CR033

Good data plumbing with Rick Houlihan - MongoDB

CLOUD REALITIES
PODCAST
LISTEN NOW
Capgemini's Cloud Realities podcast explores the exciting realities of today and tomorrow that can be unleashed by cloud.

CR033

Good data plumbing with Rick Houlihan - MongoDB

Disclaimer: Please be aware that this transcript from the Cloud Realities podcast has been automatically generated, so errors may occur.

[00:00:00] Just root your dings through your headphones. That's what I do. So the dings come through ear, not through the speakers. Root your dings. Root your dings. You need to root your, dings it out.

Welcome to Cloud Realities, a conversation show exploring the practical and exciting alternate realities that can be unleashed through cloud driven transformation. I'm David Chapman. I'm Sjoukje Zaal, and I'm Rob Kernahan. And this, and I'm Rick. Oh, I didn't know. You don't worry. I, I'm getting to you, Rick

And this week we're going to be talking about the importance of the underlying technical plumbing when it comes to data. Data gets [00:01:00] talked about a lot. It is mission critical in terms of creating strong, personalized experiences, but what goes on underneath the covers, how does the engine work and why are some.

Decisions better than others for not just the efficiency of your systems, not just the speed of response and the ability to bring data together, but also for the planet.

Joining us this week is Rick Houlihan Field, CTO O for strategic accounts at MongoDB. Welcome, Rick. Great to see you. Do you wanna introduce yourself and tell us a little bit about what you do? Sure, thanks Dave. Really happy to be here. As you mentioned, I'm the field CT o for strategic accounts at MongoDB.

Long history in the no SQL technology category. That's a little bit what we're here to talk about today, I spent many years at AWS. I left as the worldwide leader for techno for no SQL services, at AWS. And this is actually my second stint here at MongoDB, really happy to be here talking about pet, technology.

I'm [00:02:00] very passionate about and, yeah, it's a little bit about me. Over the course of the season, we've talked a lot about the importance of data becoming a data-driven organization. Obviously the absolutely critical and unbreakable relationship between data and AI and how that's evolving and the importance of dataset.

What it would be great to start with, Rick, is maybe a step back. And let's just talk about database technology a little bit. Maybe give us a perspective from your bridge on the evolution of database technologies and where we're up to at the moment. Sure. So, um, Yeah, I mean, I guess that's, that's a really good question.

I mean, honestly, database technologies evolved in many ways over the years. We weren't really born with a, a relational database, so to speak, right? The, we evolved many technologies over the years, and one of the things I like to refer to when I talk about this is, is really what drives change in, in data processing.

And it comes down [00:03:00] to, you know, what I call data pressure, right? The ability of the system to process the amount of data that we're asking it to. Process in a reasonable time or at a reasonable cost when one of those dimensions is broken, you know, we're going to invent something. And we've invented many things over the years.

You know, the relational database was the end of a long line of innovations, and it's one that's carried us for four year, 50 years. But, you know, as we've seen, the efficiency of processor technology start to decline and the, and the, and the effectiveness or the. The relevance of Moore's Law, you know, starts to, fade.

, we're starting to see the same types of problems we've seen many times over the years, right? Data pressure, you know, is the system able to process at a reasonable time or reasonable cost? And, and that's where no SQL databases have come in to play here, to try and address, you know, some of these problems.

And for those listening that maybe don't even know what SQL is. Never mind no sql. Perhaps you could help us with a few distinctions of relational databases and how no SQL fits in. Sure. [00:04:00] Absolutely. So, you know, in the, in the late seventies, um, a guy named Edgar Cod, a very, famous technologist, very smart gentleman, invented the normalized relational data model, which is really the foundation of, of modern database technology.

At the time when, when Mr. Cod's team built this technology, his mission was to enable people to ask questions of their data without having to write code, and at the. You know, up until the relational database and the introduction of the relational database answering ad hoc questions, you know, running those ad hoc queries that we're also used to, you know, creating reports, really involved a lot of engineering, because the data structures on disc were.

