00:00:00
Let's do a little bit of introductions. So I'm Mike Lynn. I've been with MongoDB for about nine years. Prior to MongoDB, I worked with MongoDB as a consultant. I started the first user group in Philadelphia, MongoDB user group. Spent 30 years in development engineering and architecture. I've worked in the financial services industry, familiar with the challenges of scale and with using both relational technologies as well as MongoDB.
00:00:30
It runs in my family. My brother wrote the first book on the aggregation framework. He's the one who introduced me to the tech stack that MongoDB is included in. So that's me. I spent the last five years working on the podcast, having great conversations and doing marketing for MongoDB, a little less engineering. So I'm really excited to get back into writing sample apps and talking with customers like yourself to help you leverage the power of MongoDB. So that's me.
00:01:00
I'm Michael. I'm the account executive supporting Clear Channel. So, happy to be here. Kelly Rich. I'm a software engineer at Clear Channel. Pretty new. I'm kind of more on the beginner side of the MongoDB stuff. So, I'm here to learn. Awesome. Stanford Chang. Principal software engineer. Clear Channel Outdoor. Two different companies. But yeah, I've been... So, tell me. Clear Channel Outdoor. Are you on the same team? Oh, yeah, I'm on the same team. So, today we're talking about RTB for Clear Channel Outdoor.
00:01:33
Yeah, it's part of CCO, basically. Clear Channel Outdoor. Outdoor. Okay, gotcha. We're here talking about CCO. Okay, very good. Appreciate that clarification. Yeah, we do the out-of-home advertising. Okay, great. So, you see those roadside displays? Yeah, yeah. So, I am being a software engineer, of course, and been using MongoDB for quite a while. Okay, good. We started with version 1 or prior to that. Okay, sorry. Yeah, I mean, yeah, so I have been, yeah, it's always, it has been one of my favorite.
00:02:10
technologies to go to, so, yeah. My very first conference is actually at MongoDB in San Francisco. Awesome. Yeah, so I keep taking Kelly with me today. This time it's really... Now we've got to get you on the stage. We've got to get you one of these blue lanyards. Deliver a talk about... Awesome. Well, great. Thanks for taking the time to meet with me today. And Ramiro, from the account team, I'm a solutions architect. A little bit of background, yeah, I'm from software engineering, been in tech and software consultancy actually for 15 years,
00:02:44
so a lot of different industries, finance, life science, e-commerce, e-learning. Yeah, so looking forward to keep solving problems. Awesome. Yeah, that's what we do. Great. So if we could start with... Maybe just a brief overview of the application as it exists today, just describe the functionality. The functionality is, RTB is real-time bidding, that's what that stands for. It's a platform basically sitting between CCO and CCO, and then also the other side.
00:03:21
of RTB is the programmatic vendors. They are SSP, supply something provider, basically they interact with DSP, demand service provider. So that's the industry, how the business is being conducted through the real-time bidding.
00:03:53
process. We interact, our platform interacts with SSP, SSP interacts with DSP. DSP, DSP interact with the actual advertisers, Coca-Cola, Uber, and all that. So this is the very first programmatic platform that CCO came up with. Internally we have the ad servers. It's basically like a scheduler that says that, so CCO, there are two ways of conducting business for CCO. First they sell, they will sign the contract with the advertiser directly, and then through.
00:04:28
that ad server, they can schedule for the programmatic. So all these things I'm talking about is a programmatic advertisement, all right? So not the roadside, those are the fixed, that's a totally different story, but for the display, digital display, like the one in the airport, on the roadside. Through the ad server, they can schedule,

say, okay, for Coca-Cola, they signed a deal with us, so therefore we're going to put Coca-Cola's ad on this day, during this timeframe, because of how the campaign is set up. But then they're between those two.
00:05:00
there are little gaps, right? So therefore, the ad server will send a request to RTB, say, hey, RTB, give me an ad, go on and see if anybody want to bid on it. So the RTB will reach out to the advertiser and say, hey, in the next 15 minutes we have a 8-second slot, anybody want to put an ad over here? So those five vendors that we interact with will send us a response. And then at that point, once we receive those five bids, we will say who paid us the most.
00:05:35
And then we chose the winner, and we get back to ad server and say, hey, ad server, Uber just won this ad, they're willing to pay us $2 for 8 seconds. So real-time. It's real-time. So all these requests are synchronous. So within a few seconds, it has to be. Okay. Okay, so the app exists today, and it's issuing requests? Receipts. Receiving requests from advertisers. No, from ad servers. From ad servers. Internal CCO scheduling system, right?
00:06:09
So that's from one side. There's an inbound track request to CCO. Okay. Sorry, inbound request to RTV. RTV will send out that one request to five vendors. And five vendors will give back. So as a customer of CCO, I have an interface that enables me to see when real-time bids are available. When an ad slot is available. No. I'm just trying to understand.
00:06:39
What does the customer see? Our customer, we have two sides. One is our internal system, the ad server, the scheduling system. The other side is external partners. Yes. They only see the request that we send to them. Okay. The request is depending on those vendors.
