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.

Jiawei Han, Simon Fraser University, `han@cs.sfu.ca`.

[R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB 1994, pages 487–499, Santiago, Chile, September 1994.]

Although it was just published less than seven years ago, this paper has already had a strong impact in the field of data mining: hundreds of followers, and also one of a few most popularly cited research papers in data mining.

I feel the paper makes two clear contributions to the field.

First, together with Agrawal, Imielinski, and Swami's SIGMOD'93 paper: "Mining Association Rules between Sets of Items in Large Databases," it identifies a new and important task in data mining: association rule mining, i.e., finding frequent patterns or itemsets (sets of items) that occur frequently together in large databases. This has proven truly useful for frequent pattern or association mining, dependency or correlation analysis, etc., with many applications. Some following studies have shown that it is also useful for association-based classification, sequential or structured pattern analysis, constraint-based mining, cluster analysis, semantic data compression, data cube computation, and so on. Identification of a crucial research problem itself makes the paper distinct from many others.

Second, it discovers a nice and elegant property in association mining, namely Apriori, which states that a length-(k+1) pattern cannot be frequent if any of its length-k sub-patterns is infrequent (under the same minimum support threshold). This property has led to a powerful pruning technique, called candidate generation-and-test: first generate only the promising length k+1 candidates based on the frequent itemsets of length k, and then test it against the database. The Apriori property prunes a large set of unpromising candidates using a multiway cross-checking technique. Moreover, the paper introduces some techniques for efficient implementation of Apriori, including an ordered join method and a hash tree technique. The Apriori pruning method plays a role in data mining similar to the golden rule, "pushing selection first", that has been playing in relational query processing. It has become a golden heuristic soaked in many efficient data mining algorithms developed later.

Thus, this is a milestone paper, perhaps the most influential one in the field of data mining so far.

Nick Koudas, AT&T Research, `koudas@research.att.com`.

[P. Ferragina and R. Grossi. String B-tree: A New Data Structure for String Search in External Memory and its Applications. Journal of the ACM, 46(2):236–280, 1999.]

Not so often, theory meets practice in an elegant way. The result is commonly conceptually appealing and of vast importance. The paper "String B-tree: A New Data Structure for String Search in External Memory and its Applications" is an instance of such a proximity. The ideas for this work have been around by the same authors since 1995. Although B-trees, provided an excellent solution to the problem of indexing one dimensional numerical domains, it was an open question whether similar data structures exist, capable of providing performance and space utilization guarantees for the problem of indexing strings of unbounded length. The String B-tree paper of Ferragina and Grossi settles this question, by showing that such a data

structure supporting common queries on strings in secondary storage is indeed possible and in fact can be efficiently implemented.

Given the vast amount of string information available, handling large strings efficiently is imperative now and would persist as a need in the future. Although strings have been a subject of primary focus in related communities the existence of structures more familiar to database people for processing strings, bridges a conceptual gap in some sense. The core ideas behind this paper have already influenced the work of various researchers, including our own in multidimensional string and substring indexing as well as approximate string join processing, and are likely to influence the work of more in the years to come.

Rajeev Rastogi, Lucent Technologies, Bell Laboratories, `rastogi@research.bell-labs.com`.

[K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger. The Notions of Consistency and Predicate Locks in a Database System. Communications of the ACM, 19(11):624-633, November 1976.]

A reasonable measure of a paper's influence is its impact on the real world. A piece of work can be considered to be truly impactful if it satisfies the following criteria: (1) The work constitutes an important innovation, resulting in a dramatic change in the way people do things or in the way systems are built, (2) The ideas underlying the work are extensively deployed in real-world systems, (3) The work helps create tremendous efficiencies, improves productivity and enables organizations or individuals to realize substantial cost savings, and (4) Complex tasks are considerably simplified due to the ideas proposed in the paper.

The choice for a paper that meets the above 4 criteria is pretty clear when I add the additional constraint that the paper should have influenced my research a great deal. Having done my dissertation on the topic of concurrency control in database systems, my choice for the most influential paper has to be the seminal paper on this subject by Eswaran, Gray, Lorie and Traiger. The paper proposed the now well-known two phase locking (2PL) protocol that is implemented by virtually all of today's commercial DBs.

The 2PL protocol represents a significant scientific innovation, that has considerably simplified application development by freeing application programmers from having to explicitly obtain and release data item locks. The protocol is elegant and simple. It is local in the sense that each transaction independently and locally makes decisions on obtaining/releasing locks - yet the protocol is very powerful since it enforces a global property, that of serializability. The simplicity and non-global characteristic of the protocol also make it ideal for implementing serializability in distributed environments.

The effectiveness of the 2PL protocol is further corroborated by the fact that it has withstood the test of time – 25 years after it was originally proposed, commercial DBs still use 2PL for concurrency control. Since my early years as a graduate student, I have always strived to do the kind of innovative, elegant and impactful research that the 2PL paper of Eswaran et. al. embodies.

Daniel Rosenkrantz, University at Albany — SUNY, `djr@cs.albany.edu`.

[K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger. The Notions of Consistency and Predicate Locks in a Database System. Communications of the ACM, 19(11):624-633, November 1976.]