Were tightly coupled to the access patterns with which they supported those O L T P workloads, right? The, the transactional workloads, that they, that they supported. But the business, you know, we always need to ask questions, right? What were the, you know, what was the average sale price in the last 30 days?

You know, who, you know, various reports and whatnot, we need to run. [00:05:00] And, and that was a real problem. So the relational database was really built with that purpose to allow people to ask questions, and over the years, you know, as processor technology evolved and became more efficient, the biggest criticism of the relational database, which was that the, the normalized relational model was inefficient compared to, you know, the hierarchical management systems of the day, it kind of went away, right?

Because the processors got to the point where we didn't notice the pain, so much. You know, again, as we've seen in the last decade, this kind of slow down in the efficiency of Moore's Law or the relevance of Moore's Law. These same questions now are starting to rear their, their ugly head again, so to speak.

As people start to look at the relational database and say, you know, it's kind of expensive. And that's, that's exactly where, you know, the no SQL databases start to come into play because they, they take, they take notes from, you know, the, the old hierarchal management systems, and they, and they allow.

Users to be able to ask those complex questions because the, today we have the C P U, [00:06:00] bandwidth and, and performance and the storage throughput to be able to run those ad hoc queries on de-normalized data. You know, back in 1970, 1980, when the relational database was introduced, that was just not even a concept that was feasible.

Today, not so much. Right? So there's, there's things we can do to optimize the model for those high velocity access patterns and still be able to run. Those ad hoc queries and generate those reports, albeit maybe not as efficiently, but that's not as important when we, you know, the, it's more important to optimize those high velocity access patterns, right?

The things that run thousands of times a second, not the thing that runs once a day, right? The thing that runs once a day, I just need to be able to do it, I don't necessarily care about how efficient it is, but if you've running a transactional workload that's hitting, you know, dozens, hundreds, thousands of transactions per second, Now that's where you're spending your money.

So optimizing, you know, the data model for those patterns, is very important and that's what no SQL databases do. So perhaps let's look at why that's interesting and strategic, [00:07:00] but more through the eyes of a forever business user. How does that show up to me? Say, in today's world where I may have, you know, multiple data backends, I may have a, I may have a data lake, I might be trying to pull information together from multiple different sources.

How does no SQL actually help me with that? And, and what's the material difference? And that's a really good question. So, you know, traditional no SQL databases, they're really not well suited for those types of workloads that you just mentioned, right? The data lake, the business analyst, and, and this is where MongoDB really starts to differentiate itself from the pack.

I mean, one of the things that we recognized early on is the developers, they need more than a database, right? They need a data platform. They need a technology that allows them to run and scale efficiently, but still be able to support. These, you know, ad hoc queries, right? The, the reports, the reporting workloads, the ola, you know, workloads, MongoDB strives to give developers this because we have integrated best of breed technologies into our platform [00:08:00] that go beyond the traditional no SQL database.

, but you know, the core of the no SQL database still exists. It's the idea. Behind no SQL is that, you know, the data model, an abstraction of a da, an abstract data model does not support a well-known access pattern as efficiently as a finely tuned or tightly coupled data model. Right? This is what we actually knew back in the seventies when they, you know, were first developing the relational database.

The biggest criticism was that. You know, the cost of the join, it's expensive to join tables, and that cost is felt in cpu, you know, back in the 1980s, early eighties, that was expensive. And so the idea for using, of reusing a relational database for O LTP workloads, you know, that didn't start happening until.

You know, the, the processor efficiency started coming into play in the late eighties, early nineties. Today, again, we've seen now that the, the, we're kind of a flattening of, of processor performance over the last decade, and, and we're starting to see a separation [00:09:00] of efficiency between no SQL workloads that are tightly coupled to the access pattern.

