EPISODE 1511

[INTRODUCTION]

[00:00:00] ANNOUNCER: This episode is hosted by Alex DeBrie. Alex is the author of *The DynamoDB Book,* the comprehensive guide to data modeling with DynamoDB, as well as the *DynamoDB Guide*, a free guided introduction to DynamoDB. He runs a consulting company where he assists clients with DynamoDB data modeling, serverless architectures, and general AWS usage. You can find more of his work at alexdebrie.com.

The default configuration in most databases is meant for broad compatibility rather than performance. Database tuning is a process in which the configurations of a database are modified to achieve optimal performance. Databases have hundreds of configuration knobs that control various factors, such as the amount of memory to use for caches, or how often the data is written to the storage. The problem with these knobs is that they are not standardized, they're not independent, and they're not universal. In reality, information about the effects of the knobs typically comes only from expensive experience. OtterTune is an automatic database tuning software that promises to overcome these problems. It uses machine learning to tune the configuration knobs of your database automatically to improve performance. In this episode, we interview Andy Pavlo. Andy is a database professor at Carnegie Mellon, and co-founder of OtterTune.

[INTERVIEW]

[00:01:32] AD: Andy Pavlo, welcome to Software Engineering Daily.

[00:01:34] AP: Happy to be here. Thanks for having me.

[00:01:36] AD: Awesome. Well, I'm super excited to have you here, because you're someone I've sort of followed and read about in the database community and you've done a lot of work. I think it's interesting because you're in sort of two worlds, right? So you're in the academic world, your associate professor with indefinite tenure in the computer science department at Carnegie Mellon, doing all this awesome database work there. But you're also in the private industry world, you're CEO, co-founder of

OtterTune. I want to talk about both those today. But maybe just anything else I missed there or cool stuff you're working on?

[00:02:02] AP: That is accurate. So indefinite tenure is what the – basically it means I have tenure as of this year at CMU, which is good news. They can't – I still can get fired, I just – I have tenure. Yeah, my background is in database systems, and part of the reason why we've been doing – with the startup, OtterTune, as well as maintaining my foot in the universities. Because I like to think that a lot of interesting things happening in the real world, and I'm using sort of the – we push things out towards research into a startup. Then, I'm sort of also getting feedback about other future problems maybe we want to investigate back at the university. So it's been a symbiotic relationship, and it's been pretty fun.

[00:02:39] AD: Awesome. Awesome. I assume – congrats on the tenure, and I assume that means that you can now advocate for NoSQL, and Blockchain, and all these other things that you've been sort of holding back for all these years. Right?

[00:02:48] AP: Not in my life. Actually, no.

[00:02:50] AD: Okay, cool. Well, I want to start with OtterTune. I think it's super interesting. So OtterTune, I would say is like sort of self-driving database in some way or automated optimization, things like that. Maybe you just tell me, what is OtterTune? What is it doing?

[00:03:02] AP: OtterTune is an automatic database tuning as a service. We can talk about the self-driving database stuff, that's sort of separate research direction. But the original idea was that, we were looking to use machine learning to automatically, to configure and optimize database systems. This is work we've done back in the university with my PhD student, who is now a co-founder with me as well, Dana Van Aken. Her research is really about how to use machine learning to automatically tune the runtime knobs of database systems. So you can think of like a database system like Postgres, MySQL, Oracle, pick your favorite one. For the most part, they're essentially a general-purpose software, and that they have these knobs that you can tweak and change for how the application expects to use the data system. So you can tweak them certain ways to make them faster for certain type of queries, or certain types of workloads versus others.

The metaphor I like to use sometimes is, think of this as like, the database is like a generic car frame, I can put big wheels on it, and they can have pick-up truck and haul things. Or I can put a fast engine and make it a race car, right, but the frame is essentially always the same. So that's what OtterTune is trying to do, is trying to tune these knobs for exactly how the application wants to use the database system. The reason why this is hard, and why we think we need machine learning and artificial intelligence techniques is because, over the last two decades, if not longer, these very popular open-source database systems have been adding lots and lots of more these knobs where it's beyond the sort of capabilities of a human to figure out how to tune them and reason about them. So just offhand, I'll say it, from for Postgres, since like beginning of 2000, they've gone from maybe less than 100 knobs to over 300, 400 knobs. MySQL went from less than 100 to over 500 now. Not all these knobs will affect performance in the ways that OtterTune can actually tune them, but there's still a lot there.

What we have found is that, with the rise of DevOps and the somewhat relegation of the less prevalence of DBAs in lot of tech companies or any company, there's no people know deeply what's going on inside a database system. So OtterTune sort of seeks to automate that, take that burden off of them for them.

