A Conversation with Margo Seltzer and Mike Olson

THE HISTORY OF BERKELEY DB

Kirk McKusick sat down with Margo Seltzer and Mike Olson to discuss the history of Berkeley DB, for which they won the ACM Software System Award in 2021. Kirk McKusick has spent his career as a BSD and FreeBSD developer. Margo Seltzer has spent her career as a professor of computer science and as an entrepreneur of database software companies. Mike Olson started his career as a software developer and later started and managed several open-source software companies. Berkeley DB is a production-quality, scalable, NoSQL, Open Source platform for embedded transactional data management.

Kirk McKusick Berkeley DB came out of the University of California at Berkeley Computer Systems Research Group's work to create a version of Unix unencumbered by AT&T's ownership rights to the original version of Unix. To do that, we needed a new kernel, written without using any of the AT&T code. We also needed all the applications and libraries that shipped with the operating system.

My colleague on the Berkeley BSD Project, Mike Karels, and I were in charge of getting a clean version of the kernel—that's another story! But Keith Bostic took on the task of getting all the apps and libraries done. He solicited volunteers for much of that work. I know he worked with

you two on that. Why don't you start the story there?

Margo Seltzer This started with a standard stupid grad-student trick. I had taken Mike Stonebraker's graduate database course at Berkeley, and I thought Litwin's extensible linear hashing was really cool. Keith [Bostic], recognizing "stupid grad-student syndrome," said, "Hey, how would you like to implement it? I need a replacement for ndbm, dbm, and hsearch." I innocently said, "Sure!"

He introduced me to Ozan Yigit, who had written gdbm, which was specifically an ndbm replacement. We brainstormed together and figured out a way to have a single hash package that would support both the persistent (n)dbm, as well as the in-memory hsearch replacement. That became what was known as, cleverly enough, hash.

Then, Keith's ulterior motive kicked in, which was that he really wanted a record package that he could put under a replacement vi, because vi was another big piece of userland that needed to be rewritten. He'd had it in his head that he wanted a record manager underneath it.

This required an interface that would let him access records by record number—that is, "I want line 59" returns the 59th line in a database, which for his purposes was a text file. He'd seen a Stonebraker paper with Heidi Stettner that showed how to implement this record-number interface on top of B-trees.[2]

And that's when Mike came along.

Mike Olson I was Margo's officemate at this point. We were both Stonebraker students. I was on the Postgres project, where I was responsible for a bunch of the storage code, including B-trees.

I'd already written B-tree implementations two or three times in my life, so when Keith began to pester me with this project, I refused. I had written B-trees over and over again, and it just seemed like pointless work to do it again.

At the time, I was convening a pretty regular Friday afternoon study group for the research group, which was code for, "Let's go down to the local brewpub and throw back some pints." Keith started crashing those and really leaning on me to do this.

Finally, to shut him up, I agreed that I would write the B-tree code. And since Margo was doing hash, we were a team.

I also had an ulterior motive: I was going to replace the B-tree code in Postgres with the Berkeley DB B-tree code, which was cleaner and better. I never got around to doing that work. I'm sorry about that, because I think it would have been an improvement for the Postgres code as well.

KM So, the two of you coded up a clean version of ndbm, no AT&T code involved. That got added to the Berkeley Software Distribution. And that's basically Berkeley DB today?

MS There's another chapter. Rewind to before I went to graduate school. I had worked at a company called Sequoia Systems, where we were building a transactional storage system to support C programs.

We had designed what we called Sequoia's Transaction-Oriented Record Manager, otherwise known as STORM. We started building it but never finished it. Both my boss and I left the company. So, in the back of my head, I had always had this idea that you could build a transaction-oriented record manager.

As another side project, I somehow convinced Mike that once he built the B-tree, we should make it transactional. We could take the hash and B-tree code, put them under a common API, and have this cool transactional thing.

That became libtp, which was another paper that Mike and I wrote.[1]

KM Was libtp part of the BSD releases as well? Did you ship that code?

MS No, the libtp code was what I like to refer to as a "graduate-student code." Keith never brought it up to production-code quality.

KM All right, now you have this graduate-student code that supports transactions, and some other code shipping with BSD. How did those two threads get woven together?