And relational database workloads that are agnostic to all the access patterns, right? The abstract data model of the relational database requires a lot of cpu,, it, it, it burns a lot of CPU time to produce the results. And in the no SQL database, essentially we're trying to store the data in the same format.

That it's actually accessed in. So essentially we don't have to join the data, we don't have to restructure the data to deliver the, the query result, not nearly as much as you would have to in the relational database. And this is really what drives that efficiency. As you were just going through the seventies and eighties there, Rick, I saw Rob nostalgically smiling.

So staring, remembering my childhood. What was it like in the seventies when you were trying to, lower the cost of relational databases? It was a simpler time, Dave, back when computers made sense. Yeah. Yeah, yeah. I love it. It's totally true. Right? Just a lot less moving parts, right? Yeah, exactly. Exactly.

Yeah. [00:10:00] Right. So what that means then, for me as a business user, it means what? It means faster response time on query. Does it mean sort of more flexibility in query? Just to give you some context here, some of the workloads that I teams, I worked at Amazon retail on the project they called Rolling Stone, which was a, a, a mass migration of relational technology, of relational workloads to no sql.

And I'm talking to some of those teams today, five years later, and they're telling me today they're just starting to hit the same level of database spend that they had five years ago. And they've grown 3, 4, 5 x. So if you think about the cost efficiency it, that's where it comes home for the business user, right?

You don't wanna spend money on things you don't need to spend. So why spend money joining tables when I could use a no SQL database that does it more efficiently? That's the bottom line. I think when I characterize it in my head, the no SQL database came out of a requirement for a different use case.

Like you said, this is too expensive from a technical perspective to do the use case, which was retrieve an object quickly when I had the exact point of [00:11:00] entry, and I knew exactly what I wanted. So keyed something very specific, and then over time I've seen no SQL capability. Grow and start to introduce topics and extra capabilities.

So this Venn diagram of the capability is starting to merge again, and I'm keen to get your view on as the database technologies evolved and mature, they're blurring the lines of what they can do. And maybe selection isn't as easy as, I've got this use case, I'll use this database. Whereas they're all sort of offering all things to all people.

They'd be interested to know what you say going from that use case world to, I can do everything type. Position, is there still a way to think about what's best for what scenario? What's best for what use case? Yeah. So, this talks to you the, the core of, you know, the developer data platform, which is Atlas.

, the bottom line is developers have been asking for decades for more than a database, right? I mean, the advent of index of search technology with solar and, and elastic and, and lucine, which backs both of those, you know, the reality is that a database today is a lot [00:12:00] more complex, like you said.

Earlier, it's a lot more moving parts and you know, developers have much more rich requirements, so, you know, in order to be able to offload the heavy lifting, so to speak, right? Developers don't want to focus on moving data around between disparate systems and learning multiple APIs to be able to, you know, support their workloads.

You know, we built the developer data platform, which is MongoDB Atlas, and it's the, it really is the core of your question, right? It's, you know, we used to use NoSQL for these certain category of workloads, but we've. API evolve and expand, to address, you know, more than just the traditional no SQL workload.

So, and today what it comes down to is that if you wanna drive efficiency in the system, then you, what you wanna do is model for your common access patterns, right? The high frequency patterns, the things that you run, you know, at, at high velocity. We still need to be able to support those other patterns and that's where the API extensions have come in over the years.

Right, we support, you know, SQL APIs exactly. For that. I think there was another question. And, and I was, I was going to say [00:13:00] it is a very good point you've just hit on there, which was the database used to be a very specific domain where the database expert Yeah. Would write your psql and you would call it.

And now the capability of I'm a developer is merging with the, I also understand the access pattern to the database, that full stack view. Cause it used to be, There's a database expert and there's a coder, and the two have to work together to create a system. And now with these platforms developing and they're ever increasing sophistication, I see it that the developer is now able to take or come down into that data domain and work in it a lot more because the platforms have become more sophisticated.