[00:05:03] AD: Truly, absolutely. So in terms of the things it's actually tuning, or the things that affect database performance, I think instance size, I think indexes, I think data model, and then also the configuration. So it's mostly configuration like buffer size, file limit, I assume things that are way more important than that, but those are the things that it's sort of going after.

[00:05:20] AP: With original version of OtterTune at the university, and then what we end up commercializing at the startup, it was just touching knobs. Because again, there's enough of them that – the things matter from around experiment and what we see in the real world. We start getting into other data problems, like indexes, query tuning, is the size that usually falls under capacity planning or provisioning. Those are where we're going next with OtterTune. The newer version that we're going to announce and release later this year. What we basically found is that the knob tuning makes a huge difference, but getting people to use OtterTune correctly, and safely has been sort of a challenge for us. When people do use it correctly, it's fantastic. But oftentimes, there are other things that, there's wrong with their database. No amount of knob tuning can fix that. If you don't have any indexes on your tables, then who cares that your buffer pool was tuned exactly, because you're just doing sequential scans. That's where we're going next with OtterTune.

The other challenge we faced is that it's hard to, in real-world applications to be able to convey back to the user, like how much OtterTune they think is better for them, if you just do knob tuning. Again, when we did the early rollouts or the trials at the university and in the startup, we talked to customers that could take snapshots of the database, clone, and work with trays, and run spare hardware, do all the tuning there, and then put it in production, and then get all the benefits there. A lot of people can't seem to do that, either for technical reasons or just time reasons. No amount of machine learning can make that better, at least from where we stand in OtterTune, since we're not the service provider, we're not running RDS, we're not running the database for them. We're on the outside, we can only do so much. So then the newer version of OtterTune that we're building out is starting to look at some of the other things that you mentioned.

[00:07:01] AD: Walk me through the process. You sort of mentioned a little bit with workload tracing and snapshots. What's the process? If I come to you, I want to use OtterTune, is it going into my production database, or am I spinning up a clone, and sort of doing that, or what am I doing there?

[00:07:14] AP: The way it works now is that, you sign up for a free account, and you would give us the right IAM permission to AWS. OtterTune currently supports MySQL and Postgres, with the regular RDS and Aurora versions, but only running on Amazon. You would initially give us permissions to sort of AWS level, see the databases, see the RDS instances, and we can collect some basic information from CloudWatch and Performance Insights. This is like benign information, like CP utilization, some of the internal metrics the data system exposes. We're not looking at any user data or any queries at that point. For that initial permission, we can then calculate some things, identify – we basically run health check to show you like, "Hey, these things look wrong. Here's how maybe some things can fix them" or "Hey, these other things don't look good either." Then we – the solution might be then to download our agent, Dockerfile that you run in your network, that then connects the database to give it additional permissions to start looking at additional metrics and telemetry that database is collecting. And that sends us back to our secure service, and then we can start providing other recommendations, identify other issues.

Then, you can then also enable the sort of ML based tuning to get sort of the full benefit of this. From our perspective, we recommend that people don't do the ML tuning on the production database. Most people will listen to us. For some reason, Brazilians do not. I don't know why. We're like four out of four

for Brazilian customers we talked to. They're like, "Hey, this machine learning, you don't want to run this production, like not right away." "We don't care." It's really bizarre. It's like driving without a seatbelt. Now, let's say, we put a bunch of guardrails in place to make sure that people don't shoot themselves in the foot. But it's –if it's your production database, you obviously want to be very cautious about this. So the next version of OtterTune we're having, we're working on now is the – we can do sort of more lightweight ML things about indexes and other stuff that you can get some of the benefit in production more immediately. But then you can then still switch on what I call the OtterTune classic and get the full-fledge machine learning capabilities. Then, you have the ability to identify, is this a production database versus a dev staging database, and be more aggressive maybe in the staging one, and then know how to link that and apply the changes to the production automatically. That's where we're going in the next year, to have a clear identification of what's dev versus production. But for now, it's all – from our perspective, it looks the same.

[00:09:36] AD: If it's running automatically on products, it's actually like – is it changing those configuration values for, if you sort of set it ahead or is it – could it like generate a report for you and say, "Hey, recommendation is to change this buffer pool, but it's up to you."

