Reminiscences on Influential Papers

This issue's contributors all work at the intersection of machine learning and databases. Coincidentally, three of them picked papers that received the VLDB 10-Year Best Paper award, and one of them picked a paper that, in my opinion, should receive the CIDR Test-of-Time award at some point. Their write-ups highlight very well why these four papers still continue to leave a mark on us personally and the real-world data systems. 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

Matthias Boehm
Technische Universität Berlin, Germany
matthias.boehm@tu-berlin.de

Stratos Idreos, Martin L. Kersten, and Stefan Manegold.
Database Cracking.
In Proceedings of the Conference on Innovative Data Systems Research (CIDR), pages 68-78, 2007.

Selecting a single paper for the *Reminiscences on Influential Papers* (RIP) column is generally a difficult task. For me, however, it was surprisingly easy because the *database cracking* paper really stands out to me, for both, personal and technical reasons.

One of the first international conferences I attended as a young PhD student—working on cost-based optimization of message-oriented middleware—was ICDE 2008 in Cancún, Mexico. Apart from the magnificent environment, Martin Kersten gave an awe-inspiring keynote on "The Database Architecture Jigsaw Puzzle", which described the pitfalls and opportunities of high-risk projects using database cracking as one of the examples. At its core, database cracking is all about workload-driven data reorganization in column stores. As queries come in, they partition the columns into qualifying and non-qualifying data, which in turn drives the reorganization of data, and thus, improves future data access. I came out of this keynote truly inspired, by the simplicity and effectiveness of the ideas, but also by the focus on system-oriented research of actually building MonetDB. Apparently, I was not alone – there are famous stories about Goetz Graefe wandering the beaches of Cancún after the keynote, thinking about adaptive indexing and merging.[1]

Back at TU Dresden, my PhD supervisor Wolfgang Lehner had established a weekly paper session, cycled through all members of the group. So on my next opportunity, I deliberately chose to share the database cracking paper. In preparation of this presentation, I went through all the background start-

[1]Interestingly, these stories are true. On confirming with Goetz Graefe, he shared the following: "Yes, that's precisely how it happened. Something about database cracking was bothering me, but I couldn't put my finger on it precisely. So I spent 1 1/2 days walking back and forth on the Cancún beach thinking about it. Much later I came to think of database cracking as "quicksort on a funny schedule". Martin Kersten later agreed with that summary description. In contrast (or "in duality"), adaptive merging is "merge sort on a funny schedule". One important difference is that quicksort and database cracking work best within memory, whereas merge sort and adaptive merging also work for external sorting and external indexes. The other related anecdote is that I visited CWI and presented adaptive merging as an alternative to database cracking. At first, that didn't go over very well but within 1/2 hour the CWI team was engaging with me on deep understanding and comparisons and that eventually led to a productive collaboration. I am still impressed with their professionalism and their open minds!".

ing from the influential VLDB Journal 2000 paper "Optimizing database architecture for the new bottleneck: memory access", over the core database cracking papers from CIDR 2005 and 2007, to advanced techniques for handling updates (SIGMOD 2007) and partial side-ways cracking for multi-attribute queries and tuple reconstruction (SIGMOD 2009). Besides the core cracking ideas, I was very impressed by the breadth and depth of the entire eco-system around the core systems MonetDB/SQL and MonetDB/X100, but also MonetDB/XQuery (semi-structured data), MonetDB/RAM (information retrieval and sciences), MonetDB/Armada (evolving databases), MonetDB/SkyServer (astronomy), and MonetDB/DataCell (streaming). After this paper session, multiple team members wrote papers on applying database cracking ideas to other areas (e.g., spatial indexing, index optimization). Not long after, I attended SIGMOD 2009—participating in the first SIGMOD programming contest—in Providence, Rhode Island. At this conference, Milena, Martin, Niels, and Romulo received a best-paper runner-up for another influential paper on "An architecture for recycling intermediates in a column-store", and without knowing it, I was sitting next to Milena during the business meeting where they got the recognition. During my later time at IBM Research – Almaden, I also worked with Romulo who joined our larger data management group as a postdoc.

Apart from these anecdotes, the database cracking paper and larger MonetDB eco-system had a tremendous impact on me. First, I fully adopted a system-oriented research philosophy of building actual open-source systems (currently Apache SystemDS[2] and DAPHNE[3]) and integrating all our research into these umbrella systems, which provides grounding and a deeper understanding of existing trade-offs. This approach led to fewer but, as I believe, much better papers. Second, several of our research sub-projects were inspired by ideas from MonetDB. Examples are compressed linear algebra (applying column compression to numeric matrices), optimizing operator fusion plans (tuned vectorized execution for operator DAGs), and lineage-based reuse (recycling intermediates in ML systems). Third, in my data management courses, we still cover database cracking—typically in lectures on physical design—as a beautifully simple yet effective idea with connections to other techniques such as indexing, partitioning and partition pruning, as well as materialized views and result caching. Fourth, and finally, I largely agree with Martin on how to assemble teams for system-oriented research. After Martin's too early death in 2022, CIDR 2023 held a great memorial sharing a variety of stories. Having played recreational soccer for 25+ years, I unknowingly used the same metaphor as Martin of assembling a research team like a soccer team with people of diverse backgrounds and skills (goalkeeper, defenders, forwards). So RIP (rest in peace) Martin Kersten, and thank you for inspiring generations of database researchers.

