DOI:10.1145/1953122.1953124

Why Concurrent Objects Are Recurrently Complicated

NIR SHAVIT'S ARTICLE "Data Structures in the Multi-core Age" (Mar. 2011) was a thrill to read; so, too, was the timely Viewpoint "Managing Time" by Peter J. Denning, the article "Understanding Scam Victims: Seven Principles for Systems Security" by Frank Stajano and Paul Wilson, the Technical Perspective "VL2" by Jennifer Rexford, and the article "VL2: A Scalable and Flexible Data Center Network" by Albert Greenberg et al.

I was especially intrigued by Shavit's exposition on concurrent objects. I first encountered lock-free and wait-free protocols in a 1993 article by Maurice Herlihy,¹ using load-linked and store-conditional instructions. Thanks now to Shavit for explaining additional techniques, including exchangers and balancers.

Also important is to point out that Shavit's pseudocode for the lock-free stack included an unstated restriction—that a node, once popped, never goes back on the stack—the ABA problem; see the related *Communications* blog page <http://cacm.acm.org/magazines/2011/3/105308-data-structures-in-the-multicore-age/comments>, as well as Herlihy's and Shavit's 2008 book.²

The key point is that concurrent data structures are more difficult than they might first seem, as I discussed in my own 1995 article³ and that it is easy to introduce subtle bugs, especially when all important assumptions are not stated or enforced; in this case, that a node, once popped, never goes back on the stack.

Joseph P. Skudlarek, Lake Oswego, OR

References

1. Herlihy, M.P. A methodology for implementing highly concurrent data objects. *ACM Transactions on Programming Languages and Systems* (Nov. 1993).
2. Herlihy, M.P. and Shavit, N. *The Art of Multiprocessor Programming*. Morgan Kaufmann, San Mateo, CA, 2008.
3. Skudlarek, J.P. Notes on a methodology for implementing highly concurrent data objects. *ACM Transactions on Programming Languages and Systems* (Jan. 1995).

Origin of SQL

I would like to clarify Erik Meijer's and Gavin Bierman's statement concerning the origin of SQL in their article "A Co-Relational Model of Data for Large Shared Data Banks" (Apr. 2011) that said, "The landscape changed radically when Ted Codd proposed a new data model and a structured query language (SQL) based on the mathematical concepts of relations and foreign/primary key relationships." There were actually additional steps and additional people involved getting from Codd's influential 1970 *Communications* article "A Relational Model of Data for Large Shared Data Banks" (cited by Meijer and Bierman) to the SQL language. In it and in subsequent articles and papers, Codd discussed query languages based on both relational calculus and relational algebra. But it was Raymond Boyce and Donald Chamberlin who, after studying Codd's work, defined the SEQUEL language, later renamed SQL, as first documented by Chamberlin and Boyce.¹ For more on how Codd's ideas influenced Boyce and Chamberlin and the rest of the System R team, see Chamberlin.²

Paul McJones, Mountain View, CA

References

1. Chamberlin, D.D. and Boyce, R.F. SEQUEL: A structured English query language. In *Proceedings of the 1974 ACM SIGFIDET (Now Sigmod) Workshop on Data Description, Access and Control* (Ann Arbor, MI, May 1–3). ACM Press, New York, 1974, 249–264.
2. Chamberlin, D. Oral History, Paul McJones, interviewer. Computer History Museum, Mountain View, CA, July 21, 2009; <http://www.computerhistory.org/collections/accession/102702111>

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.

© 2011 ACM 0001-0782/11/06 \$10.00

ACM's *interactions* magazine explores critical relationships between experiences, people, and technology, showcasing emerging innovations and industry leaders from around the world across important applications of design thinking and the broadening field of the interaction design. Our readers represent a growing community of practice that is of increasing and vital global importance.

interactions
<http://www.acm.org/subscribe>