00:07:30
available devices us okay so you have a collection with do you call them what's the nomenclature do you call those devices or displays displays okay so you have a collection with displays and there's probably some some details about the capabilities of the display right yeah that's resolution resolution okay so that's that's the size and the number of pixels and location yeah and does it also include latitude and longitude and yes okay yes so that so that's, That collection exists, and the RTP server references that collection to make available.
00:08:07
time slots that are not consumed. RTP doesn't really care about that. RTP really just receives a request from ad server. But how does it know what displays are available at what time? Ad server will let us know. Ad server. Ad server will let us know. RTP for display X, there is a 8 second slot. Anybody wants to play. Wow. Find a player for us. So this is tight scheduling. Yeah. So time is money. It is. Yeah. We suppose our contractual obligation is we receive a bid from vendor, and within 15 minutes,
00:08:40
we need to finish, fulfill that request. Otherwise, we don't. So that's the SLA. 15 minutes. Yeah. 15 to 30. Yeah. It's depending on vendors, but the shortest is 15 and up to 30 minutes. All right. So a slot becomes available. It becomes, it is made available by the system to somebody who can consume it. Yep. And they have the choice to bid on it. Mm-hmm. When they bid on it. It's.
00:09:39
hard to. really. The metadata associated with the displays, how big are those documents? What kind of metadata are you storing with the displays? When the request goes out, how big of a piece of data is that?
00:10:12
Very small. Okay. The request to vendor is small. Hundreds of bytes. Yeah. Okay. That's good. So 5,000 per? Minute. Per minute. Okay. But that's inbound requests from S servers, but then we send out those requests to vendors. So we have five vendors. So the one request comes in from S server, five vendor requests out. Okay. Then we see, in a good day, we'll get five bids, and then we'll do the calculation. But a lot of time, we only get one or two bid. How do the vendors know if it's a good deal or how much to bid?

00:10:47
That's just industry knowledge? When we send a request to vendors, we have to let them know the impression. So impression is kind of the way to measure how... Where is the impression stored? It's coming from a third party called GeoPath. Okay, so you send an API call at the same time you get the display? GeoPath is an industrial organization, I think. They update the impression on a weekly, monthly basis. So we do that through traffic studies and... Right. Yeah, okay. So we have a background service separately.
00:11:19
Okay. They synchronize the impression data from GeoPath into our measure API. Okay, awesome. Tell me about the tech stack. So what are you using for your technology today? We build various microservices. So measure API is one of them. And tech stack, everything is written in Go, most of them. And running on Kubernetes. So everything is on GCP. Oh, good. Okay, great. Good partners of ours? Oh, okay. So you're hosting the MongoDB component of your stack in Atlas?
00:11:53
Yes, Atlas. And you're leveraging GCP for the hyperscale? Great. Central. Okay. So I feel like I... I've grokked the RTV landscape a little bit, learning a little bit more about advertising. I worked for a company called Griffin-Bacall back in the day, and we did something very similar. I wrote a media planning system in FoxPro. Oh my goodness. Really? I haven't heard that term for ages. Yeah. FoxPro. Yeah. It was Dbase, and I converted it to FoxPro. Anyway, enough about that. Oh, that's so cool. So I have a little bit of knowledge. I feel like I just met an old friend.
00:12:24
Yeah. Right. What did they call this? Third generation languages. Oh, that's so cool. Yeah, yeah. So I have a little bit of knowledge of the industry. We have the tech stack. We have some of the volume, the data volume. What's the total amount of disk storage for the RTV application? Probably gigabytes. Gigabytes, yes. Some terabytes. I'm just thinking about where, what... Because we actually... I don't know. I know it's gigabyte.
00:12:57
I think... Yeah. Like, it could be 100 gigabyte. Like, when we start saying, like, oh, it's in the two terabyte range, that's when we start to barrier our discussion around looking at sharding. I had one of those on top of my head, and I forgot about it. Yeah, it's okay. Oh, I love the sticker, RTB. That looks like an AI-generated. Yeah, it actually was. We had just rolled out to the list. There's a lot of tells. We went to RTB 2.0. That's when I kind of came.
00:13:27
They had RTB 1.0. It's when I came on during the revamp to 2.0. So yeah, the reason I'm struggling with the size is because right now MongoDB Atlas is hosting the RTB database. We only keep up to two weeks. Oh, and then what happens? We online archive the data. Good for you. That's a great use of the technology. Yeah, and then we kept there for another 30 days.
00:13:59
Okay. Because the RTV system, so how, right now we have a, our Power BI is getting the data out of Redshift. Okay. And that Power BI suite has been developed for years, and our stakeholder didn't really want to give up that Power BI interface. So therefore we, so we wrote RTV, the current version of it, this is a rewrite from the.
00:14:29
previous version, and we launched the RTV v2 last year, a year and a half. So that's why the reporting section of it is never part of the RTV v2, that will be the reporting the way it is, so we actually build, basically we push the data to two places, one is Atlas, the other place is we'll send it to Firehose, and then continue to Redshift.
