Telco in 20 | Episode 41
Telco Religious Wars: Charging Databases
Released in May 17, 2022

DR: I'm Danielle Royston and this is Telco in 20. If you get any three technical people together, they will always find something to fight about. When I was a coder, yes, I was actually once a coder, there were always huge religious wars about which coding environment was best. For example, if you bring up VI or EMACS to a group of programmers, almost always, shouting disagreement would erupt. It's like Marvel versus DC, Coke versus Pepsi or Star Wars versus Star Trek, everyone has an opinion.

P.S. I'm an EMACS girl. When it comes to charging, we get just as intense. Seriously, we do. You can't just pick any database. Charging is a high volume, highly transactional business process, where the database is a really important technical decision. You have to think about access patterns, speed requirements, the amount of data you need to process, how resilient the thing needs to be. The database is a core, fundamental and key decision that you make in the design of your system. And when it comes to charging, we've got three main players at war.

There's the traditional, old school, relational database, like Oracle, womp, womp. Then there are some people out there with their proprietary in-memory data store approach. Hmm. And there's me over here with Totogi and DynamoDB, a non relational, noSQL cloud database. Well, today on the show, we have Alex DeBrie, a DynamoDB expert to walk us through the pros and cons of each approach. And without giving anything away, I'll give you one guess which way I think is right. I can't wait to talk to him, so let's take 20.

Alex DeBrie is Principal of DeBrie Advisory. He's also the author of The DynamoDB Book. Hi Alex, and welcome to Telco in 20.

Alex: Hey DR, thanks for having me. Great to be here.

DR: Yeah, I'm so excited. This is going to be a great conversation, a technical one, and I'm going to dust off my computer science degree for, I don't know, 15 minutes.

Alex: Good, good.

DR: It's so fun. So to start, as I just mentioned, you wrote a whole book on DynamoDB, and so, tell me a little bit about how you became a DynamoDB expert.

Alex: Yeah. I always say it was an accident. I backed into it accidentally. My first job in tech was data engineering, so I was doing data pipelines and more OLAP databases, Amazon Redshift, things like that. Then I was working for a company called Serverless, Inc. They make the Serverless Framework, which just makes it easier to deploy serverless functions, database Lambda, things like that. Most people were using DynamoDB, and these serverless applications, just because some of the limitations of Lambda, it worked a lot better with Dynamo where you have a HTTPS connection. It works with IM, provisioning, billing, all sorts of things, rather than a relational database.

So I'm figuring that stuff out and I'm doing it wrong a lot of the time, and then I see Rick Houlihan give a talk at re:Invent. Rick's been on your show before. It just blew my mind and I watched that eight times, and then I came out with this site, The DynamoDB Guide of just like, "Okay, this is how I understand Dynamo," because I didn't understand it correctly, and it just snowballed from there.

DR: That's awesome.

Alex: People start reaching out and things like that.

DR: That's super cool. Well, I'm a charging girl. We're going to talk about telco charging today.

Alex: Sure.

DR: What brings you to the podcast here is, telco charging, and especially the charging engine, is a high volume, transaction heavy application where speed really matters. And so the database you pick is not a throwaway decision. It actually really, really matters and it's an important technical decision in the design of the application. For example, at Totogi, where I'm acting CEO, we picked DynamoDB and the other competitors are using other types of database approaches and so it's a little bit of a religious war, because everyone thinks everyone else is doing it wrong. And so everyone's like, "Those guys suck and ours is better and this is our secret sauce."

I wanted to just have a little dialogue about the pros and cons of these different approaches. There's basically three approaches going on in the charging market today. Some of the vendors have selected an in memory data store approach. This is a proprietary database in one of the competitors, for example. Then there's, of course, tried and true traditional relational databases like Oracle. I think everyone's pretty familiar with that SQL lead approach. And then there's us at Totogi. As far as I know, I think we're the only ones that are using a noSQL, single table database, and obviously, as I mentioned, DynamoDB as our selection.

So I want to talk with you about the trade offs of these different approaches, especially when it comes to charging. Let's first start with an in-memory data store. What do you think about this approach, about some of the things that you should think about, especially when it comes to charging?

Alex: Yeah, sure. So in-memory is super interesting. I usually see that in caching type use cases. With in-memory, you're going to get a lot more speed than a more disc based database. The thing I'd worry about there is just durability, right? At some point you have to get that to disk, and when are you going to do that? Are you doing that on the initial write? Are you doing that asynchronously, and then can you lose data and is that acceptable? I'd worry about the proprietariness of doing that. If you're a telco company or a charging company that's building your own database, that's a lot to bite off, in addition to all the other stuff you have going on.

