now is center-stage in the great DBMS performance wars.

Henry F. Korth, Bell Laboratories, Lucent Technologies Inc., hfk@lucent.com

[J. N. Gray, R. A. Lorie, G. R. Putzolu, and I. L. Traiger, "Granularity of Locks and Degrees of Consistency in a Shared Data Base," in *Modelling in Data Base Management Systems* (G. M. Nijssen, ed.), North Holland Publishing Co., 1976, pp. 365–395.]

When I was approached to write a short note on a single paper that had a major influence on my research, the choice of paper was immediately clear to me. The key word was was *influence*. While I've read more excellent papers than I can count (even though the number is countable!), only a very few individual papers have had a major impact on my research agenda and how I thought about research problems. The Gray-Lorie-Putzolu-Traiger paper on lock granularity is not only the first paper to have such a degree of impact on my work, but also the most influential.

I first read this paper in 1978 as a graduate student considering research in databases, a field of whose very existence I had been unaware only a year earlier. Jeff Ullman (my advisor) had given me some fairly theoretical papers pertaining to database concurrency control (all of which later appeared in *J. ACM*). While those were indeed fine papers, I lacked any intuition about the real-world problem of database transaction processing, and thus found it hard to identify new interesting problems. Jeff then pointed me to the System R work including the lock granularity paper cited above. It was in reading these papers, and the lock-granularity paper in particular, that I connected the formal concepts of correct concurrent executions with the practical requirements of low-overhead, high-concurrency protocols. It was in thinking about the ideas introduced in the paper that I chose database concurrency control as my dissertation topic (and imagined the authors to be elderly men with short, gray hair). The impact of this work was enhanced immeasurably by an internship the following summer (1979) with Jim Gray, Pat Selinger, and the System R group (once I got over the shock of correcting my mental image of the authors and seeing a "business" office with a beanbag chair). In retrospect, the influence of this paper may be in part to having reached me at just the right time, but I think the primary source of its influence was its mix of theory, then-current practical problems, and connection to existing systems.

Betty Salzberg, Northeastern University, salzberg@ccs.neu.edu

[C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh and P. Schwarz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," *ACM Transactions on Database Systems*, 17(1):94–162, March 1992.]

The ARIES paper was important for me because it enabled me to envision the mechanisms of recovery in database systems clearly. For example, I saw how Log Sequence Numbers (LSNs) are used to enforce Write-Ahead-Logging (WAL). WAL says that before a page with an update on it made by an uncommitted transaction can be written to disk (overwriting the previous version of the page), the pre-image of the updated record must be on disk somewhere else. But it is

always important to understand some mechanism by which a theoretical rule can be enforced. The mechanism commonly used for WAL is the LSN. The LSN = L on a database page P in the buffer in main memory is the LSN of the log record of the most recent update on P. Log records contain preimages of updated records and log records are written sequentially in increasing LSN order. If the LSN of the most recent log record written to disk is smaller than L, WAL implies that P cannot yet be written to disk. First, a portion of the log containing the log record with LSN = L must be written to disk. This is one of many recovery mechanisms I did not know and I think many other database researchers did not know until preprints of the ARIES paper were made available.

Reading the ARIES paper influenced much of my subsequent research. My research on concurrency and recovery for B-link-tree-like access methods (the Π-tree and the hB-Π tree) for example, uses LSNs to determine whether an index page has been updated since the last visit. (If it has not been updated, a new search through the tree can be avoided.) Issues of latches vs. locks and support for fine-granularity locking, exposed in the ARIES paper, were essential in the Π-tree and the hB-Π tree. The concept of page-oriented vs. logical UNDO was explained in the ARIES paper and used in the Π-tree and hB-Π tree. My research on transactional workflow (DSDT) and on online reorganization uses the method of repeating history from log records (from ARIES) to recreate system tables and/or reestablish the state of an ongoing application. Now it is almost impossible for me to imagine thinking of a database system without ARIES style recovery.

Dennis Shasha, New York University, `shasha@shasha.cs.nyu.edu`

[P. L. Lehman and S. B. Yao, "Efficient locking for concurrent operations on B-trees," *ACM Transactions on Database Systems*, 6(4):650–670, December 1981.]

My favorite papers have always been the ones that cause me to change my intellectual prejudices. I read Lehman and Yao's paper on B-trees while trying to find a thesis topic in 1981. I had just studied concurrency control theory with Phil Bernstein and Nat Goodman while they were both at Harvard and was convinced that conflict-preserving serializability was the end of the story. Lehman and Yao's paper showed executions that were clearly correct, but didn't seem to fit this model. After trying to stretch the model for the next year, I realized a new one was needed and wrote my thesis on a generalized model for concurrency in index structures.