00:15:20
And they have to actually normalize that into a relational database environment. Why are you not leveraging SQL in the MongoDB Atlas platform? You know you can send SQL to Atlas. No. Are you talking about the federation? Nope. No? Okay. No. Built into the Atlas platform. If you're hosting in Atlas, not if you're hosting on-prem, this is one of the benefits of hosting in Atlas. You can send SQL, standard SQL.
00:15:52

Now it does depend on the structure of your database, and it may not be as performant as a native SQL. of SQL because there is, I mean, it's physics, but it's a great way to leverage the tools in place without having to do an ETL out and in, or in and out. That's good to know. I think right now, there's another reason that I kind of forced them to separate from us because the Power BI has such a great optic visibility on the stakeholder side, if there's.
00:16:24
any issues. Any issue, we will be always on the constant pressure from the stakeholder, so therefore, I want the... See, not my monkey, not my service. No. So I want the team to, engineering team, to focus on engineering rather than reporting, so therefore, we basically, we just duplicate the data and send it all over. I think that might be like a V2 of this discussion. That might be some where we can kind of squeeze out some efficiencies, but understand it completely. So we'll focus on the issues.
00:16:57
So I feel like I understand the architecture, I understand the application work. workflow. Sorry. Aaron. Mike. Nice to meet you, Mike. Good to meet you as well. You're on the Clear Channel side? Yes, sir. Okay. Oh, you're CSM. Yeah, yeah. I'm on the Clear Channel. Very good. It's awesome to have the CSM in the room, too. So, I feel like I understand the architecture of the application. Talk to me about issues and concerns that you have about the application. The issue is, I don't think we have an issue per se, but really just I want to hear some.
00:17:27
professional advice about how we can make better with our system. Right now, we are pushing data to MongoDB, but we're not doing much about it. To me, it feels like a kind of waste, and I hate that. When you say it feels like a waste, how so? Because, well, first of all, the business side is not relying on MongoDB for reporting. So, you're building this whole separate Tableau BI power?
00:17:57
So, what's the point to save the data in MongoDB? Right? Yep. So I want to make some use case out of it. Yeah. I feel like right now, the immediate use case will be, hey, help the engineering team understand how the system works, right? If you want to do troubleshooting. One of the common question is, our product manager will come and say, hey, Stanford, what's going on with this particular request? Mm-hm. Because every request has an identifier. So what's the, what happened with this request? How come we're not receiving any bit from this particular vendor? Yeah. Or how, what happened with that particular display?
00:18:30
How come this display never got any bit on, right? Or how come this bit, this display only got bit by one vendor and not others, right? So this kind of information can be easily answered through the data we have in the Mongo, on the MongoDB site. So I want to be able to say, hey, can we create something to, you better utilize all this information. Correlation, like, yeah. So I, I think we can definitely help for sure. And there are several ways to, to approach this. The first is. I'd love to see you relying directly on MongoDB Atlas' SQL capabilities.
00:19:07
I'd love to cut out that ETL. There's a lot of fragility associated with creating an export, and then syncing up, making sure, and depending on the amount of data that can be very costly. So I'd love to talk to you about leveraging MongoDB's SQL capabilities. Second thing is, if you have the data in MongoDB, we have, you know, obviously it's available for you to report against, and there are several ways you can do that. Writing your own application is one,
00:19:38
and we can talk about the data model for reporting and make that more performant from a reporting perspective. But there's also infrastructure elements that we should talk about. MongoDB charts? Heard about it. Okay. We're talking, we're having a chart session at 4.30. Oh, good. Great. Yeah, that could be a way to get instant wins. Yeah. Charts is just so phenomenal and it's come so far. There's even like an AI prompt built in where you can, in order to create your charts, just.
00:20:10

leverage natural language and it'll look at your data model and produce charts that kind of make sense, give you suggestions for what to display. So that's one thing. I'd love to see you dive into charts because I think you'll get some value right away. How about, so I mean that's one issue and we'll talk about that and that sounds like it's just increasing the investment, not through additional spend, but by leveraging the value that you have in the data.
00:20:40
That's one. Two, what about performance and like stability of the application? Are there any issues there? So far, knock on wood, I think we're fine. Great. Well, that's a testament to good engineering. Well, I have to say, what we experienced after the launch of the app, I think it's a testament, After launch, we actually experienced one outage, it is actually because of MongoDB. Oh, really? The TTL was not, somehow the TTL went away, so therefore the database keep growing and.
00:21:14
growing and growing, and then now we want to try to reapply the TTL onto a production database and then the indexes was not applied properly, so I have to reindex the whole database and then reapply TTL, boom, that caused the outage. Did you ever discover how the index was dropped? Like if the TTL index was dropped, I mean, that's going to be the ultimate cause of that. I'm not trying to like, you know, push blame on anyone, but like understanding how that index got dropped, that would be.
00:21:44
I have no idea, because I actually manually create the TTL before we launched, when the database was empty, and it was up and running for a while, and then there's one, and then, a few weeks later, I got a call from our... So, what I might suggest is that we look at roles and as a part of your standard operating.