DR: I think that's actually a really good point, that proprietary nature of that database, because charging is a tough problem in its own right, and so, that's not necessarily a database specific thing, but any proprietary technology when you're selecting a vendor, making sure that they can fund the innovation. This is typically a 10 year decision when you pick a charger.

It's right in the heart of the network. It's right in the heart of monetization, and so it's not something that you swap out lightly. You want to make sure that the vendor you select has a good runway in funding and can manage that.

Moving on to everyone's fallback position, which is the relational database. Again, very popular, especially Oracle's one of the largest tech companies in the world and obviously rest heavily on those laurels of a great database. So tell me about the advantages and disadvantages of using maybe a relational database in a high transaction processing environment.

Alex: Yeah, absolutely. Relational database, very common, probably doing 80, 90% of applications out there, not cloud charging, just all the applications out there.

DR: Yeah. Very popular.

Alex: So a lot of people use a relational database. The great thing about it is it's very flexible, right? You have this normalized abstracted schema and then you think how I want to access it later down the line, maybe you add indexes and things to make certain things faster. It's easy to ask different questions, but in this particular world, talking OLTP, we're talking extremely high read and write volume against this. Now that the relational stuff is starting to work against you a bit, you don't need that much flexibility.

You know the exact access patterns you're going to have. Initiate a call, end a call, send some data, different things like that, so you don't really need this generic one. What you need is something more designed to the task at hand, and at the scale you're talking about, with cloud charging, you're going to have to get rid of some of those features of relational databases anyway. You're probably going to have to be sharding across multiple different nodes here. You're probably not doing joins. You're probably not going to be doing aggregations or other things that relational databases can do, but some other noSQL databases can't do. At that point, you're replicating a noSQL database except your-

DR: Except for you're not.

Alex: Yeah, exactly. You're dragging around all this other stuff and you're managing it on your own, and you're hoping someone on your team doesn't use a feature that they shouldn't use and pull the whole thing down and all sorts of things like that.

DR: Well, I think those relational databases, like you mentioned, it's a little bit of a double edge sword, right? That flexibility sometimes bites you in the butt. Can I say that?

Alex: Yeah, totally.

DR: Where it's really easy to modify the schema, especially on an individual basis. This is my database, and so let's add an extra customization or extra field. And so that flexibility comes and bites you because people do it. They’re like, I can, and so I will. And so the minute you start doing that, that starts to impact your performance. It impacts your ability to quickly upgrade, and sometimes you need to take away those tools because people use it and it hurts you. It hurts you performance wise.

Alex: Totally. And that's what we saw with amazon.com AWS generally, they were really hitting the limits of what they could do in a relational database in Oracle. They were having issues on Prime day and Cyber Monday and all that stuff, to the point where that's where they designed Dynamo internally. They then productionize it and make it DynamoDB on AWS. But, if you want that consistent performance, if you want to be able to scale linearly to whatever you have, you're probably going to need to use a more noSQL database like that, or really, abuse your relational database in different ways.

DR: Well, yeah. I think Oracle's going to hate my podcast today, but Amazon, very famously, switched off their Oracle databases. I think there's a video on YouTube, we'll put it in the show notes, where they were celebrating turning off the last Oracle database. I mean, it wasn't so much like, "We hate Oracle, we're moving off," but more of a, "Impacting our ability to scale," and, "We need a different answer." We were just chatting about this before we started recording, which is the number of transactions that they now handle with DynamoDB for Prime Day, which is when people are ordering on amazon.com/retail. It's hitting that DynamoDB, and I think you mentioned to me, it was 89 million requests per second or something?

Alex: Yeah. At peak for Prime Day, across those amazon.com services, they peaked at 89 million requests per second against two DynamoDB. They did trillions of total requests across Prime Day. It's just quite amazing the volume that they're able to handle there and not have downtime issues.

DR: Well it's super epic, so again, charging is, the database really matters in what you select. We were really thoughtful and intentional, when we were looking at Totogi. We wanted to build a charging system that could literally handle the world's charging, right? If Vodafone became a global provider in every country. Could you handle the world's charging? We're like, "Number one, you need cloud. Absolutely." You can't build this in someone's data center, but you also need a badass database, so we went with DynamoDB, and so we approached you. That's how we know each other. We approached you to get your thoughts on our design. So third, now let's talk about DynamoDB. Why do you think noSQL, non relational, single table database is like chef kisses, awesome, for charging?

