Reminiscences on Influential Papers

This issue's contributors chose influential works that create bridges from other research communities (human-computer interaction, computer architecture, and programming models) to the data management community. Their write-ups highlight the importance of reaching across fields without forgetting the core. 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

Sourav S Bhowmick
Nanyang Technological University, Singapore
assourav@ntu.edu.sg

Don Chamberlin in his recent keynote talk at SIGMOD 2023 recalled that one of the goals behind the design of SQL was to make it *"easier to swallow by ordinary people"* and *"easy to understand [..] without any special training"*[1]. Since then 50 years have gone by and SQL has become wildly successful in the corporate world. However, during this time it has also morphed into a query language with many complex query semantics and features. While powerful, research by the Computer Education community has shown that SQL is difficult to understand, learn, and use, diverging from its original goal w.r.t. ease of use. Indeed, one needs specialized training to use SQL and, in fact, this holds true for any declarative query language in the market.

Two decades ago, fresh out from graduate school, I observed that research in data management primarily focused on data structures, algorithms, and performance. Although database usability research started 40 years ago, scant attention was paid to it in practice. Hence, in addition to these traditional issues, I started working on the usability aspects of querying databases as envisioned by Don Chamberlin.

Given that the topic of usability has a strong nexus with human-computer interaction (HCI), I will select two work that influenced my career in a fundamental way - one from the HCI community and another from our Data Management community. Furthermore, I am influenced more by novel, visionary ideas than some specific techniques. So I will diverge from the past authors of this column and select work that are not considered as traditional technical papers - one is a visionary book that influenced me to work on problems that are centered around users and another is a keynote paper that not only influenced my thinking on usability in the context of database systems but also motivated me to persist on my effort on this topic despite initial setbacks and a lack of sufficient attention from the Data Management community.

Ben Shneiderman.
Leonardo's Laptop: Human Needs and the New Computing Technologies.
MIT Press, pages 1-269, 2003.

I bought a signed copy of this visionary, intellectually stimulating, and inspirational book from Ben in CIKM 2005 (Figure 1). It was early days when I was exploring usability and databases. In this book, he asserted that *"the old computing was about what computers could do; the new computing is about what users can do"* and used Leonardo Da Vinci as the inspirational muse to lay down the vision of new computing. Leonardo integrated art with science to serve a practical purpose and produce something that also pleased his patrons. For

[1] https://dl.acm.org/doi/10.1145/3555041.3589336

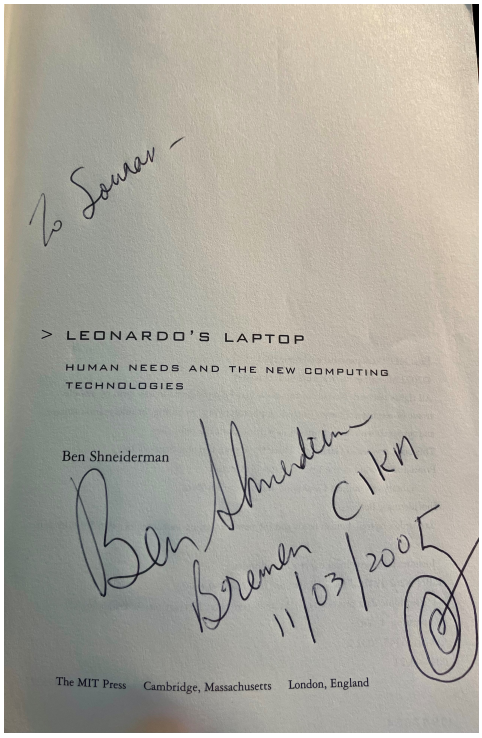

Figure 1: A signed copy of Leonardo's Laptop.

example, he painted *Mona Lisa*, to please her husband, Francesco del Giocondo, while demonstrating his visual insights and knowledge of geology, plants, and river ecology. So Ben posits that *"technical excellence must be in harmony with user needs"*. We should build products that are usable, useful, and enjoyable.

The book articulated two key steps for realizing the new computing paradigm. First, the promotion of good design w.r.t. the quality of user interfaces and the underlying infrastructure. Second, the promotion of inclusiveness by enabling diverse variety of users, young and old, novice and expert, able and disabled, which Ben referred to as *"universal usability"*. The book also proposed applications of new computing in education, medicine, business, and government.

Back then data management research was primarily about old computing - devising novel data structures and algorithms to demonstrate efficiency and scalability of a software or a technique. Ben's book inspired me to explore about the new computing paradigm in the context of databases. How do we design quality query interfaces and infrastructure to support diverse database user needs? My research on blending visual query formulation and query processing, plug-and-play visual interfaces, user-friendly query visualization abstraction, and understanding and quantifying aesthetics of visual query interfaces are all inspired by the vision of new computing. The vision of universal usability has recently inspired me to explore technologies that can enable learners, young and old, able and disabled, to learn about the topic of relational query processing. This group of users has received scant attention from the Data Management community as research and products primarily target corporate users and developers. It is worth noting that the push for universal usability of data management tools *"makes good business sense because it creates larger audiences for commerce, entertainment, and education"*.