[00:09:47] AP: Yeah. That turned out to be the number one request we had when we came out of stealth. We said, "Hey! We're no longer a research project. We're now a startup." People asked us like, "This is fantastic. I want this, but I want to be able to approve any changes before OtterTune does anything." So we have this human feature where the machine learning algorithm will run, and then it produces a report, it says, these are the changes we think we should make, and then a human has to come and say, "I want that." We don't have the ability just yet to say, "Give good explanations why **[inaudible 00:10:15]** makes these changes." That's again, it sounds like I'm sort of talking about the future is coming, but we're actively working on this now. We want to be able to say things like, "All right, here's the recommended change we want to make. Click here to let OtterTune make it or not. And oh, by the way, 95% of Postgres users, when they applied similar changes got this benefit." That would give the additional peace of mind we think the customers – we know that customers are asking us for. So that's where we're going next.

[00:10:41] AD: Yep. Cool. Are there certain types of workloads or customers that you're seeing OtterTune like really resonate with? At first, I was thinking, hey, enormous workloads, they have these

huge scaling problems. But then I'm like, hey, maybe they also have a team of DBAs that can handle this. Maybe it's like a smaller team that doesn't have a DBA. I mean, what are you seeing?

[00:10:57] AP: Honestly, at this early stage, it's everybody. We've talked to large companies that have in-house DBAs. But oftentimes, what they tell us is, either the scale is too large, and machine learning – or something like OtterTune take off some amount of burden of their daily job, or they have inherited or acquired some other company that maybe is running Postgres, whereas they're all Oracle, DBAs, or Db2 DBAs. And they don't know anything of Postgres, and they can learn a little bit, but they want OtterTune to help them get them there. So there are big companies like that. There are so many large tech companies that don't have in-house DBAs, and it's all DevOps. Then there's mid-range shops where they don't have a DBA, they're considering getting one, but then they realized how expensive that is, and how hard there are to find the good ones, and they're trying OtterTune now as a way to maybe kick the can down further down the road and see how much they can still automate through us before they hire somebody else. Then certainly, there's the smaller stage startups and companies where, again, they don't have a DBA. We don't actually recommend they get a DBA, because if you're paying for an employee just to maintain your database, rather than building your application, that's not really good for an early-stage startup. So OtterTune is useful for them to help them –as the application grows, data is congruence with them. Yep.

[00:12:13] AD: Yeah. Awesome. I think that's so cool to see. Especially that first one you're talking about like, whether they have some DBAs. But if you can just even just reduce the space they have to search or sort of like we're seeing in chess. Where like computer plus really good chess player is better than computer alone or person alone. It's like they can supercharge that sort of stuff.

[00:12:31] AP: But sometimes they really have tens of thousands of databases. We've seen like MySQL hosting companies where they have 10,000 different little WordPress apps or whatever. No human can go to in every single one, because it can't with that scale. Now, OtterTune can't do everything, right? Machine learning has its limitations. Anything that requires a human to make a value judgment. OtterTune can't do that, because the human has to decide what's the right thing. This goes back to what I've seen for the two, where we have to make sure we have the right guardrails in place to make sure that things don't happen, because the machine learning algorithms figured out that not writing things to disk make things go faster, which is true, but then you crash and lose data. That's an external cost the algorithms can't reason about. So we have a bunch of these things in place, which

actually takes the most amount of time more than the machine learning itself, to make sure we do things safely.

[00:13:20] AD: Very cool. One thing I think that's interesting on your site right now, you have this OtterTune Relational Riverside Rumble. Human versus Otter. Tell me a little bit about that.

[00:13:27] AP: People ask us oftentimes like how good is it? I'm like, "Oh, that's a good question. I don't know how good it is." We have an academic paper that came out in 2021, where we deployed OtterTune, a bank in France. We did compare against their in-house DBA, and sometimes we beat them, sometimes they beat us. But that was on Oracle. We want to know how can we do for Postgres and MySQL. So Relational Riverside Rumble is a contest we're having this year, which we're going to hold at re:Invent, at Amazon's big conference in the end of November. Where we want to put a live human DBA that's not affiliated with OtterTune against OtterTune and see what happens. If the human can beat us, they'll get $10,000. If they lose, we'll still give them some portion of money, then we'll donate the rest to the Otter Wildlife Fund, which is based out of Britain. But the key thing about this is we want to do this comparison on a real database. Because, certainly in academia, we would always compare against TPCC and the other standard benchmarks everyone uses. All the other academic papers do the same thing.

We have just found in the real world, it's – the workloads are much more varied than what's available in academia or synthetic workloads. We're still working on this now, making the arranges. We're going to do this comparison on a real database from a real customer and put the human against OtterTune, and then we'll livestream it, that week during the event, and then it will be available on Twitch and YouTube.

[00:14:51] AD: Yep. Awesome. I'm looking forward to seeing that for sure.

[00:14:53] AP: And if we win, fantastic, right? If we lose, that's good too, because it will just be like Garry Kasparov and come back the next year and do it again.