MO Let me give a little bit of background. Margo and Keith did almost all of the work on the software, starting in 1992 and continuing into the late 1990s. Berkeley DB, the DB 1.85 library, got shipped all over the planet with Berkeley Unix. It got used in a huge number of projects.

In particular, a group of researchers at the University of Michigan built an LDAP (Lightweight Directory Access Protocol) engine to look up records by key, over the network, in a really fast way. That eventually became OpenLDAP, and that's a key part of a lot of authentication systems today. They used Berkeley DB, and they did a really good job, except the DB 1.85 code didn't support transactions.

God forbid two people should do something at the same time, or your computer should go down in the middle of an update—your database could be corrupted.

Netscape (the company) wanted a directory server as

part of its product line. It hired the entire University of Michigan team into the company to bring along the code and get the work done.

I wasn't around at the time. I'll let Margo tell it from there.

MS The LDAP folks had seen the libtp paper, and they went looking for the code. They contacted Keith and me, asking, "Hey, we're using DB 1.85; we see this libtp. Where's the code?"

Netscape would pay us for it, but we would retain the rights to sell it to other people.

And we said, "Uhhhhh, graduate-student code; you don't really want to use that for production." And they said, "Well, what would it take to make the libtp code real?" Keith and I had been talking about building a production-quality transactional library for some time. We knew it was going to be a fair bit of work, and we had day jobs. Netscape said, "Well, you know, we'd pay money for it."

Keith and I thought, "Well that's novel. Who would have thought of that?"

We managed to structure a deal with Netscape such that we built out the production transactional version, Netscape would pay us for it, but we would retain the rights to sell it to other people.

We decided that we wanted to do the work, and if all we ever did was build this for Netscape—and they were happy with it—that would be fine.

But once we did build it, we figured we'd hang a shingle out. At that point that meant building a website and seeing what happened.

What happened was that lots of organizations had picked up the UMich LDAP server. Our first-generation customers were all the people who picked up that LDAP

server and wanted transactions in it.

That was how Sleepycat was formed. We're doing this deal with Netscape, and we wanted to make sure that we weren't, quite literally, "betting the house." We called a lawyer, who said, "Form a company."

MO Key plot point: In the middle of all this, Keith and Margo had gotten married. So, they owned a house that they wanted to protect.

I love this story. I always thought it was a fantastic judo move by the two of you. Essentially, Netscape provided the seed funding for the company. That let us create a business without taking any external investment. That money from Netscape was the seed funding that got the original product built. It wasn't a work for hire. Keith and Margo still owned the intellectual property rights.

It meant that we weren't dependent on Sand Hill Road venture capital for the entire life of the business.

KM Where did the Sleepycat name come from?

MS We called the lawyer and said, "Hey we need to start a company," and the lawyer said, "Well, you need a name." Fortunately, there was a cat sleeping on Keith's lap when he looked down.

We think that this is the same algorithm that was used to name Spider Plant Software, but they looked up instead of down.

MO I was off doing other things from 1993 to 1998. Honestly, I had forgotten about the libtp work until Margo reminded me after I quit a miserable job.

When I joined Sleepycat, my first proposal was that we rename the company. I was soundly voted down by my two colleagues.

KM Sleepycat used something called "dual licensing" for Berkeley DB. Take us through that.

MS Keith was adamant from day one that we had to keep the open-source license, because people had started using it and we weren't going to turn our backs on them.

To be perfectly blunt, Keith spent a ton of time working on a dual license that would be palatable for the FSF (Free Software Foundation), and in particular, for glibc, because the new glibc was using Berkeley DB. We didn't want to lose that distribution, since that gave us literally millions of seats.

So, it was basically Keith and the FSF figuring out a license that would allow us to build a business and make revenue, and allow us to be open source and continue as infrastructure inside a huge number of other projects.

MO I think it was a really impressive intellectual feat. Dual licensing, at least open-source dual licensing, was invented by Keith Bostic. Others have since chosen to use that technique to monetize open-source software.

Nobody did it, to my knowledge, before we did.