H. V. Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu.

Making Database Systems Usable.

In Proceedings of ACM SIGMOD, pages 13-24, 2007.

This paper is an excerpt from the excellent keynote talk by Jag in SIGMOD 2007, which I attended. As remarked earlier, at that time, the Data Management community primarily focused their attention on data structures, algorithms, and performance issues but not on user-level database usability. This paper brought our attention to the fact that databases are *"hard to design, hard to modify, and hard to query"*. It systematically identifies the pain points encountered by end users and posit that the usability challenges in databases are much more than skin deep. Simply slapping a user-friendly visual query interface on top of a database system does not alleviate these challenges and called for rethinking the underlying architecture of the database system to address them. In particular, it presents the notion of a *presentation data model* as a distinct layer on top of the logical data model. It is envisioned to enable effective personalization and interaction with the database through direct manipulation.

I have used this paper as a mental template for addressing problems related to visual querying. For instance, several of my work focused on blending visual query formulation and processing by exploiting the latency offered by visual query interfaces, which demands rethinking of the underlying query processing component in a visual querying environment. Similarly, our notion of plug-and-play visual query interfaces is inspired by the need of different presentation data model for different users for different data sources to facilitate more effective and

efficient visual querying.

Last but not the least, I believe this paper also played a pivotal role in putting the attention of our community back to database usability that was lost for decades. Prior to 2007, I had a hard time publishing papers on usability and data management in the Data Management venues. In fact, I barely squeezed in a short paper in ICDE 2006. In several major venues all the three reviewers would suggest that I should submit these work to HCI venues - as if usability of data systems is not our business! This bleak landscape changed since 2010. Since then, I observed more reviewers in our major venues were open to such work and as a result I was able to publish regularly on this topic. Prior to 2007, often 3 out of 3 reviewers mentioned that usability is not relevant to data management. Nowadays, sometimes it is 1 out of 3 - I believe that is progress! Looking back, I strongly believe that this keynote played a significant role in changing perception and emphasizing *"usability of a database as important as its capability"*.

Carsten Binnig
TU Darmstadt & DFKI, Germany
carsten.binnig@tu-darmstadt.de

Jun Rao and Kenneth A. Ross.
Cache Conscious Indexing for Decision-Support in Main Memory.
In Proceedings of the International Conference on Very Large Data Bases (VLDB), pages 78-89, 1999.

First of all, I would like to thank Pınar for running this column over the last few years. The fact why I like this column is that I am not only learning a lot about past papers of our community but even more I like the fact that the column often tells a very personal story of why a paper was influential to the career of an individual. Moreover, the request of Pınar started my own thought process of which paper I should choose which is an interesting exercise on its own.

To answer this question, I was sitting down and my mind was playing ping-pong with many different ideas. I was happy to discover that there was a way too long list of papers that I had read over my career which inspired me. The bad point was, I had to pick ONE. In the end, I decided to choose the paper "Cache Conscious Indexing for Decision-Support in Main Memory" which appeared in VLDB'99 from Jun Rao and Kenneth A. Ross. I chose the paper due to two reasons.

The first reason is maybe the more obvious one since it was a pick that influenced my career. When I was a PhD, I started to work with Donald Kossmann on database testing which was a highly important topic but at that time not at the core of the database community. After my PhD, I really wanted to shift towards a topic that was more at the core of our community. Around 2008, I started to work on ideas related to leveraging modern hardware for in-memory databases in the context of my (industrial) work on SAP HANA. In this realm, I was reading many papers that contributed to this line of work.

In this body of work, the paper from Ken was starring out. For me personally, it is a "seminal" paper since it early on explored modern CPU architectures with multi-level caches for designing optimal memory-based index structures. The paper looked into the question of how to redesign tree-like in-memory index structures and make them cache-conscious for read-mainly scenarios. In the years after, we saw many papers along similar lines. The paper itself combines a set of simple yet elegant ideas to make an in-memory index cache conscious such as avoiding pointers and using offset computation for the index traversal. As a consequence, data can be kept more densely in the index nodes without "wasting" space for large pointers to child nodes thus making the index cache-friendly.

The result of the paper is an index structure called CSSTree (Cache Sensitive Search Tree) which had later on been followed by the CSB+-tree [1] (Cache Sensitive B^+-Tree) that supports also more write-heavy workloads without giving up the cache-optimal properties. Beyond these topics, Ken has also contributed significantly to a broad range of other important topics regarding databases on modern hardware (e.g., SIMD and GPUs). Today, the whole line of databases on modern hardware is still highly active, and new debates are needed in the context of the cloud given the challenges that hardware is scaling slower than data.