I totally agree with that. There's, there's enough abstraction to say we don't need a very specific database person anymore to tune it to within an inch of its life. Well, if you think about it, the relational database and the normalized data model is really an abstraction between the application and its data, right?

I'm going to have application layer objects that we've defined that developers work with to, to craft these objects that are used in the application. And when I [00:14:00] store those objects in the database, I'm going to break those objects apart into little bits and pieces and store them as rows across tables that, you know, eventually get joined back together to reproduce the object that the application is.

I mean that's a lot of, that's a lot of data movement just to, to deconstruct and reconstruct the objects. You know, the developers know what objects they need. They built them. Yeah, they designed them right. And they know how the applications use those objects. So honestly, it's better to be able to tightly couple the database to those object data structures because that's essentially is exactly the mantra in no sql, right?

Data that is accessed together should be stored together either in a common collection. Or an embedded document and, and what, what, and the choice between the common collection or the embedded document approach really comes down to how is the data actually accessed? I, is it accessed in bits and pieces, or is it accessed in big, large chunks?

Right? If it's bi in bits and pieces, we're going to want to create small documents that are linked together with indexes. And what we found at Amazon was really interesting when we broke [00:15:00] down the access patterns of 10,000 services that were running on those relational database clusters, right? 3000 Oracle server instances, 70% of the access patterns were for a single row of data.

On a single table, another 20% of those patterns were for a range of rows on a single table. And at first we kind of said, well, this is crazy what's going on here? But when you think about it, it actually makes a lot of sense because O L T P workloads are often extremely right, heavy. They, they al sometimes the data almost never gets read.

It just gets written. Over and over again, and every now and then somebody will read it when they're troubleshooting or they wanna see an order status or whatnot. But most of the time the data is really being worked with in those little bits and pieces. So in the no SQL environment, essentially, sometimes creating those big, large embedded documents is not the most efficient way to work with the data and the people who know how to do that.

It's not the DBA. Right, it's, it's the developer because they're the ones that wrote the code so they know what data they need. So once we kind of get developers over that hump of thinking about how to store the data the same way the app uses it, it's funny how fast they move [00:16:00] because they're really working with familiar.

Structures now. Right. Things that they don't have to decompose. Right. One of the points I didn't wanna lose back there somewhere, Rick, was the point of efficiency and cost reduction and the point you made in the Amazon case study about the fact that, you know, it's taken them five years to get back from a, from a processor consumption point of view to where this started.

And obviously there's been a, a, you know, maybe it's not exponential, but huge growth in the data sector during that period. So that, that efficiency is amazing, but there's a sustainability point in there as well, right. You, you, you're burning less energy when you're doing that. That's exactly right. And I say that all the time.

You know, if you think about the cost of the relational database, it can be measured in global warming. And that's no joke. I mean, the amount, it was the biggest criticism of the relational database in the normalized data model at the time was the CPU efficiency of the joint. And we just kind of ate it over the years as, as Moore's law kind of made that pain.

Go away. It never really went away. Right? We, if you think over how many, how much infrastructure has been deployed just to join tables [00:17:00] and, and, and this is not my thinking, right? I mean, this is not like new thinking either. Edgar caught himself, called this out in his, in his, you know, seminal paper, a relational model of data for large shared data banks.

He talks about the efficiency of normalization on storage for deduplication of data on disc, for transmission of bulk data across systems. But he also talks about the efficiency of denormalization, that introducing strong redundancies into the stored set will reduce the load on the CPU U for some queries.

He says some queries, but the reality is what are those queries? Well, those are the ones that are running thousands of times a second. Those are the ones we care about, right? The other ones. I don't care how much CPU U it takes to carve through my de-normalized data and produce a summary that I'm looking for.

Well may maybe stepping up the stack a little then and coming at data from a slightly different angle. So one of the things that we've talked about in a number of episodes of the show this year is, For all of the talk of of big data searchability [00:18:00] and the most recently large language models, organizations are still actually struggling with how to become a data-driven organization, like the leverage of data.