Alex: I think there's two reasons to talk about. One is specific in noSQL, and then one is specific to DynamoDB. So talk about noSQL. With a relational database, it's generally kept on a single server and you just have to scale up that server, but you reach the limits of any particular server size you can have so, at some point, you need to shard your data across multiple nodes, by some particular key, customer ID, or something like that. That's what DynamoDB is going to do. It shards on this partition key, very transparently, under the hood for you, and it allows you to linearly scale your database.

No matter what noSQL database you're using, whether it's Cassandra, DynamoDB, MongoDB, they all have that basic sharding setup. But I think what's interesting and unique about DynamoDB is its proprietary to AWS, and it's offered as a service, but it allows them to pool all their different customers across their infrastructure, within a particular region and share that capacity, and they can just build much higher level tooling on scaling up, scaling down your different databases. It's not like a single tenant instance based database, like you would have with a relational database or even with Cassandra or MongoDB, where they're going to be putting that on an instance for you, you're not sharing with neighbors. But then if you want to scale up, well, now you have to spin up a new instance, reshard your data, all those different things.

Dynamo's sharing that across all their customers in the region, so even if you double, triple, 10 X, 100 X your traffic pretty quickly, Dynamo can scale pretty rapidly to handle that and you're going to be a drop in the ocean compared to all that's happening across their entire region. If you look@amazon.com/retail, the other different huge customers they have, all that stuff, they can handle that pretty transparently.

DR: Yeah. I hear how people are always like "Pew, pew. You guys suck. DynamoDB isn't the right thing." One thing they bring up all the time is, "Well, DynamoDB is not ACID compliant." So what do you think about that?

Alex: I mean, two things, I would say. DynamoDB does have asset transactions now, they've had those for a couple years and you can do that. But also, you are doing extremely high volumes of operations, like you're going to be doing in cloud charging. You probably don't want to be doing a lot of transactions, no matter what you're doing. You're going to try and limit that just to avoid contention on groups of items, things like that. So Dynamo, if you're working on a single item at a time, you're going to get those asset characteristics that you want, so you can model your database in a way to handle that stuff.

DR: I think something else that you mentioned there was, the cloud nature of DynamoDB. In our case, with Totogi and with DynamoDB, this is a cloud database, right?

Alex: Yeah.

DR: The resiliency, the fail over, the scale up, the capacity is all managed by Amazon as a service. And so what's so nice about that, is we literally just riding coattails of Amazon, clearly with 89 million requests per second, right? I mean, this thing is internet scale, and so I think that's another really great point on DynamoDB that you made there.

Alex: Just one little anecdote to bring that home that Rick Houlihan always mentions. He shows this one performance chart of a company during the super bowl. They had run a commercial, they were expecting a ton of traffic and what that looked like, and they were seeing, that actually their performance got better during this spike because the shared request [inaudible 00:13:25] was able to cash some of their configuration information. So, even though traffic is spiking to this enormous amount, their performance actually got better rather than degrading, which is just pretty wild, if you're thinking about that for a database.

DR: And that is what I think telcos worried about with capacity in their database, just to give you a little bit of history. Typically, these applications are installed on premise. There's a team, a set of people that do a dimensioning exercise. How much capacity are you going to need over the next three to five years? What are the spikes you're going to see? How are we going to fail over? And you buy all these servers and months go by and what really strikes fear in an IT person's heart is that, we did the math wrong on capacity.

Network usage ebbs and flows, right? It could be anticipated. Mother's Day coming up, and so lots of people are going to be calling their moms and staying on the network, talking for a while. But then there's those unexpected events, like a flood, or some sort of natural disaster, where people are trying to get a hold of loved ones and you have that spike. Whenever you haven't pre-provisioned the capacity to handle the demand, you'll have an outage, and that's where customers are really upset at their telco of like, "I couldn't get through."

So that's what they're really fearful of is that miss, and like you just said, with DynamoDB and Amazon, it gets better with scale. we don't have that problem, and we don't have to pre-provision it. I'm not paying for it to be up and running. I just scale as demand increases, and so it's awesome.

Alex: Yeah. Once that Superbowl ends, you turn it back down to your usage and you're not paying that peak usage-

DR: It's so great.

Alex: ... whereas, like you're saying, the on-prem one, you got to get those servers six months in advance. You're probably way over provisioning, because you don't want to go down on the Super Bowl, or Mother's Day or whatever we're saying.