A second reason why I chose the paper is actually because I highly value Ken as a researcher. I only had a few times where I could personally interact with him. One occasion to meet and discuss with him was his visit to Brown University when I was there from 2014 to 2017. These few moments were sufficient to impress me deeply. Firstly, Ken is a very modest and quiet, but extremely knowledgeable person. Secondly, while Ken is known for his solid work on developing database algorithms and data structures for modern hardware, I learned

that he also has a second, completely different area of work, which is not even in computer science, but in medicine. Personally, I find this strong ability to research in-depth and breadth extremely fascinating.

[1] Jun Rao and Kenneth A. Ross. "Making B^+-Trees Cache Conscious in Main Memory." In Proceedings of ACM SIGMOD, pages 475-486, 2000.

Peter Alvaro
University of California, Santa Cruz, CA, USA
palvaro@ucsc.edu

Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek.

The Click Modular Router.

In ACM Transactions on Computer Systems, Volume 18, Issue 3, pages 263–297, 2000.[2]

I first read the Click paper in 2009 in Randy Katz's networking seminar at UC Berkeley. At that time, I was a second-year graduate student in Joe Hellerstein's lab, studying data management systems but interested in distributed systems and languages. Riding the tailwind of Boon Loo's dissertation work on "Declarative Networking," I was thinking a lot about transplanting other database ideas (in particular, query languages and dataflow-based execution) into new domains. I was (perhaps narrowly) focused on the idea of "declarative" programming, but I struggled to reconcile the ideology of "what not how" with concerns such as modularity, reuse, and performance. For example, small queries can be quite beautiful, but they become unwieldy as their complexity increases - and programmers tend avoid them because they prevent low-level control of execution.

During my first reading, the Click paper reaffirmed the view of the world I share with a small number of colleagues: that query processing is a rich way to model general computation. As Click shows, routing is a special case of query execution. With the right set of tweaks and extensions, the zoo of routing and switching protocols that we had been studying in class decompose into a set of simple operators (called "elements") that process tuples (i.e., packets) one-at-a-time, alongside a collection of rules about how operators may be composed. An instantiation of a router is hence a dataflow graph that reacts to the insertion of tuples (e.g., those arriving via an external network interface) by outputting tuples (e.g., via an outgoing interface).

Reading more deeply, the paper began to upset my understanding of the role of abstraction in systems programming and its relationship with reuse and performance, teaching me a number of lasting technical lessons. Critical to both the generality and degree of reuse in Click is its ability to combine operators whose interfaces are "pull" (as operators in traditional query processing are, exposing iterators) and those that are "push" (as operators in streaming databases are, fronted by queues). This generality was required to model interfacing with the boundaries of systems where, for example, packets arrive at times outside of our control, and sending packets requires waiting for a device to be ready. Rather than being simply a workaround to handle the edges of the system, however, it makes the programming model surprisingly powerful. The "trick" that makes permits programmers to intermix push and pull operators (but only in ways that are "type-safe," since it does not make sense to directly compose an operator that wants to push to its outputs with an operator that wants to pull from its inputs) is the explicit representation of queues. Rather than hiding them inside streaming-style operators, programmers choose where queues are placed, which influences when scheduling decisions are made. By doing so, Click permits the implementation of operators that (like Eddies [1]) perform scheduling itself.

The Click programming model is not declarative in the sense typically meant by our community. There is no calculus or algebra that gives rise to a space of plans to automatically cost and search: programmers of Click perform query planning by hand. Instead it offers what Kohler[3] called "picture-frame declarativity," allowing implementers to operate freely on both sides of the abstraction. Implementing a new routing protocol is a matter of creating a new query, and can be performed visually, at the level of logical dataflow. Allowing programmers to explicitly place queues gives them control over when scheduling decisions are made, without requiring them to worry about how. Implementing completely new functionality or scheduling policy requires peeling back the abstraction and implementing a new operator in low-level code. Hence Click elements "frame" imperative processing, allowing programmers to enjoy the benefits of high-level programming while maintaining control of the system. Click is fast - faster than I knew a high-level programming model could ever be.

[2]Extends the SOSP 1999 paper from the same authors with the same title.

[3]in a separate talk

It is hard to measure the impact of Click on my own thinking and research. The explicit representation of queues as programmatic constructs rather than hidden plumbing became a key feature of the Dedalus language [2], which became the foundation of my thesis work, some of which departed from pure query languages to target streaming dataflow systems. I have also lost count of how many times I have read the paper. Each time, I find something new. Every database researcher should read it often.

[1] Ron Avnur Joseph M. Hellerstein. "Eddies: Continuously Adaptive Query Processing." In Proceedings of ACM SIGMOD, pages 261-272, 2000.

[2] Peter Alvaro, William R. Marczak, Neil Conway, Joseph M. Hellerstein, David Maier, and Russell Sears. "Dedalus: Datalog in Time and Space." In Datalog Workshop, pages 262-281, 2010.