[2]https://systemds.apache.org/
[3]https://github.com/daphne-eu/daphne

Matteo Interlandi
Microsoft Gray Systems Lab, USA
matteo.interlandi@microsoft.com

Thomas Neumann.
Efficiently Compiling Efficient Query Plans for Modern Hardware.
In Proceedings of the VLDB Endowment, Volume 4, Issue 9, pages 539–550, 2011.

When I was asked to write about a paper that had a significant impact on me, one immediately springs to mind. While other papers had a more substantial influence on my PhD studies, this particular one not only shaped my view of database Query Processing (QP), but also consistently resurfaced over the years even after I graduated.

This paper, which won the Test of Time award in 2021, needs no introduction. It pioneered compiler-based QP, a paradigm shift from the traditional interpreted processing model for queries. Traditionally, databases have employed an interpreter model where single tuples or vectors of tuples are pulled from downstream data sources or operators using an iterator. This paper introduced several contributions that I continuously re-encountered in my life as a researcher.

Firstly, it proposed the division of query plans for execution into pipelines, identified by pipeline-breaking operations such as hash table creation. This method allows for more efficient processing and better utilization of modern hardware, while simultaneously blurring the boundaries between operators. Secondly, it flipped the conventional pull-based iterator model into a push-based execution. This approach enhances data flow and boosts the efficiency of query processing. Lastly, it advocated the use of compilers, like LLVM, to generate efficient code instead of relying on interpretation. This has paved the way for new possibilities in optimiz-

ing database performance and has established a new benchmark in the field.

The influence of this paper extends beyond its initial publication, inspiring numerous follow-up works and stimulating discussions within the database community. For instance, several subsequent works have focused on enhancing the approach by finding ways to reduce the compilation overheads. There has been a continuous debate on whether a compiler approach surpasses a vectorized one. Both sides have significant proponents, and this discussion has led to a deeper understanding of the strengths and weaknesses of each approach. Some more recent works have attempted to amalgamate the best of both worlds by integrating vectorized execution with compilation. For example, the latest paper from Wagner et al., presented at ICDE 2024 [1], introduces an interesting incremental compilation framework. Lastly, the definition of "modern hardware" has evolved since 2011. Lately, we are seeing a significant trend towards hardware specialization. The lessons from this paper remain not only relevant but also applicable and valuable in the context of these specialized hardware environments. For instance, I have spent most of my time in the last few years on understanding and implementing QP on GPUs. And as far as I know, compilation for QP on GPUs is still not mainstream, although possible [2].

In conclusion, the influence of this paper on both my personal development and the broader database community extends far beyond its original contributions. It has ignited new research directions and ongoing discussions that continue to shape the field of database query processing. The paper's enduring relevance and its significant role in advancing database technologies are a testament to its impact.

[1] Benjamin Wagner, Andre Kohn, Peter Boncz, and Viktor Leis. "Incremental Fusion: Unifying Compiled and Vectorized Query Execution." ICDE 2024.

[2] Wei Cui, Qianxi Zhang, Spyros Blanas, Jesús Camacho-Rodríguez, Brandon Haynes, Yinan Li, Ravi Ramamurthy, Peng Cheng, Rathijit Sen, and Matteo Interlandi. "Query Processing on Gaming Consoles." DaMoN 2023.

Theodoros Rekatsinas
Apple
trekatsinas@apple.com

Nilesh Dalvi and Dan Suciu.
Efficient Query Evaluation on Probabilistic Databases.
In Proceedings of the International Conference on Very Large Databases (VLDB), pages 864–875, 2004.

Thank you Pınar for the invitation to contribute to this column. As researchers, we seek the gratification of solving hard, technical problems and sharing the stories of our own achievements. It is a great opportunity to switch focus and, instead, reflect on how someone's hard work and research has shaped your own career.

I first read this paper in 2010, when I started my PhD and I was starting my research on probabilistic inference and reasoning. While many papers have shaped my research, I can confidently say that no other paper has quite influenced my career as much as this quintessential probabilistic databases paper.

