Reminiscences on Influential Papers

This issue's contributors dive into a history of concurrency control and storage hierarchy for database and data-intensive 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

Goetz Graefe
Google - Madison, WI, US
goetzg@google.com

H. T. Kung and John T. Robinson.
On optimistic methods for concurrency control.
In ACM Transactions on Database Systems (TODS), Volume 6, Issue 2, pages 213-226, 1981.

This fall I will teach "topics in database systems" at UW-Madison. When I took this course as a first-year graduate student four decades ago, David DeWitt had his students read two then-recent papers that he correctly expected to become classics: "Access Path Selection..." [29] and "On Optimistic... Concurrency Control" [21]. Of these, the former still strikes me as brilliant for constructing join plans using dynamic programming, for its focus on "interesting orderings" in complex-object assembly (joining and grouping on, effectively, object identifiers), and for its clear-eyed perspective on the perils of cardinality estimation. Kooi's impressive thesis soon addressed some of these perils with multiple kinds of histograms [20].

The latter one of these two classic papers resonates with the saying that "it's easier to ask for forgiveness than for permission" but clearly there is much more to it. Many people, it sometimes seems, remember the paper's title but have not really studied the text. They may rely on the title's promise but ignore multiple pitfalls. Many pitfalls have been called out by Theo Härder and C. Mohan [19, 25] – below is one further perspective.

First, the original design of optimistic concurrency control assumes page-level concurrency control. This was competitive at the time, e.g., with page-level locking in System R [8], but it is not competitive with record-level locking [22, 24, 27]. The introduction of record-level concurrency required new techniques for logging and recovery, e.g., compensation or update-back log records [14, 26], and for concurrency control, e.g., locking gaps between index entries or between key values [13]. More specifically, equivalence to a serial execution requires repeatable read plus phantom protection [9]. For example, in an ordered database index, a search with an empty result set requires concurrency control for gaps between key values, as does a range scan.

Second, optimistic concurrency control requires transaction-private update buffers, which are a form of multi-version storage. A fair comparison of optimistic and pessimistic concurrency control (i.e., end-of-transaction validation versus pre-access locking) must consider locking in a multi-version context, not in single-version storage. In a multi-version store, locking can ignore read-write (rw-) conflicts and wr-conflicts until the updating transaction attempts to commit. Detecting ww-conflicts upon the first conflicting access prevents conflicting writers and doomed transactions, saves wasted work and its rollback, ensures at most one uncommitted version (per granule of concurrency control), and permits creating new versions directly in the database (or its shared buffer pool) - all with the hope and expectation that rw- and wr-conflicts rarely delay transaction commit. This optimism is the foundation and essence of deferred lock enforcement [12].

Third, validation and write phases together must be atomic, as described in the original 1981 paper. For transactional durability [18], the write phase is more than draining a transaction's private update buffer into the system's shared buffer pool: each transaction's write phase must include forcing the commit log record to stable storage. With fairly long atomic validation-and-write phases, shared database servers require concurrent validation. Transactions validating concurrently must share information about their read- and write-sets. The required data structure manages, in a thread-safe manner, a many-to-many relationship between transactions and database objects - very much like a traditional lock manager. In this way of thinking, optimistic concurrency control is a form of deferred lock acquisition [12]. Its expectations for performance, scalability, conflicts, and system throughput are near but not quite equal to deferred lock enforcement.

Fourth, participants of distributed transactions transition from optimism to pessimism when they vote in a two-phase commit. More accurately, they pledge to wait for and to implement the coordinator's global commit decision. This pledge must be firm; "asking for forgiveness" is no longer an option so that "asking for permission" is required, i.e., lock acquisition. Again, optimistic concurrency control is a form of deferred lock acquisition. Participants must retain both shared and exclusive locks until they receive the global commit decision [11]. Increasing concurrency during the commit process beyond the traditional level has been described as controlled lock violation [15]. Other transactions may read and modify recent updates but they must not commit until the commit logic of the earlier updater completes successfully. In a sense, controlled lock violation is an optimistic technique within pessimistic concurrency control.