DR: Yeah, exactly.

Alex: Now you've got a bunch of servers that are doing nothing the rest of the year.

DR: That you paid for and they're just timing out, right? You got to refresh them, no matter what, in five years or whatever it is.

Alex: Yep. And someone's going to pay for that down the road and it's going to be your customers.

DR: Yep. It's crazy. Well, I was looking at your background and what's crazy, was that you were not a CS technical guy from the start. You took a little bit of a turn at the beginning, and you were a corporate lawyer. It's true. Tell me about that change because, making a change from being in a legal profession all the way over here to talking databases all day long. Was that a scary move for you?

Alex: Yeah.

DR: How did you decide to make that leap?

Alex: Yeah, that's a great question. I definitely don't have the CS background like you do, but I would say that law actually isn't that much different than programming often. Especially corporate law, you're basically following all these, if/else statements to figure out how things work like that. In terms of practicalness of making that change, my last year of law school was when I started to get into programming and I was working on the startup with two buddies of mine. And I worked as a lawyer for a year and it was fine. I enjoyed it. But my wife was also a lawyer. We had a couple kids, it was a really hard lifestyle. I mean, I really like programming a lot more. My brother-in-law was working at a startup here in Omaha and I made the jump over there.

DR: That's awesome.

Alex: I just love doing that work a lot more. It's so fun to just be able to create stuff out of nothing at any given day where, corporate law, we've only done it nine months, but it was starting to feel repetitive. Every M&A transaction is the same. Once you've done one, you've done them all.

DR: Yeah. I mean, I did the same thing. I have this computer science degree from Stanford, arguably one of the most valuable degrees in the world. So I did authentic CSE stuff for, let's say, maybe five or six years, and I moved into HR, which no one does ever, but I did. My mom was like, "Are you nuts? You went to school for this thing." But I look back on my lifetime of experience and it's actually made me who I am now.

Alex: Totally.

DR: It's made me a better leader, and so hopefully, you're having that same experience.

Alex: And I'm sure you, being a leader of a tech company, just being able to understand the technical concept and what's possible, what's not possible, what's hard, things like that, is super useful. For me, having a law background, it's a lot of, reading a ton of information, figuring out the important things and being able to write it out clearly of what matters, and I think that was super helpful in writing The DynamoDB Guide, in writing The DynamoDB Book and then also in design reviews and things that I do now. It's a lot of writing, a lot of communication. A lot of the skills are the same there.

DR: But I think you're right. At the end of the day, it's just following a bunch of rules.

Alex: It totally is. Yep.

DR: So anyways, Alex, it was so great to have you on the podcast. Thank you for your insights and thanks again for coming on.

Alex: Likewise, thank you DR. I loved it.

DR: Awesome. Stick around because we're ending each podcast with a Telco in 20 takeaway. I have 20 seconds to tell you something you need to know.

Not to beat a dead horse, but to beat a dead horse. Like Alex said, one of the reasons why Totogi built on DynamoDB is so great is because it uses a cloud database, which the other guys do not. That means besides getting awesome technology built into the design of the actual database, you get all this other cool shit, like you don't have to support the database stack. AWS does it for you. It's available in 26 geographic regions, and as they build up more regions, they have eight more planned. They proliferate that service.

You can get it anywhere in the world and it has scaling resiliency and failover built right into the service. On the other hand, just compare that to the way telco has traditionally deployed charging. You have to stand up each stack individually, procure all the hardware, buy the database licenses, set up all the extra capacity, and don't forget to think about resiliency and failover.

Not at Totogi. Because we use DynamoDB, our database has all of this built in, provided as a service by Amazon. It's totally badass. Go read about why Gartner thinks cloud databases are the future much to Oracle chagrin. We'll put a link to that in our show notes. This is the future, and if you want to talk to me in person about these amazing concepts, I will be up in Windsor next week, flapping my gums with the TelecomTV boys at DSP, The Davos of Telecom. Hello, so cool.

It's going to be a great event. High level discussions with executives. There will be a fun party. I'll be there. I can't wait. So jump on a plane, and when you land in London, DM me on Twitter at TelcoDR. You can also connect to me on LinkedIn, and if you have any questions about Totogi or charging, you can also give me a call, or WhatsApp me at 925 Telco Dr. If you love this podcast, subscribe on Apple Podcast and follow us on Spotify and don't forget to sign up for our totally awesome, 50% open rate email newsletter, at telcodr.com. Later nerds!