Reminiscences on Influential Papers

When I started my PhD, I wanted to do something related to *systems* but I wasn't sure exactly what. I didn't consider data management systems initially, because I was unaware of the richness of the systems work that data management systems were build on. I thought the field was mainly about SQL. Luckily, that view changed quickly.

This issue's contributors cover a variety of topics targeting different layers of the systems work in data management: data management on specialized and resource-constrained hardware, design of storage managers, and synergies across data management, data processing, and machine learning systems. They all underline the richness of the field. Enjoy reading!

While I will keep inviting members of the data management community, and neighboring communities, to contribute to this column, I also welcome unsolicited contributions. Please contact me if you are interested.

Pınar Tözün, *editor*
IT University of Copenhagen, Denmark
pito@itu.dk

Tilmann Rabl
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
tilmann.rabl@hpi.de

Christophe Bobineau, Luc Bouganim, Philippe Pucheral, and Patrick Valduriez.

PicoDBMS: Scaling down Database Techniques for the Smartcard

In Proceedings of the 26th International Conference on Very Large Databases, pages 11-20, 2000.

I was always fascinated by complex software systems and advances in hardware. When I started my PhD, in the search for a topic, I enjoyed many database systems papers and while I generally followed the trend of thinking about larger and larger databases, I also liked work that considered specialized and constrained environments.

One of the papers that was an inspiration during my PhD and continues to influence my work is "PicoDBMS: Scaling down Database Techniques for the Smartcard" by Bobineau et al. It is a very nice instance of a prototypical database systems paper of the time: it is easy to follow, but contains many interesting and thought-provoking problems and solutions. I especially like the focus on special but widespread hardware and the pragmatical solution, which trades off performance and efficiency in a practical setup.

Based on the preposition that structured data should always be managed in a database system, the authors design a database management system for smart cards. The first thing I learned from the paper is that smart cards not only come in the form of memory cards but can contain fully programmable microprocessors that can execute arbitrary code, but only while powered through a reader device. Back then, and even now, smart cards have very constrained resources, which requires rethinking database architecture.

The authors did so in an extreme way. They design custom storage formats that optimize for space efficiency by selecting between classical flat storage, domain storage, which encodes attributes as pointers to a separate domain table, and ring storage. Ring storage stores pointers between tables (e.g., key relations) and between tuples and attribute domains in a circular fashion, to enable bidirectional lookups with very limited space overhead. Just as interesting, the authors discuss the problem of query processing with a very limited amount of memory. The authors explain how to process queries using extreme right deep query plans, which pipeline

all operators, including joins and aggregations, thus not materializing any intermediate results. They show that their design is feasible for tables with thousands of records.

Besides being interesting technically, the paper also is reminiscent of an approach of privacy and security, which is almost unthinkable today. The idea of giving the user full physical control over their data has been traded in for promises of convenience and better applications in times of cloud-based services, which yet have to demonstrate to be an improvement for the user in privacy-relevant use cases.

The other aspect that the paper addresses and which, in my mind, gets too little appreciation in database research today, is the constrained environment and small applications. Running a DBMS on a smart card means dealing with very limited memory and relatively small amounts of data. Thinking about efficiency at scales below gigabytes and tens of cores, even today makes sense and helps to improve the long tail of applications outside of Internet companies.

Avrilia Floratou
Microsoft Gray Systems Lab, USA
avrilia.floratou@microsoft.com

Mike Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, and Stan Zdonik.

C-Store: A Column-oriented DBMS.

In Proceedings of the 31st International Conference on Very Large Databases, pages 553-564, 2005.

When I was invited by Pınar to write for the "Reminiscences on Influential Papers" column, I initially thought the task would be straightforward. However, upon further contemplation, I realized it was quite challenging. There are many papers that have influenced not only the areas and problems I have been working on, but also my perception of what constitutes true innovation in a field that has been established for over four decades. A truly exceptional paper not only alters the direction of progress in research and industry, but also affects how aspiring researchers approach problems. After much consideration, I ultimately chose to discuss the C-Store paper, which I believe was not only influential for the database community, but also had a significant impact on my understanding of what constitutes high-quality research.

As a fresh graduate student in data management, I did not fully understand the impact that simple ideas can have. Many students, including myself at the time, tended to create complex problem statements and develop complicated solutions, disregarding simpler ideas as being too obvious or unoriginal. The C-Store paper helped me to overcome this tendency. The problem statement was quite straightforward: "How can we design databases to optimize for read-mostly workloads?" The solution and approach were also simple: "Let's store the data in a columnar format and explore how the rest of the DB engine should be designed based on that." Although it is certainly challenging to design and build such a system, the problem addressed and the fundamental design choices behind the system were all quite simple and resonated well with the database community. Another important lesson I took from this paper is that combining ideas from previous work (e.g., the decomposition storage model proposal from the 80s and the substantial work done in the context of the Monet system) with novel ones to build a functioning system and then demonstrating its efficiency can be a significant research contribution. Focusing solely on how to differentiate from others rather than finding the best solution to a problem may actually hinder potential for transformative change.