00:22:17
practice, don't issue MongoDB Atlas admin to your developers, right, so you can, there's a fairly granular roles-based access control model that will enable you to have the developers have access to the data but not be able to remove indexes. Yeah. And it could be, I've done it, so I was responsible for trade plan at Merrill Lynch Wealth Management and I hired a consultant and I had written this amazing system that, I wrote an agent.
00:22:47
that ran on every server in Merrill Lynch. Right. And it fed back to a JavaScript front-end, React front-end and it reported on the health of the environment and the consultants working on it were adding it to their system. And I'm like, I'm going to push to QA for testing, and I'm like, okay, let her rip, let her rip. All of a sudden I hear, like, bing, and I look at my screen, it's all red, like, there's all these red things. He pushed to production. Oh, my gosh. So I, not single-handedly, but double-handedly took down trading at Global Wealth for, like,
00:23:20
my lead was like three minutes, cost the company millions of dollars. Anyway, so I understand the complexity. But, yeah, I'm not complaining, I'm just saying that it's good exercise that really gave us a chance to kind of review how things should be done. So I, so far, since then, there's no major issue whatsoever. Okay. All right. So one, leveraging your investment through implementation, we'll talk about that. And I'll prepare some forwarding issues around content around that.
00:23:51
Two, some roles-based, SLAs and performance, there was only one associated with possibly, and alerting, we can look at alerting to make sure that... If an index drops for some reason, you're alerted. The third part of that is in, flew out of my head, alerting, oh yeah, so in the alerting space also looking at the length of time that a process is taking, that might be something.
00:24:21
that you want to plug into. Are you leveraging an APM platform today, or like a monitoring, what are you doing for monitoring? Monitoring, I mean in overall, or just MongoDB? Well, I'm clearly interested in MongoDB, but I'm wondering if you have like a monitoring platform in place, like PagerDuty, or like Datadog, or something like

that? No, we started with PagerDuty, but then we stopped it. I think our infrastructure team right now is really relying on, they build this, what.
00:25:00
based on the error logs, like a health query, and then they, yeah, we don't, we're not using any third-party tool for alerting. It's just basically in-house, leverage what GCP provides and then just send out a slack. So if you have a page dedicated to health of the system, I'm wondering if you could add to it a check on the oldest document. You see what I'm saying? Like if you, yeah, so add a chart showing the age of documents over time and then alert based on.
00:25:38
I have to say the online archive works pretty well. Doesn't it? Yeah, it works better. So that's another question. What's the difference? I guess online archive is really for, it's Atlas service, right? Yeah, yeah. So for people who are not using Atlas, they're relying on TTL. Correct. But if I'm an Atlas user, so which one should I do? Should I go with Atlas? Yeah. TTL or online archives? Well, I mean, I would say it depends. Almost every question that I get asked in a design review session is the answer is it.
00:26:10
depends. Okay. Whatever your comfort level is. Okay. But if you're, I mean, you're going to get more value out of leveraging the services in Atlas. So I'd say look at it. Okay. Look at it. I mean, because you're going to manage the TTL indexes on your own. There's risk associated with that. Right. But with a service like Atlas, we maintain a 99.999 SLA for those services. So leveraging online archive instead of writing your own and managing your own TTLs is probably going to give you a better experience. I think the way I'm doing is writing, well, okay.
00:26:43
So if I apply a TTL, once I online archive the document, is it going to go, it shouldn't go away? No. You can tell it where it goes. Okay. Yeah. You can specify that it goes to a cooler storage or... I think it's a mixed approach. We online archive first and then... But I still have a TTL index. Do you ever have to restore from the TTLed collection? Not yet. Okay. But I have my connection string already, so in case I have to connect to an online archives.
00:27:14
cluster, I can query out of it. Yeah. Awesome. Not yet. Cool. So, any other issues or concerns that you have? I do have a question about what we'd like to see, if I can get some advice from you, is that the nature of how RTV works, the system itself, it's all, we heavily rely on PubSub, so it's a hub. Because of how the data is being passed around, we build several microservices, so in order.
00:27:45
to link these microservices together, we use PubSub, and then because of the nature of those microservices, our data now, you know, the one add request, generate five requests from vendor, and get... process and then we go through this auction process and then notification process. We generate a lot of, our whole flow, even though it's one synchronous flow, we generate multiple documents, I mean, several collections. So now it makes the query very challenging because let's say my manager say, show, tell.
00:28:20
me everything about this S request ID. I have to query multiple collections in order to... Why multiple collections? Because each service, because they are distributed, they are not connected, they are all synchronous. But can the services, the individual microservices, leverage a common collection? Like if microservice one is dealing with ad service and adding an array element to a field.
00:28:57
saying I touch this and here's what's going on. the status is, then it passes it off to another. Why can't that other microservice also reference the centralized database for kind of this like streaming state management? The way it is designed right now is that each microservice does not directly write to MongoDB. We have another service basically as a subscriber, a consumer that subscribes all the events that are being emitted by those microservices. So that sync service is the only one that writes out to MongoDB.