The data model, like the overall data architecture, any kind of master data architectures that are, that are running, we don't seem to have made that any easier yet, it seems to me. So I wonder what your perspective is on, you know, how easy might it get for organizations to leverage their data over the course of the next five years or so?

Oh, that's a really good question. You know, they, it we're talking about, A, a transformation, right? A technology transformation in the industry, in the enterprise. And, and what we ran into at Amazon was,, it was really mixed results with that exercise. And that was, and, and part of that was developers trying to use a new technology, essentially running through the same problem that happens, you know, endlessly, is that we don't really know how to use this technology.

So what are we going to do? We're going to try and use it the way we know how to use a database. Well, It's not [00:19:00] the right way to use an o SQL database is to, is to use the same data models, right? So what we needed to do was get developers over that hump, right? How do you get, developers familiar with this new technology in real time so that they can be effective using it?

And, and what we found was, that it's a little different. Teaching people no, SQL is not the same as the relational database, right? As we all know, in the relational database, you have a lowest common denominator, right? We have, you know, parent child relationships, many to many relationships, and we can map these things out in the e r d and we can, you know, essentially define a logical data model that maps directly to our stored.

Data model, right? In no sql, that's not the case. Essentially, what we still want that logical data model, but then I need to understand how am I accessing the data and then I'm going to go ahead and craft a document structure to map to that, right? And so in order to be able to get developers familiar with this process, what we found was it's not about doing workshops and, and brown bags.

That you can get away with that with the relational database cuz it's the lowest common denominator. And I only have [00:20:00] to teach you one thing and then you're off and running. Right. And, and honestly, all of us around the table here, we can look and say when, how many times did someone have to explain that idea to you, right.

Of a parent-child relationship? Right. Once you never had to hear it again. Right? Because it was. You know, it was very fundamental. Reality is no sql, it's about the access pattern, right? The access pattern defines the data model. So what we actually did at Amazon was we stopped the whole idea of the traditional developer relations approach.

No more brown bags, no more, you know, workshops and, and hackathons. Instead, what we did was we focused on what we called design reviews. We would sit down with the team for an hour, we talk to them about their workload. We map out a document data model or an object data model structure that works for them, and then they're off and running.

So it's, it's, it's kind of like once you show the developer how to solve the problem they're working on with this technology, guess what? It's like you get that light bulb moment. That's what we always, I call it when we see around the table, someone sits there and goes, oh my gosh, I never thought I could, I could actually model data that way.

So the key to the problem does that get [00:21:00] around the torturous process that enterprises in particular have been wrestling with for. I don't know, the last 20, maybe even 30 years of They've got organic growth of data. Yeah. They've got data in pockets all the way across their organization. Yeah. Like D does what you are describing get around the.

Torturous process of normalizing all of that, pulling all of that together, you know? Right. No, I mean, I, I wish I could say yes. I, I don't know. I mean, it might make it easier. Right. I'm just going to say, Rick, I think there's probably a business in it. If we can have, you just had a, a unique idea, David. You know, there's always going to be And that's copyright cloud realities productions, by the way.

Exactly, exactly. No, there's always going to be a bit of data architecture involved here. One of the things that I think was said earlier how we don't no longer the developer can become the dba. You know, I would say to a certain extent, yes, they can help inform the model more, but the DBA and those skills are still.

[00:22:00] Heavily required even with no SQL databases, right? To, to be able to recognize which access patterns and how to model the data for those access patterns. A lot of that skill set is actually very similar to the skills that DBAs use when they optimize a relational data model. Right? Right. What do you do now?

You sit there and you look at the query query analyzer and what's burning this, where to put the indexes and all that. The difference is that with no sql, it's more important that the data model is right. Then it, then it's, it's difficult to correct after the fact. Right? So you're going to see it's actually good because you get the data, you get the DVAs more engaged in the beginning of the process instead of after everything's falling apart.