[00:15:00] AD: Exactly, and you'll learn something. It kind of gives me like the Netflix prize too, when they had that competition, it ended up improving their recommendation. So yeah, that's cool stuff. I want to talk about, what's the reception like in the database community? I imagine there are camps like,

"Hey, it can't be automated. They're too varied." Or is it like pretty much like, "Hey, this is where it's going, and we should be moving forward on this." What does that look like?

[00:15:22] AP: I mean, the real question is like, what would DBAs think about this? The DevOps people, developers, everybody else, everyone's on board with this. I think partly because they maybe don't know the full complexity of database systems, and also, they know or maybe they face the challenges, they scrambled to try to solve them before, and they know they don't want to do it again. Everyone who's not a DBA has been on board with this as well. The DBAs have actually also been very enthusiastic and supportive of this as well. I mentioned that competition we did or the research study we did with the French bank. That was at the behest of their DBA teams, because they wanted to see how well can OtterTune do at their scale, and their environment, because they were certainly struggling with it. If you just think about the original idea of doing automated knob tuning, that's not something DBAs actually spent a long time with anyway. Because maybe they had five or six knobs that they always tune, and that got them maybe 60%, and 70% of the way there, obviously squeaking at that last bit of performance is super hard. They had so many other things to do. They were not doing knob tuning.

For classic OtterTune, again, the knob tuning, everyone has been super supportive of this. Where we're going next, it's query tuning, index tuning, resource provisioning and sort of cloud configuration stuff. Basically, modernization of – running data is in the cloud. They also seem to be very supportive of this as well. One, because they have so many other things to do, and if OtterTune take care of these things for them, they can spend time doing the higher-level things that, again, we can't easily automate like application design, schema design, figuring out what the right queries are right for different reports and things like that, backup, and recovery, like all those mechanisms. OtterTune is just making easier to them to do the more fulfilling things in their day-to-day job. When we pitch it sort of that way, everyone else seems to be supportive as well.

[00:17:11] AD: Nice. Okay. Then a few last business specifics about OtterTune. Have you all raise funding? How many employees do you have? What does it sort of look like as a company?

[00:17:19] AP: We've just raised our Series A in April 2022, Intel Capital, Race Capital, and Excel. Prior to that, we did a small sort of seed round/angel funds with database friends, like the founders of Databricks, Snowflake, MemSQL, Cockroach, a bunch of database people. This guy, actually,

[inaudible 00:17:36], who's been very, very supportive of us. When we raise the first initial funds, I basically went and hired all my best former students from Carnegie Mellon. We set out to rewrite the software, and build a commercial version of the academic project. So we did that, and then we announce – we've sort of launched self-service version OtterTune. As I was saying, what we learned from releasing the software was, the non-coding makes a huge difference, but getting people to use it fully has been – I wouldn't say struggle, but it's been non-trivial, it's been hard. Harder than I anticipated.

With the Series A, the new funding, we're building a new version of OtterTune that starts to look at all these other things that people care about in databases. That's what we're hoping to announce at the – the same we'll be having the contest, it will be sort of the announcement of the new version of OtterTune. We've raised total, I think, about 14 and a half million, and we're currently, I think, 25, 26 people. We've started hiring people outside of my sort of the CMU network, right? All my students do databases, but people do front end, and marketing, and product design. We've been very fortunate to hire some fantastic people that weren't my students to help us build the product. So we're super excited.

[00:18:42] AD: Yeah. Cool. I want to move into the academia side here. But when you mentioned that hiring all your former students, what are most of these former students that are really getting deep into database stuff? What are they doing? Are they doing more acad? Were you hiring them at academia? Do they go work for big tech companies that are optimizing Google's MySQL or what sort of things are they doing?

[00:19:01] AP: You can sort of break it up between like sort of undergrads, master students, and PhD students. Since I started at CMU in 2013, most of my PhD students have gone in academia. My first student is now a professor at Georgia Tech. My latest PhD student is now starting as a professor at University of Michigan. Dana, I mentioned, she's at **[inaudible 00:19:18]**, and actually my other PhD student, he's at Databricks. For PhD students, it's been a mix, but more than good academia. For undergrads, the very best go to do PhDs and databases. It's been mostly split between – I keep some of them at CMU, which is somewhat selfish, but they want to say it's fantastic. But then a large some of them go **[inaudible 00:19:41]** MIT. For master's students, most of them go to industry, and most of them, they hang out in our group, they end up going to database companies that are either startups to build data systems, or Google, Amazon, and larger companies, specifically work in databases. I think

most of them, not exact tally, in the last five years, a lot of them end up at like Snowflake and Databricks. But pretty much, you name any startup company, I might have a student there.