00:29:30
So I think here's some meat, I think, for optimization. I would love to see like a data model, like what does the collection footprint look like? I do have that. I've collected a set of the collection, I mean, documents. Okay. Yeah, I'd love to take a look at it. Right now? Yeah, sure. Okay, cool. So, I don't know if there's, oh, great, yeah, if you could send those to me. Okay, sure. Yeah. You can just go through it now though.
00:30:01
Oh yeah, so I have an ad request. So we basically... We break out one... What am I looking at? Why don't we ditch this? So there's an ad request. That's the first collection. The second collection is called auction results. And you're referencing back to the... So this is a reference model? Yeah, so this is the very first. The initial ad request coming from ad servers.
00:30:33
Why a separate ad request ID? So you've got an object ID, which is unique. And indexed for free. Then you're creating an ad request ID. Correct. That's another habit of ours. Almost like our... I know it doesn't make sense. Well, I'm not judging at all. There may be a valid reason for it. Back in the old days...
00:31:14
So there is another field called request ID. Go has methods that will enable you to transfer an object ID to a string and back again. Yeah, he started playing with us. Yeah, so actually in our... At the end of the day, so the way you've implemented it here, it's not bad. It's not good or bad. It will cause you to have another index on that. That's why I recently learned that, yeah, why didn't we just utilize the underbar ID?
00:31:44
Because it's already indexed. So every collection always has a request ID. Okay, good. And they can use the add request ID. We can just kind of join them all together in order to have a bigger picture. Joins? MongoDB doesn't do joins, does it? No, no, no. So, as you can see, in order to tell a story about a particular ad request ID, I have to go through all these collections. So ad request, starting with ad request, and then we have the transactions collection handles.
00:32:17
all the... Give me the reference to the ad request ID. I see it. Yeah, there's the ad request ID, transaction. This is for the transaction information with a particular vendor. Okay. So if you have five vendors, so that you're going to have five transactions per ad request ID. Okay. So that ad request will be referenced by all vendors and will be stored in there. Right. Perfect. And then there is auction. So basically, if we receive the bid... The other side of this. Yes. Yeah. Okay. Once we receive the bid from vendor, then auction, and then the auction result.
00:32:49
So who is the winner? Who won the bid? Okay. Are there any elements of the decision criteria for who wins the auction? So sometimes not the highest bid is not always going to be the winner. Where does that rule live? Auction service. So that is a part of the RTP. So that is an API call? It's an internal process. Yeah. It's not related. Yeah. Okay. All right. And the data for that, where does that live?
00:33:22
Like the service associated with evaluating the bids as they come in and spitting back a, here's the winner. So bid service, we can send out the bid to vendor, grab the bid from vendor, and then publish the message to collect the bid. Publish that to PubSub. A subscriber, which is auction service, subscribes to the topic. Through a subscription auction service, we see the bid information. So that's how all each service exchange data through PubSub. So I'm curious as to whether or not we can pull that into the MongoDB estate, that service,
00:33:56
what it's doing. Because it sounds like, I mean, I don't know what that looks like. So I don't know what the decision criteria elements of data are, but if I were architecting this I would try and consolidate as much as possible. And the rules are probably looking at elements of the data that you have in MongoDB. If that's the case, I'd say let's talk to those folks about maybe leveraging the data directly from MongoDB rather than

PubSub. But we do have a very, okay, the performance is an issue here, well it's a concern here. Because the whole chain, we receive a request from a server, reaching out to a vendor, coming.
00:34:31
back, finding a winner. This has to be done within maybe one and a half second. Wow. Well, I will put MongoDB side-by-side against any database provider in terms of scale and speed. So I'm not concerned about that. Okay. Like, sub-second response time is like the norm. We have customers that are leveraging for high-frequency trading and like thousands, hundreds of thousands of transactions. So I have no question about the capability of meeting your SLAs with that.
00:35:05
What I don't know is the elements of data that go into making that decision. So if there's additional fetches that we have to do to other data source elements, that's going to be, like you look at the front to back, you know, for the total SLAs. There is, because we have some other, the display API that holds the information of the display. Network API holds the information. So there are other microservices that RTB needs to connect. Okay. So we'll leave that for the side. The one thing that I took a note that I wanted to talk about a little bit was the PubSub. What technology are you using for your streams or your publication?
00:35:40
Like is it like RabbitMQ or like what? Oh, it's GCP's native PubSub service. So you're adding things to a queue and tagging them. You can say that, yeah. Okay. And then the subscriber side of it is looking for specific tags and pulling that down. So built into the MongoDB platform, we can do that. We recently announced Atlas Streams, that could be something that could eliminate some complexity associated with your application. So having it all be in the MongoDB family of services could be interesting.
00:36:10
Would that be kind of breaking the pattern that we kind of, all the services now reference to one single database? Yeah. It would. It would, right. Okay. Which would make us happy. Yeah. But, I mean, so the services within the MongoDB suite are all independent services with their own SLAs. I see. So, I mean, I'm just thinking in terms of, like, reducing complexity in the applications. But if you're happy with GCP. I'm not here to say, let's tear apart your application.