We worked very hard to stay on the good side of the Debian software distribution, for example, the most closely aligned with the FSF of all of the distributions. That meant that our software was available to the entire planet. All the Unixes and Linuxes bundled it. We had a really good revenue source from any vendor who didn't want to make its software open source; it could simply pay us for a proprietary license to use the IP (intellectual property), because Sleepycat owned the IP, so we could license it in both those ways.

KM How did it work? What was dual licensing, exactly?
MO It was a very short license, but it was viral in the same way as the GPL (GNU General Public License). Our software was free for you to use, as long as you also made *your* source code available under an open-source license. If you didn't want to open-source your own code, you could come to Sleepycat and pay us for a different license—one that allowed you to distribute our software as part of your proprietary product.

We didn't have a separate product for paying customers—no special features, no early access. We just changed the terms under which you used our product: no charge for use in open source; fee required for use in closed-source products.

We had a very good business. We had lots of customers. Netscape started us off, but later there were Sun Microsystems and many others.
KM How did that adoption happen?
MS Netscape used Berkeley DB 1.85 just for its browser bookmarks. No transactions, no big deal. You know, you lose some bookmarks—probably not the end of the world.

But there were companies using DB 1.85 to manage credit card data, which scared the living daylights out of us. Those are the kinds of people for whom there was real upsell potential.
KM Berkeley DB was open source, and Sleepycat was an open-source company through the dot-com bust in the early 2000s. What was that like?
MO I joined the company in 1998. Margo and Keith were running a pretty good business off of a server in their living room, from people downloading the software, but

they were working other jobs. Margo recruited me as the first full-time employee, but they had been running the company, sort of as a side hustle, for quite a while.

There were lots of dinky little companies with embeddable databases on the market at the time. You had them bundled with a bunch of operating systems. People were using FileMaker Pro and ISAM (Indexed Sequential Access Method) and all kinds of stuff.

Many of those businesses were eking out a marginal living in the days leading up to 2001. When the bottom fell out of the tech market, a lot of them could no longer raise money. Without investors, they starved to death.

Because of the Netscape money that started the business, we never got addicted to venture capital. We were running lean. We were profitable from the very beginning. Now, granted, Keith and Margo and I weren't paying ourselves very much in 1998 or 1999, but the business was doing fine.

In 1999, we probably had 13 or 14 head-to-head competitors. In 2002, that number was maybe four or five. A lot of our competitors just got cleaned out.

That was fantastic news for us. All of the customers of those other companies suddenly needed an embedded database. We had a pretty good one.

We came through the crash not just OK, but great, because of all the new customers we were able to take on.

MS You know, leading up to the bust was the boom. Everybody was launching websites. Pretty much every one of those websites needed a database behind it.

The architecture of first-generation web servers was such that paying the extra overhead of going to a

relational database system was just too big. The demand for the embedded market was really driven by these "Web 1.0" companies.

That was where Sleepycat really made it. We became the back end for pretty much every first-generation web service.

KM In that bust a huge number of people running websites would have disappeared, so you would presumably have lost some customers.

MO For sure, some of our customers went out of business. But all of the companies that had bought the products of our failed competitors needed to figure out some way to replace what they had lost. And there were a lot more of them.

KM So, you grew before the bust and during the bust. How did things go after the bust? How did Berkeley DB evolve over the years?

MS Who knew the web was going to take off? But it did, and people started assuming data wasn't going to go away. We had a really great, reliable, single-node system. But if your single node crashed, then your service was down. By the early 2000s, that was no longer acceptable.

Our customers started asking us what we were going to do about it.

We decided to build a high-availability product, using replication. It was a single-writer system: All the writes go to one node, but you could have as many read-only replicas as necessary. This let our customers scale their web services.

Because we were an engineering-driven company and our customers were engineers, we had incredibly close

relationships with the people building applications on top of Berkeley DB.

Chris Newcomb at Valve Software was using Berkeley DB as the billing engine for the Steam gaming platform, a high-end, distributed-server application. Chris eventually went to Amazon and brought Berkeley DB with him there, so we became one of the first backing stores for Amazon's Dynamo key-value store.

At the same time, Sharon Perl was building out Google's account management infrastructure and chose us as the initial store behind Google accounts as well.

Key customers like those worked closely with us to make sure that our high-availability product was actually highly available.