Right. And, and theoretically, hopefully they prevent the building from burning down. Right? So I would say that those skills are still hugely valuable, with no SQL database. It's the, um, the classic it rise and fall and rise and fall again, isn't it? So there was the. The database domain where the specialist went in and then we said, oh, the developer can do it.

It's dead easy, abstract, don't act as bad. And then we went, hang on a minute. That's not fix the problem. We need data centric [00:23:00] people to come and fix it again. So it's the, um, it's the, it's, it's like we were talking earlier on other podcasts, it's like enterprise architecture it was in, and then everybody, we don't need enterprise architecture anymore and then realize that well actually there was a bit of a value in somebody looking at the whole thing.

End two end and maybe we should reintroduce it. I think data's gone through that cycle. It's come back and gone. No, it is a discipline and a skill we need. Yeah. Well, you know, we do need to understand how data moves around and how, and, and, and honestly, a lot of what we do in my team, we, we sit with these, with development teams.

We break down their workloads, but a lot of it is the fact that we really understand how data moves inside out in, inside the MongoDB. Right. Knowing the internals of the database. It's the same thing with Oracle or Postgres or anything, right? You need to understand how row column storage works to be able to optimize the data model, right?

And, and developers don't necessarily understand things that deep. Right down in the database, right? They might know how to create a normalized relational model. They could probably create foreign keys and things like this, and then they're done. You know, I mean, maybe, maybe if you're, if you're get a really [00:24:00] smart developer, they'll understand what they need to index, right?

But you know what I mean? It's like, but developers themselves are not really good at optimizing data models, right?, it takes a data mind to do that. And to do that, you gotta need to understand how the system uses the data, accesses the data, stores the data, things like this, so again, those skills are just gotta have them.

We gotta have 'em. Although you do touch on a point about the continued abstraction and sometimes you need to understand what's going on inside the system. Yes. To best use the system, I have this fear that it all go dark and everything breaks. And it's going to be a three and a half inch floppy disc that comes out of a drawer, restarts everything, cuz they work out that's where the key was and they have to rebuild it all back up from there.

Perhaps to bring today's conversation to a bit of a conclusion. Then I, I wonder what advice you could give to business owners and business decision makers in terms of making some smart decisions around how they leverage their data. So I know that. A lot of organizations are, are looking at the data and the way [00:25:00] that we talked about earlier.

It's like trapped in little pools all across the organization and they're trying to bring it forward to drive insight. What is the fastest route through that, do you think? Okay. Yeah, no, absolutely. This is, you know, the biggest thing and one of the biggest values of MongoDB, and it's always been one of the mission statements of the company is making it.

Incredibly easy for developers to work with data. Okay? And that that is really what the Atlas Developer data platform is all about, right? We have integrated feature functionality in there that pr, that that means that developers don't have to spend their time hooking up pipes and plumbing to other systems to be able to get the features and functions they need.

Right. So if I were to say anything to business owners today is to look at data platforms that have the integrated functionality that your developers are asking for, right? They want full tech search on top of their data. They want online archive, right? They want federated queries. These are all things they want.

Vector search, right? These are all things that if you wanna move fast, then the database you're using better provide those things because [00:26:00] otherwise your developer's going to spend half their time hooking up pipes and plumbing managing ETLs synchronizing data across multiple. Disparate silos just so that they can get the functionality they need.

This is the mission that that, and this is the, the mission of MongoDB and it's what we heard many years ago. I, when I was at AWS, I advocated for exactly this type of integrated platform approach, right? The, the mission of the cloud in the beginning when it was infrastructure as service was don't focus on undifferentiated heavy lifting.

Right. It's not valuable to your business to understand. Right. Exactly. And what we want is we want. Teams to be able to use the tooling to be able to deliver the functionality, not have to build the tooling to deliver the functionality. Right? And this is where MongoDB really accelerates and separates from our competition.