00:36:43
I'm just here to look for opportunities. I'm not sticking to GCP for the sake of GCP, right? There's things to make it better and more reliable. Might be interesting to kind of do, like, a proof of concept and see if you're able to eliminate hops between the MongoDB estate and the GCP. So, we're going to talk about MongoDB Atlas and GCP, if that could be more performant for you. When you say MongoDB state... MongoDB Atlas, the suite of services. So, MongoDB is not just a database. Oh, right. You know, you have time series indexes. Right.
00:37:13
You know, you're leveraging some of them. We even have, like, a data federation capability. And there's, like, I don't know, 15, 16 services in the Atlas platform. Right. And one of those is now streams. Okay. So, maybe that's something we can double click on over time. Any other concerns? Or, like, what would you hope to get out of this discussion with me today to make you feel like it was worth your time? Really, just talking to professionals. Yeah.
00:37:44
So, the most impact that I've been able to have with customers is when I dig into those, the data model. Okay. And it sounds like the individual collections that you have are relatively distinct. And I don't see any overlap or reason to be concerned about the data.
00:38:30
removing the time it takes to do that calculation by doing it at write time. And if your application is write-heavy, you can pre-compute, store that computed value, and then read it. Sorry, if your application is read-heavy and you're very concerned about the SLAs for reads, you're going to want to move the compute, move the

calculation to the write time. So pre-compute a list of array elements and store a value in another field. That way, you're not doing the compute at read time, you're doing the compute.
00:39:00
right time and read is ultra-fast. Second thing is for your data analysis, the BI side. So that's still not in MongoDB. I was going to say we can dramatically impact the cost exposure that you experience by implementing an infrastructure that makes use of analytics nodes. So these are nodes that are specifically designed for serving analytics data. You have that?
00:39:30
We have that. Oh good. Okay. Great. Terrific. Yeah. So in order for me to give you any more advice, I'm going to need to dig in and spend some time with those data models. So if you can send me those. The other side of that is the query side. So, leveraging and looking at the data model is really irrelevant unless I look at the queries that you're issuing against the database. A lot of times what people do, especially if they don't don't have the level of experience that you have with MongoDB, they will...
00:40:00
they will leverage a pattern where they're pulling data and working on the data in code and then pulling more data and working on the data in code and that can be effective like a lot of like Node.js developers do this because it's it's cumbersome for them to actually wrap their mind around the aggregation framework but leveraging the ag framework can eliminate many passes back and forth between the database and the application because you're you're formulating the business logic in a query the query is sending it to the to the database and the database is ultra fast when it has to do things like joins.
00:40:32
between like I'll give you an example if you have data in multiple collections which you do you know that you can leverage the dollar lookup to join data a lot of people they issue a query to one collection pull that data back get the ID issue another query to another collection pull that back all of that is is overhead that that can be eliminated by by formulating an aggregation query rather than multiple finds what is the most commonly mistake making oh absolutely without a doubt.
00:41:02
So, I'm doing a talk this afternoon, 1.15, and I'll cover this, but by far the biggest mistake that people coming to MongoDB from other technologies, the biggest mistake that they make is just converting tables to collections. Just saying, I have this relational database, and it has, I don't know, 15, 20 tables, because it has to, you have to divide the data, because the relational rules, just converting each one of those tables to a collection, and then writing an application to pull from each collection.
00:41:36
That's by far the biggest mistake, because you're not getting the value of the really rich structures that you can create in MongoDB. Embedding versus referencing. So, we talk about the value. The default preference for MongoDB data modeling is always going to be embed. Why is that? Why do you think that is? Because you have less... Yeah, less... Less...
00:42:08
I'm going to show you how to create a huge document. That's precisely it. So it's a balance between that. So how do you make that determination? Well, we look at the workload and how the application references the data to make that decision between whether we're going to reference it or not. One thought I have is that looking at our multiple collection approach, right, does it make sense to create one collection that just combines all these documents all together? Because I know all these things can be grouped by the request ID. So does that even make sense to create a collection that has everything?
00:42:38
That is, like, you're spot on with that thinking. And if you take that to the furthest degree, like, you'd have all of your data. And, by the way, one of our field CTOs is a huge proponent of saying just create one collection, and you can have polymorphic data structures so that each document, you know, Rick, right? Yeah, yeah, I was

just thinking about Rick. Rick has this, like, Right. He overloads the object ID with multiple components of data which represent values and which would.
00:43:09
typically be fields. So he overloads the object ID and you can have a single collection, compound key. And you can support that with MongoDB phenomenally well. We actually used create one application with that. And it works so well. It works so well. The challenge becomes when you have a lot of data and your document sizes approach the 100 megabyte, you don't want to have a 100 megabyte data element. So it's a balancing act and there's almost never one right answer you should embed or.
00:43:43
you should reference. It's always based on the workload of the application and there's like a four step process that we go through and that's in the data modeling. So what I'm going to do as a part of preparing this report, I'm going to include information on the data modeling exercise. And I would highly recommend. We're going to be launching a new... I guess service that, like I typically travel around the country doing these things face to face and delivering talks. We're doing virtual developer days. So now it is an investment in time. It's like we're going to try and make it about four hours max, but you're going to get the.