In summary, optimism in concurrency control is found in both deferred lock enforcement and controlled lock violation, which are advanced locking techniques with optimism, with perfect equivalence to a serial execution, with early detection of doomed transactions, with no need or overhead for transaction-private update buffers, and with fewer false conflicts than traditional optimistic or pessimistic concurrency control. Even if these locking techniques supplant the original design for optimism in concurrency control, they are continuations and refinements of the classic 1981 paper on optimistic concurrency control.

Raja Appuswamy
EURECOM, France
`raja.appuswamy@eurecom.fr`

Jim Gray and Franco Putzolu.
The 5 minute rule for trading memory for disc accesses and the 10 byte rule for trading memory for CPU time.
In Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, pages 395–398, 1987.

In 1998, Richard Snodgrass edited a new column in SIGMOD Record entitled "Reminiscences on Influential Papers". In the introduction to this column, he wrote "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." When Pınar reached out to me for a contribution, for which I am very thankful, I instantly knew which paper I had to choose.

Over the years of my research career, I have worked on several systems that rely on caching at various levels of the memory hierarchy. I had always viewed caching from an algorithmic point of view. The most interesting part in caching, I had thought, was the design of clever algorithms that could identify the optimal data items to cache. I was fascinated by the simplicity of design and the incredible potential of the Adaptive Replacement Cache [23] when it was introduced by IBM and adopted by ZFS, and toyed around with extending it to a multi-level memory hierarchy [4] during my PhD years when I designed the Loris file system for MINIX 3. And then, I read the five-minute rule paper, and as Richard said in 1998, it definitely triggered a radical restructuring of my mental model.

Jim Gray and Franco Putzolu introduced the five-minute rule in 1987 [17]. In a surprising twist, Jim and Franco turned the caching problem into an economic one. They argued that bringing disk-resident data to memory is not only time consuming, but is also economically expensive. At the time, the Tandem disk cost $15K and could provide 15 accesses per second. Thus, it cost $1K per access per second. Add in support for CPUs to handle interrupts and channel/controller costs to manage the disks, you get to $2K per access per second. A Kilobyte of main memory, in contrast, costs only $5. So, assuming you have 1KB of data that you access every second, if you "rent" 1KB of memory and keep your data in it, you can save $2K worth of disk accesses–a fantastic bargain. If you access the 1KB every 10 seconds, that is 0.1 accesses/second, you can still save $200 of disk access by spending $5 to rent memory. Continuing this argument, it turns out that accessing data once every 400 seconds becomes the break-even point, when accessing data from disk is as expensive as renting memory. Restating Jim and Franco, "as 400 seconds is about five minutes, hence the name five-minute rule".

I love this paper for several reasons. First, it is written by one of the titans of computing I respect deeply–Jim Gray. My first encounter with Jim's work was the seminal tome on transaction processing. While writing this article, I found out that Jim himself had contributed an article to this very column back in 1998, where he describes the story of how TPC-A and TPC-D benchmarks came about, and why SIGMOD sort trophies used to be awarded on April Fools Day! His description of the memory hierarchy in the popular "How Far Away is the Data" example is yet another masterpiece like the five-minute rule that breaks down a complex topic into a fun, easy-to-understand exposition that I routinely use in my lectures.

Second, in the paper, Jim and Franco provide a quantitative framework for making informed decisions about memory and disk usage. Using a case study, they even show how the five-minute rule was used in practice to help a designer choose an appropriate configuration (hybrid memory-disk versus all-in-memory) for a database server. Storage technology and economics have changed radically since the rule was introduced in 1987. Yet, the framework proposed in the original paper is general enough to apply to various levels of today's three-tier (performance with DRAM/SSD/PMEM, capacity with HDD, and archival with tape) memory hierarchy. Thus, the rule has been revisited three times, in 1997 [16], 2008 [10], and 2019 [3], with each iteration providing surprising insights into the behavior of "media-du-jour", be it NAND flash in 2008 or persistent memory in 2019.