We have focused on delivering that mission. It's not purpose-built silos that need to be stitched together by developers. It's a tightly coupled, fully integrated platform. That accelerates the development of, you know, a process. And that's really, if I were to talk to stakeholders today, that's the number one mission.

[00:27:00] Make sure that the platform that you choose delivers the feature and functionality that you need, because otherwise you're going to be supporting multiple platforms and hooking all this stuff up yourself and spending a lot of money doing it.

We've been looking at this week, so each week I will do some research on what's trending in tech, and this week I want to focus on how can businesses get started with real time data and why is this important? So in today's digital services economy, many businesses are now service providers, and in this environment, businesses rely on real time applications to survive.

And according to research, 72% of consumers say that poor personalization decreases their trust in brands. [00:28:00] So how can businesses get started with real-time data? So first, you need to consider the poor data experiences that your existing infrastructure may be delivering. So, for instance, look into the current processes and what they mean to you and to your customers.

And next, you need to assess the data intensity of your existing applications. So consider the rate at which your data intensive applications are likely to increase in the year ahead and beyond, so then you will have a good idea of where you stand, where you're headed, and what infrastructure your applications require in the future.

And this infrastructure, this data infrastructure, will enable you to use real-time data and analytics to provide your customers with the experiences they expect. So a question to you, Rick, do you also see an increase in demand for realtime data ANA analytics, and do you have some advice for companies that want to get started with realtime data?

, no, I mean, absolutely. It's a, it is probably [00:29:00] one of the number one concerns that we see is real-time data processing stream, data processing, lots of workloads that, use MongoDB and leverage MongoDB, you know, for those types of, of data processing services, I guess. You know, a lot of, a lot of that really comes down to event driven processing.

, one of the things about Atlas MongoDB native, even enterprise version supports, you know, the chain streams, atlas supports, triggers, these are technologies that are used absolutely to be able to support or, or facilitate realtime data processing, event driven data processing. Um, I guess advice to customers that are involved in, in those types of workloads.

Again, it, it comes down to, you know, platform selection, right? Um, and, and honestly with real time event driven processing, the most important thing is about guaranteed processing, right? And, and so this is where I see organizations. Failing more often is when they try to build these types of systems themselves, [00:30:00] themselves that provide guaranteed at least once processing of those event streams.

You know, if you're, if you're rolling your own infrastructure to do this, you are going to be responsible for maintaining, you know, and guaranteeing the execution of those events. You know, if you go with the service provider that can provide that function triggers off of the chain stream, for example, with Atlas, or, you know, DynamoDB has streams in Lambda, most managed database.

Offerings do provide this type of functionality, facilitate that type of processing. So, They're going to provide those guarantees, right? That's not something that you're going to have to do. So if you are involved in real time event driven processing, then I would definitely look at managed service providers that can provide a level of guarantee around that data, about around those data events and the triggering of processing around those events.

Um, or you're going to be the one that's going to have to build that and guarantee it for yourself. So a lot more work, right? Good advice. Yeah, don't try to reinvent the wheel, but use what's already in store. Yeah. Yeah. I do like that point you make about, we don't trust [00:31:00] organizations that don't personalize around us as well.

I had a recent experience where, something went on on the.com website and the.co.uk knew nothing about me. Robbie, is this, is this some of your specialist websites? Unfortunately, no. David, this is, this is, um, one way you might, I would, I would advise you. Don't put your personal data into there, Rob.

No, no. Yeah. Don't, don't put you, this, this experience across them was, um, the same company, different domain. Simple. Hadn't shared the data between the two. I logged in, didn't have a clue about me. And the item I perched a virtual ticket wasn't there. At which point I went, where's it gone? Log in. Yeah. On the other machine.

And I didn't spot the domain change. And then when I spot the domain change, of course I worked it all out, but until then I was like quite confused cuz it was through the app that I downloaded that I couldn't find my ticket. And I'm like, How's this go? Where? Where's it gone? Rather frustrated. Yeah. Yeah.