00:44:15
entire day, developer day, of the intro lab, the data modeling lab, the aggregation framework, the search lab, and even RAG, retrieval augmented generation. So the first one of those is going to be, I think, on September 19th. I'll make a note to remind you that that's coming, and if you can attend, that'd be awesome. So you get the value out of that. Do I have time for one more question? Absolutely. In the transaction collection, I have a field called request and response.
00:44:47
Those are the requests that we send to our vendor, responses that we got from the vendor. Because those schema were defined by the vendor, so therefore we cannot have a specific request. We'll have to leave it as an interface using the Golang's term, meaning that it can be anything. It's just a free form, free form JSON. What is the best way? But sometimes we like to, we need to query against those data sub-documents. Yeah, sometimes you can't control what the data model looks like.
00:45:18
Correct. The good thing about this is like, so is the data, what does the data look like? It's the JSON. We will process those. And the values, if you had to say the majority of the values are string. String. So you can leverage a search, you can leverage a MongoDB search with a wildcard index which will index all the text fields and it's going to speed up access like phenomenally. Even though that request field is a document, it doesn't need to be like a description and then all. No, no.
00:45:48
No, it can, yeah. I mean, so it's the Atlas search service is based on Lucene. Are you familiar with Lucene? No. So if you want to be able to search for that. for things and you have a general idea of what you're searching for. Atlas search is the way to go. And what that's going to do is it's going to eliminate the need for you to separately export your data to a search database, a search focused database like Elasticsearch is popular.
00:46:19
We had many of our customers were doing these implementations whether it's spin up Elasticsearch with Lucene and then doing this ETL between and indexing that data in Elasticsearch separately. And it works well, it worked really well with Elastic but we implemented Atlas search so that you could eliminate the process of establishing all of that infrastructure by hand and spinning it up and maintaining it and all that stuff. So Atlas search will, all you need to do, and it works so phenomenally well, all you.
00:46:50
need to do is establish a text search index and it behind the scenes in your Atlas infrastructure spins up a complete separate infrastructure with Lucene with all of your data. All of the components that are required to index your data for fuzzy searches and you just basically, it works the same way, it works with the MongoDB query API and the aggregation framework and you just leverage your text search index. So what I'm

suggesting is that for that collection that you don't know, I would say establish a wildcard index on all text fields and you're going to get some good performance there.
00:47:23
If you want to take a look at this, there's a really great way to do it without like messing with your existing Atlas infrastructure, you can do it on your laptop. You can, using the Atlas API, it's the Atlas API or Admin API, I think it's the Atlas API. So you can simply, on your local device, use Atlas, the command line interface to Atlas. And you deploy an instance on your local laptop, it's going to spin up a little MongoDB instance,
00:47:56
you establish a text search index. I really want to try that, but I really hate to do that in the production area and get.
00:48:29
yelled at by our EE teams that, why are you spending extra money for this or that. Well, the great thing about it is there's no additional cost for the infrastructure associated with that. You will get charged for the ingress and for the additional compute associated with that, but that's going to be fairly minimal. And you get so much more value out of it. Plus, if your application has any, and it should, search, you're going to, like, wrapping search into your application just makes it so much more usable.
00:49:14
The other thing that you were saying before on the query side, if you know you're querying by certain fields, you can just add schema validation on that. Even in search, search will allow you to have those ad hoc queries. But then schema validation would allow you to say, my application relies on those three fields that are from somewhere else. Great point. Schema validation. The other thing is schema versioning.
00:49:44
So as your vendor changes the data model on you, you may want to have maybe somewhere in your pub sub or something. Like introspect, look at that data model as it's coming back and look for changes. Increment a version number so that your code now understands.
00:50:31
How long we've been at the company. We've been at the company, I've been like 98.9% more than, he's like 99.99%. Anyway, and I say that not for like, you know, just giggles, but it's because he's really good. He knows, he's written an article on schema versioning that I'll include in this report. I'll include a link to that. It's like how you can really intelligently implement schema versioning. Yeah, because we did recently went through some on the vendor side, they added new fields. So we have to, you know, at least let us know, then we'll have to change it, right?
00:51:05
But... Yeah, that's always a challenge. Yeah. A schema version is especially useful when you have, like, two at the same time, where you can say this one request has this particular thing, and then this other one has a different. So we can detect that programmatically, then we can handle it. Yeah. And it's part of one of the, what is called, design patterns, so it's pretty much in there. So it's schema versioning. And just not to say something else, but on the schema version, you can use the polymorphic.
00:51:40
for those different vendors. Because like 80% of the things are common, but that 20% is like, oh, this vendor has this thing, this vendor has this thing, you're using polymorphic. He's talking about polymorphic, different shapes in your data, but also inheritance. So you can have a group of things. old standard for vendor collections, which has to have this, but can also have this. And Go is really good with inheritance and inheritance model. Yeah. Cool.