[00:20:07] AD: Yep. Awesome. So let's go deeper on the academia stuff, because that's where I have heard about you and learned a ton from your stuff. Maybe let's start with this – I don't know exactly how you call it. Database tech talks, or your seminar series, or whatever it is. Maybe tell people. But there's just this enormous wealth of videos from super knowledgeable people talking about databases, in association with your class. So maybe tell us a little bit about the work you're doing there.

[00:20:30] AP: I've run a seminar series every semester, through the fall and spring semester, where on Mondays, we invite people from industry to come up talk about their database systems. It originally started when I was in grad school. I was sort of co-teaching the class of my advisor at Brown University, and New England and Boston area had actually a lot of data startups. I consider it sort of my duty as a researcher in academic to understand what's going on industry. We'd invite them to come to the class and give a talk. Then, when I graduated, started at Carnegie Mellon, I sort of continued the same thing, but rather doing it part of the class. We have this consortium at CMU called the Parallel Data Laboratory, where we can invite people from the outside to come give talks on Thursdays about data systems, because that's what I cared about. The reason I think this is super important, because databases are unlike other sort of areas in computer science and systems, where not only do you have to compete as an academic with other academics to other universities. You have to compete also with the large tech companies, the Googles, and Microsofts and Amazon, because they're betting data systems. Then there's all this other VC money sloshing around, funding all these other database startups. So there's all these other people as well, right?

I think it's very rarely – if you think of operating systems, there's no – everyone uses Linux. There may be some VC back OS startup, but I can't imagine – I don't know any, right? But in databases, there's so much so. To make sure that my research is grounded in solving, trying to solve real-world problems, and I'm not just reinventing the wheel that somebody else at another company has solved. I make an effort to try to understand what everybody's doing. We started inviting people to come give talks. Before the pandemic, we invite them on campus, set up meetings with other researchers, and students, and take them out to lunch. It's a big production. We maybe only invite maybe five, six people per semester. Then the pandemic hit, and I was like, "All right. Well, everything's on Zoom. I'll invite everybody who I always wanted to invite, but I couldn't fly out people in Europe, people in China, people in Australia."

That's been super fun. Just every Monday now, we have somebody else come give talks about databases, and I learned a lot from it. Prior to the pandemic, I would try to have a theme at least about what the seminar series would be about. Like one semester, it was embedded databases. So like SQLite, RocksDB, WiredTiger. Then we did timeseries databases. We did Harvard accelerate databases; it gave us funding of GPUs. But now, honestly, it's whatever Andy thinks is interesting, so it's bit random. There isn't really a theme bought in. But I would say that I try to get people to talk about query optimizers, which is usually the black art, the mystique part of a data system. We've had some really good talks with them. But recently, it's been like, hey, this showed up in Hacker News. I want to learn more. We invite them.

[00:23:13] AD: Yeah. Oh, man, that's awesome. I believe we're recording this in mid-September 2022. I believe that the seminar starts tomorrow. Is that the first one of it?

[00:23:21] AP: Yeah. The first speaker tomorrow is actually Thomas Neumann, which – if you haven't talked to him, you should get him on this podcast. He is honestly probably the best database researcher in the world. I don't say this word lightly. He's a genius. He's got three kids. He teaches two classes per semester. I think he's the department chair at – he said at TU Munich. Then he was running or doing something for like the COVID vaccine distribution in Germany. He's doing all these things, but he still finds time to program these databases eight hours a day. He's got to be on cocaine or – but I know, he doesn't drink. He's amazing, and he's like the nicest person ever. Anyway, so he's giving a talk about his database system called Umbra, which is the second one he's been building since the last one he sold to Tableau and Salesforce called HyPer. Anyway, this is just – I think Thomas is a great person, and I think more people outside of academia need to know who he is, so that's why we landed him first.

[00:24:12] AD: Awesome. I would definitely look him up after this. I'll put this in the show notes, but like, absolutely check out this databases series, because it's just amazing. The breadth and the depth that you can get from this stuff and whatever sort of database, like you're saying, embedded, time series, interesting stuff on MySQL, Postgres. There really is a lot of cool stuff, so we'll put some stuff in the show notes on there.

[00:24:32] AP: What's been really good is, we specifically asked not for them to bring marketing people or PMs. It's the engineers actually building the systems. We tell them, like your podcast, say hey, "If you say something inappropriate, we can cut them out." We tell them that and sometimes they still dish the

dirt about like, "Hey, there's a dark corner. Here's something stupid we did. Here's something stupid we see our customers doing too." That's been super, super awesome. Even though again, everything's public on YouTube and Zoom, anybody could comment. You don't have to be affiliate to the university, and they still are very open about what they talk about.