And it was that experience of they hadn't connected, their data sets up between the domains to no single [00:32:00] view of the customer. Right. So you had, you had like a web app that had one set of data and a, and a and a mobile app that wasn't seeing the update. Exactly. And probably cuz it was like a pipeline in the back that synced those two things.

And guess what? Something broke or something. The queue backed up. Something was delayed. And, and this is again what we talk about. If you're building a system, then you want. A, a unified view of, of the data, right? You don't want data in different silos because you get into, you get into this kind of disjointed customer experience.

, you have one system that's aware of change and another that's not, that's not a good thing. Right? And so, yeah. Yeah. And my natural trust of that organization drop that day. Cause of course, hang on a minute, they've lost my ticket. Exactly, exactly. I'm going to have to fill in a form. I'm going to be on a contact center experience.

I have no clue how to get this back. Yeah. Yeah. Exactly. Yeah. Yeah. But no, it's, it's, it's a very good point about the, um, better data architecture gives a much better experience, doesn't it? That's right. And a single source. Right. I, this is one of the things, again, I, I, I love AWS. I spent a decade there. It was a fantastic experience.

Would never [00:33:00] take it back. I, I was never a fan of purpose-built for the exact reasons we talked about, you know, I don't want, as a developer, I don't want to have to glue up a bunch of, you know, disparate services over and over again to deliver solutions, right? I want my data platform to deliver that feature and functionality that I need, and, and I wanna spend time on ETLs and, you know, data consistency, data coherency, you know, types of issues and code for that.

Right. I mean, really, do you wanna have to write code that accounts for the fact that the data in this silo might be different than the data in that silo? No. You, you don't want to have to do that as a developer, right? Yeah. So it's just better. It's better, it's better for everybody if we have a single view of the data and an API that sits on top of that single view of the data that provides the functionality we need.

That's what we do with Atlas. Thank you so much for your time and insights today, Rick. I know I, I've learned a, an awful lot about the underlying elements of how important good plumbing is in this situation and the uplift you're going to get from the business [00:34:00] value in your data. And not to mention like, be kinder to the, the planet.

So thanks again. Great insights. Thank you very much. Glad to be here. Now look, we end every episode of this podcast by asking our guests what they're excited about doing next. And that could be, I'm off on holiday at the weekend and, you know, can't go fast enough. Um, or it could be something you're excited about in your, in your professional life.

So Rick, what are you excited about doing next? You know, um, actually coming to MongoDB, I was, I had a lot of questions about how effective the program that we built at Amazon retail was going to be, bringing it out to a broad set of customers like this across different, organizations. I had mixed results at AWS, and as it turned out, Um, you know, it was the, the, the, what was lacking was the, what I call the wood behind the arrow, so to speak.

We had 230 services at AWS and those sales teams, they didn't really care about anything but what the customer wanted, right? They'd hit their numbers selling on e C two. They didn't care about DynamoDB. Here at MongoDB we're all lined up behind one product, right? So what I'm really excited [00:35:00] about is extending what we've done over the last year and delivering.

For our customers over the next year or so and, and watching the program that I brought from AWS really flourish cuz it really is as we go through this exercise of working with customers and bringing this, what we call MongoDB Day experience, and, and running our design reviews. It's just so exciting to talk to so many customers about so many different workloads.

It really gets me up and gets me going every day. So that's what I'm, that's what I'm looking forward to. Awesome. Well, we wish you very good luck with that, Rick. Thank you.

Thank you so much for being on the show to our sound and editing wizard, Ben, and of course to all of our listeners, We're on LinkedIn and Twitter, Dave Chapman, Rob Kernahan, and Sjoukje Zaal. Feel free to follow or connect with us and let us know if you have any ideas for the show.

And of course, if you haven't already done that, rate and subscribe to our podcast. See you in another reality next week

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of over 360,000 team members in more than 50 countries. With its strong 55-year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2022 global revenues of €22 billion.

Get The Future You Want | www.capgemini.com