Third, this paper really made me think about how other aspects of data management in addition to caching would change if we considered economics, and cost of data engineering, as first class citizens instead of performance. This made me focus my research over the past few years on the lower two tiers of the storage hierarchy. Focusing on the capacity tier, I was inspired by Pelican [5]–a rack-scale cold storage system that packed thousands of HDDs in a single rack that was right-provisioned to service only a fraction of HDDs running simultaneously. Pelican provided accesses latencies in seconds, between HDD and tape, and provided near-line data access for "cold", or infrequently accessed, data items. Applying the five-minute rule framework to such cold storage devices (CSD) showed us that it might be economically beneficial to leave cold data in such devices and perform query execution directly on the CSD rather than moving it to HDD. This motivated us to develop the Skipper query-processing framework [7]. Focusing on the archival tier, I came across the work on database preservation by the Digital Preservation community that made me realize that long-term archival of databases is an incredibly challenging problem, both from technical and economic points of view. I wrote about some of these challenges in the SIGMOD Record Brainstorming article [1] "Towards Passive, Migration-Free, Standardized, Long-Term Database Archival". Around this time, storage researchers were starting to investigate radically new archival media, and synthetic DNA was one such media that got a lot of attention [6]. This triggered our work on project OligoArchive [2], where we investigated the use of DNA for long-term database archival.

Fourth, the paper's history teaches an important life lesson for academics: rejection, just like awards in some cases, is not a concrete predictor of impact. Despite being such an insightful piece of work, this paper was rejected during its first round of submission. Quoting David Patterson [28], "Jim Gray wrote to Jim Larus: The B-tree paper was rejected at first. The Transaction paper was rejected at first. The data cube paper was rejected at first. The five-minute rule paper was rejected at first. But linear extensions of previous work get accepted. So, resubmit! PLEASE!!." I have had my papers rejected enough times now to know and tell my students that it is a part of academic life. But using THE five-minute rule from a titan such as Jim Gray as an example of rejection usually sends home the message right away.

1. REFERENCES

[1] Raja Appuswamy. Towards Passive, Migration-Free, Standardized, Long-Term Database Archival. *SIGMOD Rec.*, 51(2):61–62, jul 2022.

[2] Raja Appuswamy, Kevin Le Brigand, Pascal Barbry, Marc Antonini, Olivier Madderson, Paul S. Freemont, James McDonald, and Thomas Heinis. OligoArchive: Using DNA in the DBMS storage hierarchy. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA,*

January 13-16, 2019, Online Proceedings. www.cidrdb.org, 2019.

[3] Raja Appuswamy, Goetz Graefe, Renata Borovica-Gajić, and Anastasia Ailamaki. The five-minute rule 30 years later and its impact on the storage hierarchy. *Commun. ACM*, 62(11):114–120, oct 2019.

[4] Raja Appuswamy, David C. van Moolenbroek, and Andrew S. Tanenbaum. Cache, cache everywhere, flushing all hits down the sink: On exclusivity in multilevel, hybrid caches. In *2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–14, 2013.

[5] Shobana Balakrishnan, Richard Black, Austin Donnelly, Paul England, Adam Glass, Dave Harper, Sergey Legtchenko, Aaron Ogus, Eric Peterson, and Antony Rowstron. Pelican: A building block for exascale cold data storage. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, page 351–365, USA, 2014. USENIX Association.

[6] James Bornholt, Randolph Lopez, Douglas M. Carmean, Luis Ceze, Georg Seelig, and Karin Strauss. A DNA-Based Archival Storage System. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, page 637–649, New York, NY, USA, 2016. Association for Computing Machinery.

[7] Renata Borovica-Gajić, Raja Appuswamy, and Anastasia Ailamaki. Cheap data analytics using cold storage devices. *Proc. VLDB Endow.*, 9(12):1029–1040, aug 2016.