[00:25:04] AD: Awesome. I love it. Okay. What I want to go to now is just like, you get to see all these people, you're in this database area of academia. What are the interesting trends, patterns you're seeing? I have a few I want to ask about like what's exciting to you right now that you think is really interesting.

[00:25:21] AP: I think, again, for database systems, I think we're in –I haven't really articulated these thoughts yet entirely. But I think we're actually in the golden era of databases. Similarly like the golden era, the movies or TV, where there was just so many different choices that were like all very high quality. That's sort of where we're at now. I don't know if that's a byproduct of VC money, or open-source software. I don't know what happened. But there's a lot of great choices now that are really, really good. Some are open source, some are cloud based, but there's so many choices now. For me, just thinking more long term, because I can't in academia. I don't see – I don't foresee what's the next major thing coming on the horizon, where there'll be a bunch of people screaming about a new system. Like if you look at like Snowflake when it came out in 2013, they were sort of the first cloud native vectorize data warehouse. Redshift came soon after, but they were hacking **[inaudible 00:26:18]**.

By now, in 2022, vectorization is super common. Query compilation is less common, but the research basically shows that you get the same, roughly the same benefit overall if you vectorize the query engine versus do code gen query compilation. Some systems do both, some systems choose one or the other. But the point I'm trying to make is like, I don't foresee there's some major thing coming on the horizon, like new hardware, major, major change where the database world is going to get flipped over upside down again, and everyone's screaming they've built something new. Intel killed Optane the summer in 2022. I actually thought that was going to be a big game changer, but there were some companies building persistent memory optimized systems. That's not happening anymore. I really don't know what it is.

Potentially, I think it's going to be just the heterogeneous hardware, like the dominance of Xeons, of the x86 architecture. I'm not saying GPUs, but like custom ASICs, things with ARM or RISC-V. I think the

next decade, the hardware is going to get pretty wild. So there could be database systems that are built on that are optimized for these different pieces, or different hardware types. But the stuff that actually matters is up above, like the query optimizer, for example. Who cares if your engine is super-fast, if you choose crappy query plans. I think where we're at now for this next decade is sort of commoditization of high performance, data warehouse engines, and transaction engines. What exactly is going to come out next? I don't know. Honestly, I think it's the stuff above. Not the push OtterTune, but that's an example of the up above stuff. Who cares how fast your query engine is if you can't build the right indexes, if you can't tune things correctly. That's what we're sort of focused on now.

I think newer hardware could be a thing. I am excited to see how more of this sort of, like the roster of zig, these sort of safe memory languages, how they're becoming more prominent and kind of systems building of those. But I don't think it's a game changer in terms of the architecture of the system is dramatically different.

[00:28:16] AD: Absolutely. You mentioned both like analytical and transactional ones. We've seen a little bit of talk of like the whole HTAP, like hybrid transactional analytical. I think Snowflake had something at their latest conference talking about that. What are your thoughts on that? Are we going to see more hybrid type stuff are we still going to see more specialization into, at least those two worlds, analytical and transactional?

[00:28:36] AP: So full disclosure, I was a tech advisor for Splice Machine, which was pitching itself as HTAP engine. Then they went under, I think last year. I think it'd be very hard for a newcomer to come along and try to build an HTAP system from scratch. Because if you think about it at a company, the stakeholders, or the people that care about the OTP are oftentimes – the OTP is often different than the data warehouse engines. The OTP, the people building the front applications, they want the best OTP engines you can possibly get. I'm not saying what that is, which in their minds, they want whatever that is. Then the same thing with the data warehouse people, they want, whether it's Snowflake, or Redshift, or Fireball, whatever it is. They want the best OLAP engine.

For someone to come along, say, "I have something that can kind of do an okay job of both." That's a hard sell. I think the Splice Machine approach were like, "Hey, let's build an all-in-one solution that comes out of the box as all-in-one solution." I don't foresee that happening. Instead, I think is going to be essentially what Snowflake is doing where with Unistore, where you have sort of best-in-class data

warehouse or best-in-class OTP engine. Then you add some capabilities that can do a little bit of what the others can do. That's essentially how things are going to go forward. I think that, whether that's called each HTAP, or whether that's just like, "Hey, we add now analytics or transaction engines." I don't see that being a key like differentiator in the marketplace.