C-Store was a highly influential development in the database community. As one of the first columnar database systems, it has inspired further research in various areas such as columnar formats, compression techniques, operating on compressed data, and interactions with hardware. The system had a significant impact on industry as well: it serves as the foundation for the Vertica analytic database. Multiple other database technologies such as DB2 BLU, SQL Server Columnstore Indexes, BigQuery Capacitor are all based on fundamental concepts behind C-Store. Furthermore, many of the ideas behind C-Store and columnar databases in general have influenced Big Data analytics platforms such as Hadoop and Spark. For example, in my own work, we studied how to create efficient columnar storage formats for Hadoop and HDFS. Today, Apache Parquet and Apache Arrow are two columnar formats that are leveraged by many modern analytics engines. I believe that the research presented in this paper is a prime example of the kind of work that is needed to truly push the boundaries of the field and advance the state-of-the-art in data management.

Arun Kumar
University of California, San Diego, CA, USA
arunkk@eng.ucsd.edu

Tian Zhang, Raghu Ramakrishnan, and Miron Livny.
BIRCH: An Efficient Data Clustering Method for Very Large Databases.
In Proceedings of ACM SIGMOD, pages 103-114, 1996.

This is one of the foundational papers of the "data mining" area, bridging the areas of databases and ML. Nowadays, since "data mining" is used mainly for algorithmic aspects of applied ML/AI, this paper can be seen as canon in the new "ML systems" area. Its impact, both immediate and transitive, has been massive. On the former, its "cluster feature" idea revolutionized how clustering algorithms work, especially on larger-than-memory data. On the latter, it showed that by taking a joint view of the abstract math involved in ML programs and data systems principles such as decomposing computations on data and staging data movement across the memory hierarchy, one can build better ML systems. That also means we must be mindful of both system metrics such as runtime and ML metrics such as accuracy.

That basic philosophy of reimagining ML workloads as DB-style workloads–decoupling the what of the math from the how of the execution–really inspired me as a PhD student. For instance, one of my papers that introduced the paradigm of "learning over joins" resembles BIRCH in many ways, e.g., unpacking the ML math to explain data access patterns, carefully counting I/O and CPU costs, formally analyzing its efficiency, and extensive empirical evaluation, all in service of making end-to-end ML/AI workloads faster, more scalable, and easier for users. That philosophy has guided my subsequent research as an academic in this now popular sub-area of "data management/systems for ML."

As the BIRCH paper showed a quarter century ago, the synergy between the "DB systems" area and the "ML systems" area is high. After all, DB and ML/AI are two sides of the same coin of data-intensive computing, both at the heart of the software foundations of Data Science. So, I hope the DB community steps up to fulfil its intellectual responsibility to both science and wider society by helping democratize ML/AI and Data Science.

Danica Porobic
Oracle, Zürich, Switzerland
danica.porobic@oracle.com

Surajit Chaudhuri and Gerhard Weikum.
Rethinking Database System Architecture: Towards a Self-tuning RISC-style Database System.
In Proceedings of the 26th International Conference on Very Large Databases, pages 1-10, 2000.

I started learning about computer architecture only at the beginning of my PhD studies and I was immediately impressed by the elegance of RISC designs, even though CISC was much more popular at the time. Before starting my PhD, I have worked on the edge of a complex database system codebase and have experienced some of the pains Chaudhuri and Weikum discuss in this paper. When I first read this paper, I appreciated the succinct representation of the problems, but the analysis and recommendations seemed a bit abstract and vague.

Few years later, towards the end of my PhD, I was thinking about the designs of transaction processing systems for clusters of multicores and "rediscovered" this paper and its clear and well presented discussion of complexities of data management systems and their causes that I was starting to understand better. However, the application design considerations and research agenda recommendations were still vague to me.

After spending a few years in the industry, gaining better understanding of the real world application requirements and working with a mature and well structured data management system, I think I finally understood the analysis and recommendations Chaudhuri and Weikum presented in this visionary paper. It is certainly possible to design clearly separated system components with well defined interfaces and behavior and pick and choose these components for each application and deployment scenario. This paper goes a step further by laying out a vision for tuning and interoperability that would enable different teams to advance the state of the art much faster than anyone can develop a complex, monolithic, system. While many of the problems authors have identified are still there, the proposals have stood the test of time and it is high time they are adopted by the broader community.