00:52:11
Okay. So I'll include that in there. All right. Any final questions or, like, concerns that you have? Anything. So I'm going to ask you to send me those, the data model and the query patterns, like the queries that you're issuing against those. Okay. That would be helpful for me, if I could do that. Well, whatever you can share, that's fine. Yeah. Everything will remain with me. I'm not going to be sharing this with anybody but the account

team. Yeah. I have no problem sharing that. I'm just thinking about the query pattern. The query pattern is all in my head.
00:52:42
Oh. Right now, we don't have a system that is really doing all the queries. That's one thing I really want to accomplish, is that eventually come out some kind of application allowing our product manager to answer those questions, to find out the answer. Okay. So I'm going to make a suggestion. Yeah. I'm going to bet. I'd be willing to bet. that there's so much value in the data that you're collecting, even if it's transient and TTL, that you could implement a really powerful... So we talk about Atlas Search, and that's really valuable when you kind of know what you're searching for, right?
00:53:13
You may know the field, and you want to index that field with text search and have it like a fuzzy search. That's really good. But what if you don't know what you're looking for? Someone comes to you and says, I have this request that's out, it's with this vendor ID or whatever, and I don't know where the status is. Leveraging MongoDB Vector Search, are you doing anything with artificial intelligence today? We would love to. My boss would be really super happy if we are doing something. So just imagine you don't know what you're looking for, but you know you can describe it.
00:53:43
That's where Vector Search comes in. MongoDB is a vector database. It has been for the longest time, like longer than any other vendor. We implemented Vector Search back in, I'm going to say like 2011, with the advent of GeoJSON. GeoJSON is essentially... essentially the same thing as a vector. Are you familiar with artificial intelligence applications? I'll explain it really quickly. Computers don't know strings, they know numbers. In order for you to be able to interact and converse with a computer, there needs to be.
00:54:17
this interface between words and numbers. The large language models are created specifically for that. A large language model like GPT-35, 3-5 Turbo, has this vision of the universe. It's got billions of parameters which describe the universe. When you send a string of text to the large language model, it's going to find it in its understanding of the universe and it's going to return a 4,096 array element of its understanding of what this is. You say dog, it's going to say, well, a dog has fur, it's domesticated. It's like 4,000 elements.
00:54:51
in this array. Imagine thinking in 4,900, 4,096 dimensions. We can't do it. We can't wrap your mind around it. The LLM is essentially this star structure, and at the center is zero knowledge. As you progress outward, there are spindles from the center. Each spindle is a domain of, like, it's domestication, it has fur, it lives in North America. Each one, there's 4,000 of these, and each one of those 4,000 spindles can also have.
00:55:22
spindles off of it, and when you send a word to it, it's going to find it in that spindle and send you the description, which is this 4,096 array element, or however many parameters. It can be, like, 1,500. So this becomes powerful when you want to be able to search through the data that you do have. If you've got a text element that describes what this thing is, you index that with MongoDB's vector database. So you essentially send that data to a large language model and get that encoding, create.
00:55:55
a new field in your database called encoding, and you index that with vector search. Now, going forward, anytime you want to search for something, you encode that using the same large language model. You encode the question, where is this thing that's with vendor XYZ, and it searches and compares. It does a cosine similarity comparison, just like GeoJSON. Just like if I had said, I'm at latitude this, longitude this, altitude this. Find me the nearest point to that in this.
00:56:26
database of latitude longitudes. So it does a cosine similarity, very similar to the Haversine formula. So if you have two latitude longitude pairs, I can do this Haversine algorithm, which is nothing more than a cosine similarity check, and find out the distance between those two points. That's exactly what happens with

MongoDB's vector search. The vector search has an encoding, an array, in each document. You send your query, dog, to the large language model and get another array. You line those up together and you say, what's the comparison along these spindles? How similar are they? And you get a score.
00:57:04
like it can describe for you how close they are, so in a list of documents in your database you can find the closest matching element and say, I don't know what you mean, but this is what I'm looking for, I think this is what you're looking for. But it can also do things like auto-complete, depending on the large language model you use, like what would be the next word in the sentence, I mean the possibilities are limitless. If you want to see this in action, I've written an application called the AI Lab Assistant, if you take a lab with me, you'll get it, I'll send you a link to it, and you can ask.
00:57:38
it questions, and what it's doing is, it's taking your question, sending it to a large language model, comparing it to the list of questions that I have encoded, and it's doing it. And you can see that in the code, how it actually works. But really, a powerful way to add usability to your applications. I just feel like we have so much data here in the database. Yeah, you can just add it to the database. so much value. We can somehow, yeah, have a little tool to answer some questions. I think that would be a good start. Oh, that would be an amazing POC. Yeah, and we're.
00:58:08
actually doing that at 3 p.m. today on the Gen-AI session. Yeah, so Apoorva, my colleague, is going to be delivering the RAG session. Yeah. She's phenomenal, she's so good. And I'm gonna be there on that workshop helping. I will too. I think I have American Airlines at that time, but if not, I'll be there too. Awesome. So I'll include links to some of the things I talked about and hopefully we'll get you some some joy.
00:58:39
Any other questions before we wrap? Thanks so much for taking the time. Thank you. I hope we can add value. Thank you so much. Thank you very much.