[00:29:55] AD: Yeah. I think it's so hard to really merge those worlds in a way that does degrade performance on each side, so it's a tricky one. What about like more specialized use cases? I think you have SpiceDB coming this year, who's like a specialized off-ish database, something like application-ish type stuff. But even like time series graph, search, do you think we're going to go deeper and deeper into those more specialized use cases over the coming years?

[00:30:18] AP: Specialized engines are always going to outperform sort of general-purpose ones. The question is whether the specialized engine is going to have such a significant difference in performance that can overcome the limitations of a general purpose one. By general purpose, I would say like MySQL, Postgres, Mongo, Oracle. What people think when they think of a database system. The challenge oftentimes is, is the benefit you're getting from a specialized engine. Is that true because the engine is different or the API is different? So an example of this, we look at a graph database. If you're trying to do graph traversal, you can write this, an application code in SQL without recursive queries. But you can do the graph traversal by writing a query, getting result. Then the application code does something and then you run the next query. So there's a lot of back and forth, and that's going to be slow. Oftentimes, in the graph databases, they have API calls, API commands, where you can do the graph traversal in a single query, and then avoid that back and forth. That's what they may claim as one of the reasons they're better than relational data to graph workloads. You obviously can build a graph like veneer on top of a relational database, and to do that traversal on the server side without having to back to the application and get almost all, if not the same, or if not more benefit, right?

The specialized engines, I think I'd be hesitant to say we're going to build something from scratch, and that's why it's better versus like, you just have a better API on top of Postgres. You mentioned SpiceDB, I think they were running – they had plugins. They could run off cockroach, I think also Spanner, but they basically had a specialized API on top of a relational database. I think that's where things are going, and this goes back to the statement I was saying before like, what's different in the next decade? I think there's so many good databases out there, it's very hard to see what's – from my perspective, what's something you could build from scratch and engine that you simply couldn't do in a relational

database, or you couldn't take Postgres, and put in extensions and foreign data wrappers the way timescale did and do better for time series that way. It's very hard to see what that actually is going to be at this point. Tiger Vito, for example, would be one where they're doing things – it's a very, very narrow application use case. It's just that ledger. That's why they can build something that super-fast, and it just so happens that that use case is super important for a lot of applications.

[00:32:38] AD: Yeah. People are willing to pay for that, almost for that specialized type thing where other ones might not have that. You mentioned, Spanner, Cockroach. We've seen other distributed SQL. You go by Vitess, PlanetScale. What do you think about the whole distributed SQL thing? Are we going to see a lot more of that going forward or will – just Vanilla Postgres, Vanilla MySQL get good enough for most people to where most won't need that?

[00:33:01] AP: All right. I have to admit. I am so surprised that what I call the distributed SQL systems, the Yugabytes, the Cockroaches, that they took off as much as they've done. I say this as someone who was an early proponent of NewSQL. I worked on – my PhD research was on a system called H-Store, which was commercialized at VoltDB. I admit, I was totally a proponent of like, "Hey. This is the right way we do transactions," I still do for some things the way VoltDB does it. But there were issues of, at least in the case of VoltDB, that it wasn't like a drop-in replacement for Postgres or MySQL, but some of these newer systems are. So the other challenge I also saw from the early NewSQL days was just how hard it was to supplant or replace Oracle or other enterprise transaction database systems.

The Cockroach guys, the Yugabyte guys seem to have sort of cracked that nut, certainly for Greenfield applications. I don't know if they're – I don't think if people are replacing Oracle for Cockroach. I'm sure they are, but I imagine, I suspect it's mostly new users using them. I think that part is different than what the NewSQL guys were able to do. Do I see more distributed SQL systems coming along? Again, I think it's – I'm sure, yes, but like those systems are so well funded. It's hard to really build something from scratch and make a big difference. Plus, you're also competing against Aurora, and Amazon and Spanner, and so forth. So, I would say, yes, but I think the market is sort of getting so established already now. That would be hard for a newcomer to come along and make a difference. But I would say, again, someone could take Postgres, pack it up, and similar to what Yugabyte has done, and that would give them the jumpstart in the space.

[00:34:41] AD: For the last, I would say 20 years, like open-source databases have really sort of taken off. MySQL, Postgres has really grown. Mongo, Cassandra, different things like that. Do you think that's going to continue to be true, or will we see closed source but, maybe cloud managed, cloud native ones have other distinct advantages that open-source ones didn't have that proprietary ones before that? Is there an ebb and a flow to this, or do you think that sort of open source is going to continue to grow? What do you see there?