The paper was published in 2004[4], before the data deluge we are witnessing today and the ever increasing need to deal with noisy, erroneous, and uncertain data. The focus was on bridging the gap between information retrieval and databases and the goal was to design a database system to efficiently evaluate SQL queries with uncertain predicates (e.g., string-matching predicates in the presence of typos) and provide a ranking over the query results. The score used to rank the results was the probability of existence (or correctness as I always liked to think about it) associated with each query result. The paper extended traditional set semantics with probabilities to introduce the *possible world semantics* – think of a raw enumeration of every possible state that a database whose base tuples have a certain probability of existence can take – and presented a new dichotomy: for many practical queries the probabilities associated with the results tuples can be computed in polynomial time, but there are some queries, which admit no efficient solution and belong to the class of #P-complete problems, i.e., really hard counting problems. Beyond the impact that this result had later on the design of efficient probabilistic inference systems, the proofs themselves are still relevant and instructive today.

For an immature PhD student knowing little about probabilistic inference; all I knew was that it is NP hard, this paper was a revelation. It introduced me to the world of efficient inference, the world of large-scale reasoning under uncertainty, and planted the seeds for my later interest in efficient machine

[4]... and received the VLDB 10-Year Best Paper award in 2014.

learning models. Moreover, the rigorous analysis and theoretical arguments of this work appealed to the engineer in me who always looks for correctness on the designs of proposed solutions. From that moment on I was convinced that I would always try to obtain such a deep understanding of a technical problem as the one demonstrated by the authors of this work. In retrospect, it was the perfect paper for me and I highly recommend that anyone interested in large-scale probabilistic reasoning reads this paper.

Madelon Hulsebos
University of California, Berkeley, CA, USA
madelon@berkeley.edu

Michael J. Cafarella, Alon Halevy, Zhe Daisy Wang, Eugene Wu, and Yang Zhang.

WebTables: Exploring the Power of Tables on the Web.

In Proceedings of the VLDB Endowment, Volume 1, Issue 1, pages 538–549, 2008.

In this paper, Cafarella et al. construct a large dataset of tables collected from web pages and describe a search system to navigate such tables at scale. The authors also introduce the Attribute Correlation Statistics Database (ACSDb), which they analyze and showcase in applications such as schema completion and attribute synonym identification. It has received VLDB's 10-year Test-of-Time award, illustrating its high and broad impact, as also described in the paper "Ten Years of WebTables" in 2018.

While data management research is typically concentrated on systems and algorithms for storing and processing data, less attention is given to the content of the data itself. Large collections of data objects help us better understand what data we are actually dealing with, and anticipate it. WebTables is no different and helps us better understand the data stored in (web) tables. I learned about the WebTables project while helping out with a study comparing the effectiveness of data visualizations for synthetic data versus real-world data. Turns out, synthetic tables are nothing like real-world tables. From that point, I have become fascinated with the nature of real-world tables – the variation in shapes, sizes, structure, semantics, messiness. Especially during my research on tables stored in CSVs: it is wild to see what people store in CSVs, and how this stretches our assumptions.

Beyond data insights, this research was instrumental in surfacing tables from the web through Google search and unlocking advancements in data management research (e.g. data integration), as highlighted in the "Ten Years of WebTables" paper. But to me personally, this paper has been important because large-scale datasets are essential in training machine learning (ML) models, and so has WebTables[5] been important in the development of ML models for structured data. These tables actually powered my first neural models for semantic table understanding, the starting point of my PhD.

The WebTables paper illustrates the value of patterns across tables by leveraging correlations among attributes in the millions of tables in the ACSDb database for, for example, automated schema completion. A similar idea underpins the GloVe word vectors, where word co-occurrence matrices are extracted from a large collection of texts, and used to obtain semantic word vectors through matrix factorization. A few years later, the transformer architecture fueled the training of token-level embeddings, and well, the rest is history. Recent research in data management, including my own, now intends to extend these capabilities to structured data.

Finally, this paper is also one of the early examples I know of a "dataset paper" that shows the value of constructing, documenting, and analyzing a dataset at scale. 15 years later, we even have tracks at VLDB and NeurIPS dedicated to the notion of dataset papers, and rightfully so. While the data management community, myself included, introduced new collections of tables over the past couple of years, I like to emphasize that we still have just scratched the surface of what is needed in terms of structured data to truly get to what is possible with AI. While WebTables unlocked many applications on web data, there is a pressing need for large-scale datasets representative of offline databases and interactions with this kind of data, which are harder to obtain at scale. So, I hope that my reflection on the importance of data can also serve as an influence on future contributions.

To close, I want to sincerely thank Pınar for inviting me to write this piece: it has been a long time since I have written a personal reflection on research. It reminds me how refreshing and liberating it is, complementary to writing research papers, while it is also fruitful for developing new questions and ideas.

[5]The WebTables corpus, unfortunately, was never published, but similar datasets were later published through Web Data Commons and Dresden Web Table Corpus.