Then we got approached by the Toronto Stock Exchange, which wanted an XA (extended architecture) interface for its distributed transactions. We built an XA framework, again working very closely with its engineers.

MO I want to call out Greg Lavender, who just took the CTO job at Intel, and the folks at Innosoft as well. They embedded us in their messaging and directory products very early. Sun Microsystems bought Innosoft, in my view, to take them off the market because Sun was losing too many deals to Greg and team. That brought us deep into the product line at Sun.

Most of our customers used our product in pretty conventional ways, paid us good fees, and weren't much trouble. But we were lucky pretty much always to have half a dozen or so customers that just dragged us north by our hair, pushing hard on scalability and performance and reliability. By having those customers early who really

cooperated with us, evolving the product, we were ready for what the market needed before the market got there.

MS There are two pieces that stick in my head. One was conversations with Google and Amazon, when they said, "Things that happen once in a million times happen millions of times every day for us." That was a new way of thinking about what scale really meant.

And the other piece was that computerized trading on Wall Street was becoming a thing. One of the key applications was matching buy orders with sell orders. Every house on Wall Street needed to do that, and they needed to do it super efficiently. Pretty much all of them built exactly the same application on top of Berkeley DB.

"Things that happen once in a million times happen millions of times every day for us."

We would find ourselves in interesting situations where customer A would want a small feature that would let them build it faster. We'd figure out how to do that and we'd release feature A.

Then we'd go to company B, which had the same problem; we couldn't tell them that their competitor was using feature A to solve the problem. So, we would brainstorm with them about how to design it, and sometimes they recognized that the feature we just added was quite helpful! Other times, they'd say, "No, no, no, we can't do that; we need something else." The entire group of Wall Street customers pushed us hard on scalability and reliability.

KM You said that the replication product had a single write point. Did that eventually get fixed as well?

MS For the C version of Berkeley DB, it did not. We remained single-writer forever. But to clarify: If that writer crashed, one of the readers took over as the writer so that

the system kept running.

To the best of my knowledge, the C core product has remained single-writer. I believe that the Java product, which is now the engine behind the Oracle NoSQL database, supports multiple writers, but I am not 100 percent certain.

KM Replication, OK. What else got added to Berkeley DB?

MS In the early 2000s, XML was all the rage! We drank some Kool-Aid. We had some folks who were ready, willing, and able to build an XML database on top of Berkeley DB. We took the plunge and did it, and there was essentially a zero-million-dollar market in XML databases. That was pretty much a complete bust.

Java, however, paid off. It was a Keith idea. We'd always had a Java API. Initially, it was a really bad Java API, so we hired someone who actually was a Java programmer. We got a much better Java API.

Keith felt that a Java API on top of our C storage engine was a bad technological match, and he was completely correct. He argued that we really needed to build a native Java product.

Linda Lee and Charlie Lamb came on board to build this native Java product. We thought there was going to be a big market, but that market did not materialize before the acquisition.

Post-acquisition, inside of Oracle, that Java team has gone on to become the Oracle NoSQL team and the Oracle NoSQL cloud team. Berkeley DB Java Edition continues to this day as the engine underneath the Oracle Cloud NoSQL database.

KM Well, that very nicely takes us into the Oracle

acquisition.

MO Berkeley DB steadily improved over the entire life of Sleepycat. We had a good product, and we were pretty good at selling it.

It was a nice little business. Every year we would take our profits and pay them out to ourselves as dividends for our ownership stake, and to our employees as profit sharing for the great work they did in building the product, winning customers, and keeping them happy.

Everybody loved that! There is this Silicon Valley disdain for "lifestyle businesses," but a whole bunch of cash hitting your bank account every year is actually a pretty nice lifestyle.

MS It also created a very different mentality. Since every person in the company stood to gain by every dollar of profit the company made, the employees treated company money like their own money. You didn't see our sales team spending ridiculous amounts of money or making promises for features that engineering couldn't deliver.

MO We'd been paying ourselves dividends and growing the business at 30 percent pretty consistently year on year. That was a solid performance, but we were concerned about our ability to sustain that growth with our presence in just the U.S. market. Expanding to Europe or Asia was obviously going to cost us a bunch more money.

None of us or our spouses wanted to reverse the flow of funds from our bank accounts back to the company.