[00:35:09] AP: So cloud makes it very difficult, right? Everyone wants to be open source. Everyone likes open source. But the stuff that you end up building to run a cloud native database is not –it's the backend, a lot of the backend stuff, the infrastructure, and that doesn't mean open source. Kudos for cockroach, I think Yugabyte as well for maintaining open source. I don't how much is in the service version versus the open-source version, what's the difference between the two, but at least people have that option. I foresee Postgres continuing its dominance, and replacing further and MySQL is being the default choice for new applications as the open-source database. But again, I think a lot of people are going to choose to run it on RDS, or whatever, pick your cloud service rather than the – trying to run it bare metal themselves. It's just not worth it.

Postgres is getting a lot better certainly **[inaudible 00:36:02]**. In the 2000s, MySQL was the first choice for new startups, new tech companies as the go-to database, instead of using Oracle and all these other ones. Mongo became sort of the default choice in the early 2010s. Postgres is sort of wearing that crown now. There'll be a lot of applications, a lot of new products being built with Postgres as the first choice. Then maybe as those things evolve in scale, over time, they'll hit the limits of what single node Postgres can do, even slightly replicated, partitioned version of Postgres can do. Then, this is where things like Cockroach and the other services that look and smell like Postgres could come in. In that case, the open-source version of Postgres keeps getting better, but the cloud versions are just making it so that they scale. That's how I see things going.

[00:36:46] AD: Yeah. I think that's probably right. I would just sort of strapped those reading, like some of the papers coming out of Amazon in the last couple years, both the Amazon Aurora paper, and then the new Amazon DynamoDB paper. You mentioned a little bit about the backend stuff that they're doing there, and part of that is – even in just like RDS where they're running Vanilla Postgres or Vanilla MySQL, there's a lot of backend stuff. But then like, with Aurora, specifically, how they're chunking your data into 10 gig segments, and sort of spreading across AZs, and doing all this stuff. Even if they did

open source that, it's just such an operational burden for one team to go run that for themselves as compared to a dedicated Amazon team that can run that for thousands of customers.

[00:37:24] AP: I will say, there is actually an open-source version of something that smells like Aurora called Neon.

[00:37:29] AD: Oh, yeah.

[00:37:30] AP: This was started by the co-founder of MemSQl or now Single Store. I guess, full disclosure, Nikita is a friend of mine, I'm an investor in them. He's an investor in OtterTune. But they are pitching themselves to be the open source of Aurora. I'm pretty excited to see where they're going with this. As far as I can tell, at least from what he's told me, what you get in the Neon service matches what's in the open-source version.

[00:37:50] AD: Awesome. I love that. I'm super excited about Neon. We're trying to get Nikita on right now, so hopefully look for like an upcoming contest. But that one, so yeah, I think there's just this whole like serverless era too. Part of that's marketing, but there is some interesting stuff there about just how much easier it's getting to actually use databases as a renew or developer and things like that. I love that part.

[00:38:09] AP: Again, not to go back to OtterTune. But like yes, it's super easy. But I think, to me, there also is another good argument why you want something like OtterTune. If not OtterTune, whatever database automation. Because it's super easy now to, with no code systems, with this database as a service, whether the service or not, you can pop up something pretty quickly and get a lot of people using it, collect a lot of data. And the infrastructure is super easy for that. But like, didn't understand how the application is using the database, what they're doing, using it incorrectly, you didn't set it correctly at all. Beyond the infrastructure, that's super challenging. We've had customers tell us that they thought Amazon was already tuning the database for them on RDS, or Postgres, MySQL. I'm like, "Jeff Bezos is not doing that. The economics is in their favor not to make your thing run faster." I can't prove this, but our impression is that the default settings for Aurora are slightly tuned a little bit better than the default settings for RDS, and they want you to – so that way, when you switch to RDS as opposed to Aurora, you're paying 20% more and you think it's better. But still, it's just a little bit of nudging will get you.

[00:39:11] AD: Yeah, that's funny. Yeah. Cool. Andy, I've just loved this conversation. Again, I just love all the talks and things that you've been putting out. Thank you for coming on Software Engineering Daily. If people want to find out more about you, about the series, what site they'd be looking for you?

[00:39:25] AP: So it's Otter, like the animal, Tune, T-U-N-E. A play on the words ottertune.com. Then you just search CMU database group, there's the YouTube channel for that, there's the Twitter account and then there's the website with all the details and everything. Again, we try to make everything open source, everything public. So even if you're not a student at CMU, you can still participate in everything we're doing.

[00:39:44] AD: Awesome. I love it. We'll have links to OtterTune, to the database group, all that stuff in the show notes. Andy Pavlo, thank you for coming on to Software Engineering Daily.

[00:39:52] AP: Thanks, Alex. Thanks for having me. I love it.

[END]