[8] Donald D. Chamberlin, Morton M. Astrahan, Michael W. Blasgen, James N. Gray, W. Frank King, Bruce G. Lindsay, Raymond Lorie, James W. Mehl, Thomas G. Price, Franco Putzolu, Patricia Griffiths Selinger, Mario Schkolnick, Donald R. Slutz, Irving L. Traiger, Bradford W. Wade, and Robert A. Yost. A history and evaluation of System R. *Commun. ACM*, 24(10):632–646, oct 1981.

[9] K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger. The notions of consistency and predicate locks in a database system. *Commun. ACM*, 19(11):624–633, nov 1976.

[10] Goetz Graefe. The five-minute rule 20 years later (and how flash memory changes the rules). *Commun. ACM*, 52(7):48–59, jul 2009.

[11] Goetz Graefe. *On Transactional Concurrency Control: A Problem in Two-Phase Commit*, pages 321–326. Springer International Publishing, Cham, 2019.

[12] Goetz Graefe. *On Transactional Concurrency Control: Deferred Lock Enforcement*, pages 327–365. Springer International Publishing, Cham, 2019.

[13] Goetz Graefe. *On Transactional Concurrency Control: Orthogonal Key-Value Locking*, pages 159–210. Springer International Publishing, Cham, 2019.

[14] Goetz Graefe, Wey Guy, and Caetano Sauer. *Instant Recovery with Write-Ahead Logging: Page Repair, System Restart, Media Restore, and System Failover, Second Edition*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2016.

[15] Goetz Graefe, Mark Lillibridge, Harumi Kuno, Joseph Tucek, and Alistair Veitch. Controlled lock violation. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, page 85–96, New York, NY, USA, 2013. Association for Computing Machinery.

[16] Jim Gray and Goetz Graefe. The five-minute rule ten years later, and other computer storage rules of thumb. *SIGMOD Rec.*, 26(4):63–68, dec 1997.

[17] Jim Gray and Franco Putzolu. The 5 minute rule for trading memory for disc accesses and the 10 byte rule for trading memory for CPU time. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, SIGMOD '87, page 395–398, New York, NY, USA, 1987. Association for Computing Machinery.

[18] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317, dec 1983.

[19] Theo Härder. Observations on optimistic concurrency control schemes. *Information Systems*, 9(2):111–120, 1984.

[20] Robert Philip Kooi. *The optimization of queries in relational databases*. PhD thesis, USA, 1980. AAI8109596.

[21] H. T. Kung and John T. Robinson. On optimistic methods for concurrency control. *ACM Trans. Database Syst.*, 6(2):213–226, jun 1981.

[22] David B. Lomet. Key Range Locking Strategies for Improved Concurrency. In *Proceedings of the 19th International Conference on Very Large Data Bases*, VLDB '93, page 655–664, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[23] Nimrod Megiddo and Dharmendra S. Modha. ARC: A Self-Tuning, Low Overhead Replacement Cache. In *2nd USENIX Conference on File and Storage Technologies (FAST 03)*, San Francisco, CA, March 2003. USENIX Association.

[24] C. Mohan. ARIES/KVL: A key-value locking method for concurrency control of multiaction transactions operating on B-tree indexes. In *Proceedings of the Sixteenth International Conference on Very Large Databases*, page 392–405, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[25] C. Mohan. Less optimism about optimistic concurrency control. In *[1992 Proceedings] Second International Workshop on Research Issues on Data Engineering: Transaction and Query Processing*, pages 199–204, 1992.

[26] C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Trans. Database Syst.*, 17(1):94–162, mar 1992.

[27] C. Mohan and Frank Levine. ARIES/IM: An efficient and high concurrency index management method using write-ahead logging. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, SIGMOD '92, page 371–380, New York, NY, USA, 1992. Association for Computing Machinery.

[28] David A. Patterson. How to Build a Bad Research Center. Technical Report UCB/EECS-2013-123, Jun 2013.

[29] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, page 23–34, New York, NY, USA, 1979. Association for Computing Machinery.