In the mid-70s, I was working in the General Electric Research and Development Center, where within the group I was part of, John Hutchison led the design and implementation of "MADMAN", a CODASYL network-model database system. The driving application was a database system to be deployed in a factory that manufactured nuclear fuel. Database correctness and accuracy were essential to the factory; if the database lost track of where some of the fuel was, the factory would be shut down. The issues arising in this system and the requirements arising in this application enticed Phil Lewis, Dick Stearns and me to look into database concurrency control. We read the manuals of several commercial database systems, which contained detailed descriptions of various locking and rollback schemes, but found very little about the abstract requirements on a database concurrency control, or what would be achieved by using the

available mechanisms. While we were formulating a general approach to consistency, locking, and deadlocks, particularly in distributed systems, we encountered a preliminary version of this paper. The paper provided a complete package. It gave strong and convincing motivation for its ideas, used exactly the right level of abstraction, contained clean definitions, gave examples, and included solid proofs. Each theorem was accompanied by an example illustrating the pitfalls of not following a stated necessary condition, and an example of how things work out well when the necessary condition is satisfied. The paper made a clear case for why serial schedules preserve consistency, and why any schedule equivalent to some serial schedule preserves consistency. The paper formulated the concept of two-phase locking, and proved its relationship to serializability.

The paper also introduced predicate locks, which seemed like an appealing idea. The paper pointed out that checking for conflicts between arbitrarily complex locking predicates is undecidable, and then defined a special class of "simple" predicates, giving a procedure for testing conflicts between such predicates. The paper inspired Harry Hunt and me to investigate algorithmic and complexity issues involved in processing predicate locks, with various constraints on the permissible predicates.

Peter Scheuermann, Northwestern University, `peters@ece.nwu.edu`.

[Yuri Breitbart, Avi Silberschatz, and Glenn Thompson. Reliable Transaction Management in a Multidatabase System. SIGMOD 1990, pp. 215–224.]

I became interested in transaction processing in multidatabase or federated database systems while organizing with Clement Yu the NSF Workshop on Heterogenous Database Systems, held in December 1989 in Evanston. The objective of the workshop was to identify new techniques and functionality to support the interoperability of autonomous database systems without requiring their global integration. Prior to that time, the majority of research in multidatabase systems was concerned with data and schema integration and retrieval languages, largely ignoring transaction processing issues. At the workshop, Yuri, Avi and Glenn presented preliminary ideas about transaction processing that required making some reasonable restrictions on the type of data or operations that the transactions can use in order to preserve the systems autonomy. I found the ideas very intriguing, and this area of research one full with rich opportunities.

A few months later, Yuri, Avi and Glenn published their SIGMOD paper, which presented the first formal treatment of transaction processing in multidatabase systems in the presence of failures. What Hector-Garcia Molina did earlier with Sagas for semantic database consistency, this paper accomplished for syntactic database consistency. One of the main ideas of the paper was to argue that two-phase commit protocol, which has been accepted as the standard for guaranteeing the atomicity of transactions in distributed systems, was not applicable to multidatabase systems. The paper showed that two-phase commit was not applicable for an environment where the autonomy of the local systems cannot be violated. The idea of inapplicability of the two-phase commit was very controversial at that time, and the paper presentation at SIGMOD generated a lively discussion. The paper also presented a new, elegant model for consistency and concurrency control in multidatabase systems in the presence of failures as well as efficient algorithms to implement it. Since then, many researchers have acknowledged that the use of two phase commit protocol in a distributed environment with either strong requirements of site autonomy or with a large number of local sites is not feasible in a distributed database environment. "Reliable Transaction Management in a Multidatabase System" has stimulated many researchers, and many, including myself, found ourselves productively working on extending their model.

Dan Suciu, University of Washington, `suciu@cs.washington.edu`.

[Gerd Hillebrand and Paris Kanellakis. "Functional Database Query Languages as Typed Lambda Calculi of Fixed Order". PODS 1994, pp. 222–231.]

Database query languages and functional programming languages should be very similar. After all, they both study some *languages*, and both claim to use logic as their foundation: First Order Logic and the

Lambda Calculus (also a branch of logic) respectively. Yet, at a closer look, the two areas share almost nothing in common, neither at the theoretical nor at the practical level. In query languages, evaluation means checking a formula in a model, in programming languages it means performing beta-reductions of lambda terms. I was deeply impressed by the surprising connection established by Hillebrand and Kanellakis between these two paradigms in *Functional Database Query Languages as Typed Lambda Calculi of Fixed Order*. The paper encodes databases as type lambda terms, in a rather natural way. Then, it shows that the fragment of Typed Lambda Calculus (TLC) corresponding essentially to first-order functions can express all relational queries. This is nice, and perhaps in line with our expectations: we knew that query languages are essentially first-order. But then they show that if we allow second-order functions, then TLC can express precisely the PTIME queries, and these are, of course, precisely the recursive queries. It is striking to me that there exists a connection between second-order functions in programming languages, and recursion in query languages.

So why is this paper influential? Other papers have impacted existing systems in direct ways; clearly applications of the Typed Lambda Calculus are not imminent. In my view, the deep connections that a researcher knows and that have not yet found their way into everyday life are the strongest weapons in his arsenal. This paper made me think, and here lies, for me, its greatest influence.