We really had two choices: We could go find a venture capitalist to back the business, or we could sell the company outright, taking all the risk off the table.

We realized that venture investment would really change us. There'd be no more of that dividends-going-into-our-bank-accounts foolishness. We were opinionated nerds who had driven strategy on the basis of our engineering taste. We wouldn't be in a position to do that in the same way with a traditional venture capitalist calling the shots in the boardroom.

That left us with option two: Just sell the company. We hired an investment bank, and they got us a bunch of meetings. Turning that crank yielded a few interested parties, of whom the most interested was Oracle.

I'm leaving out a ton of detail, but we negotiated the acquisition and announced the deal on Valentine's Day in 2006. A pretty good portion of the Sleepycat team is still at Oracle.

MS Interestingly enough, a few of the original Sleepycat engineers, led by Dr. Michael Cahill and Keith Bostic, left Oracle around 2010 and formed the company WiredTiger to build a new storage engine reflecting the hardware changes since 1990. That engine is also open source. It's currently the underlying storage engine for MongoDB, a major player in the distributed NoSQL database space, including its Atlas Cloud Service.

KM From Sleepycat to WiredTiger! That's great.

MO At Cloudera, the company I started after I left Oracle, we hired a number of folks from the original Sleepycat team as well. The Sleepycat experience created a pretty close-knit crew.

KM Presumably, Berkeley DB is still out in the open-source world.

MS For better or worse, Berkeley DB continues to be open

source. The reason I say "for better or worse" is that every academic who wants to show how fast their database is, builds one with 2020 technology, and then compares it to Berkeley DB, which was built for very different hardware. They then run it out of the box, which was designed for a tiny embedded environment and has a microscopic cache.

So, they run it on 64-gigabyte servers, give their system 60 gigabytes of cache, and give Berkeley DB 64 kilobytes of cache. And then they say, "Wow, we're really fast!"

MO I want to talk a little bit about the philosophy behind the software.

The Unix philosophy is to build a rich set of composable tools. We wanted Berkeley DB to adhere strictly to that philosophy. We didn't want to encumber it with a whole bunch of capabilities and features that weren't really core to its mission: reliably storing data for applications.

We used to get yelled at all the time because we didn't have a SQL interface. Our view was that there were plenty of good SQL interfaces in the world, including one at MySQL, that you could graft onto Berkeley DB. We were a table manager for MySQL back in the day. Anybody who wanted to add those capabilities could take our finely crafted high-performance engine and add whatever they liked. We didn't want to encumber the core library with that.

We had an exceptional technical team. Our two founders, Keith and Margo, are world-class engineers. One of my favorite quotes by Keith is that in every release of the software, he wants to remove more lines of code than he adds. That doesn't really happen often, but that relentless focus on simplification and streamlining

benefited Berkeley DB.

Oracle has continued to develop both the C and Java releases, and both are available as open source. Until I retired in 2019 I was on the mailing list, and I got the announcements of every new release. Oracle did finally add SQL.

KM We see that in the BSD world as well. We try to take out or refactor a significant amount of code to keep the overall size down. Meanwhile, the Linux Foundation touts the fact that it's adding a million lines of code a year to Linux.

MO You know the old saying: So simple that there are obviously no bugs, or so complex that there are no obvious bugs.

MS Right.

KM Any concluding remarks?

MO I'm really proud that we won the ACM Software System Award, and I'm proud that the Berkeley DB team won the earlier ACM SIGMOD System Award. It's pretty unusual that a software artifact created 30 years ago is still in as widespread use, and is as influential, as Berkeley DB.

This was a really wonderful project to work on. Sleepycat was one of the very best companies that I ever worked for. It was fun to be with my friends. It was great to be serving an audience that we understood well. I feel like we did a lot of really interesting and innovative things.

References

1 Seltzer, M. and Olson, M. (January 1992). LIBTP: Portable, modular transactions for UNIX. https://dsf.berkeley.edu/papers/ERL-M92-02.pdf

2 Stonebraker, M., Stettner, H., Lynn, N., Kalash, J., and Guttman, A.. 1983. Document processing in a relational database system. (April 1983). https://dl.acm.org/doi/10.1145/357431.357433

CONTENTS