EPISODE 1503

[INTRODUCTION]

[00:00:01] ANNOUNCER: This episode is hosted by Alex DeBrie. Alex is the author of *The DynamoDB Book,* the comprehensive guide to data modeling with DynamoDB. As well as *The Dynamo DB Guide*, a free guided introduction to DynamoDB. He runs a consulting company where he assists clients with DynamoDB data modeling, serverless architectures, and general AWS usage. You can find more of his work at alexdebrie.com.

Data analytics technology and tools have come a long way in the last decade, but it can still take weeks to prototype, build and deploy new transformations and deployments, usually requiring significant engineering resources. Plus, most data isn't real time. Most of it is still batch-processed.

Tinybird Analytics provides you with an easy way to ingest and query large amounts of data in real time, as well as to automatically create an API to consume those queries. This makes it extremely easy to build fast and scalable applications that query your data. No backend needed.

In this episode, we interview Jorge Sancha, Founder and CEO at Tinybird.

[INTERVIEW]

[00:01:17] AD: Jorge Sancha, welcome to Software Engineering Daily.

[00:01:19] JS: Hey, Alex, thanks for having me. It's great to be here.

[SPONSOR MESSAGE]

[00:01:31] ANNOUNCER: Rookout it is a developer-first observability platform that provides an unparalleled ability to collect any piece of data, including logs, traces and metrics, from the deepest levels of live code in their production environments, with the click of a button. Unlike traditional monitoring tools and APMs, which tend to focus on metrics that DevOps engineers and SRE's care about on the infrastructure level, Rookout is built from the ground-up for developers who care more about the actual code and business logic of their applications. Try it at rookout.com today.

[INTERVIEW CONTINUED]

[00:02:22] AD: Yeah. So, Jorge, you are the founder of Tinybird, which is a real-time analytics company. Tell us more. What does Tinybird do?

[00:02:29] JS: So, Tinybird is a platform and a set of tools that speed up and simplify development over huge amounts of data over data at any scale, really, just using the SQL query language. It helps developers ingest and transform data at any scale. And then it also enables them to build API's that they can expose and integrate into their products.

So, you can think of it as a data warehouse for developers, one that they can use as a backend to build and scale API's over any amount of data. It also works serverless. So, it's designed specifically for developers that might not be familiar with infrastructure, such that you don't have to worry about how many servers you need to provision, or what's the right configuration for each of them, etc. You just throw data at us and we take care of helping you scale that up.

And essentially, our goal is to enable any developer to build and scale data products and applications over any amount of data, whether that's big or small.

[00:03:29] AD: Awesome, I love it. And I want to get more into the architecture and all that stuff later on. But one question I have here is, like, what data size should I have before I start considering Tinybird?

[00:03:40] JS: I mean, the beauty of it is it doesn't really matter. This works really well anyway. It's more about what type of data more than the size of the data, and what you want to do with it. So, it's great for analytical data. It's when you want to figure out things based on data that you're generating constantly, more so than to build a transactional application, like an e-commerce or something like that. It is to analyze data and do things based on that, whether that is informational, or automation, or personalization. There's a bunch of use cases for that.

But one of our goals is that it's always fast. So, it doesn't matter if you're starting with a few hundred records per minute or something like that, or thousands of records per second. This is designed to build up.

And our key value prop is not just the scale. It's the speed at which you can develop at any scale. So, how quickly you can get to market with your solution? So, that's the core of our focus, is that combination.

[00:04:43] AD: Awesome. And so, you're talking about – You mentioned, like, speed of analytics queries as well. Can you give me a sense of like, hey, some of your large-scale users? Like, how many rows are we talking and like what's the query times for doing analytics over those rows?

[00:04:56] JS: Yeah. So, we have customers that are ingesting 250,000 requests per second, for instance. And based on that, they are building a bunch of different use cases, from a web application firewall that it's API's that are analyzing the data that is coming in constantly to detect denial of service attacks and mitigate them automatically. To using that data also to build usage-based billing. Or also to show it to your customers as user-facing or in-product analytics. So, those are the sizes that we can manage and more. That's just one customer.

So, obviously, the aggregate of that is much more. And then we have, on the other side, on the query side, up to thousands of requests per second as well in terms of API's that are exposed. And those have to be very low-latency such that it can scale. Because otherwise it becomes prohibitive. So, we see latency – Depending on the use case. But we see latencies of 30 milliseconds, or 60 milliseconds and sometimes 200 milliseconds, but it really depends on your use case and what you're trying to optimize for.

[00:06:09] AD: Absolutely. And we hear this. I think in the last couple years, we've heard more and more people talking about real-time, real-time data, things like that. Like, what does this mean real-time? Like, how real-time is this especially for large aggregation type things? How behind is my data there?

[00:06:23] JS: This is really interesting. And this is something that we've been finding a lot in conversation since we started. A lot of people hear real-time and immediately disconnect from the conversation, because what does it really mean real-time? Where is the line that divides real-time from not real-time?

But, also, because a lot of data teams are working with batch processes and have been doing that for many years. And a lot of people immediately say, “Oh, we don't need real-time for a use case.” The thing is that real-time is so crazily better than not real-time, that once you start solving problems in real-time, first, it’s super sticky. Because you start thinking, “Wow! I didn't even realize this was possible.” And you start thinking, “Hey, what else can I do with real-time?”

But also, it opens up – When you sort of reduce the order of magnitude of time that queries take, it opens up new ways of working both from the point of view of development. It’s not the same if your query takes a few milliseconds when you're developing than if it takes 30 seconds. Your experience of development is very different.

But, also, from the operational point of view, if you can know exactly what's going on right now with your e-commerce, or the marketing campaign you've just put out, or the players in your games, whether they've just hit a specific goal or something like that, and you can react to that automatically, it really changes the way that companies operate. And we've seen that repeated over and over. How they start thinking about their business in a different way. And it opens up new opportunities.

So, when we talk about real time, we talk about the combination of where data is being generated. And when it's being consumed. And we're talking about sub second, couple of seconds, there. That's enough to consider it real-time and to enable all of those new ways of working.

[00:08:21] AD: I love that just really tightening that feedback loop, how it just changes behavior. It reminds me of like software release cycles and how that's changed. It's not a quarter release anymore. You're just doing continuous integration. And how that it's not just speeding up. It's changing how you build stuff and iterate and all sorts of things.

[00:08:37] JS: Exactly. It's a new paradigm. And it's really difficult to explain until you actually use it. One of the beauties of Tinybird is – You know, how, for instance, Kafka, it enables you to ingest data and you have those topics that you can consume at different speeds by different consumers. So, it's great in the sense that you don't have to decide in advance what you're going to do with the data. So, Tinybird works like that as well. Like, you can start ingesting in real-time. And so, one use case. But actually, because the data is there, and writing a new use case is just one SQL query away. And then you activate an API and then you can start consuming it. It drives a lot of usage. You can bring in other data to enrich it. So, we've seen great expansion within our accounts, because, first, the real time aspect of it. And also, because it's so quick to develop and put new things in production that it becomes a sort of a no brainer to do more things. So, it's been beautiful to see.

[00:09:39] AD: Yeah, awesome. I love it. Okay, I want to dig into the architecture, the technical bits a little bit. I mean, first of all, like what's happening under the hood with Tinybird that like power these? Performantly serve these huge OLAP queries. Like, what's going on there?

[00:09:51] JS: Yeah. So, we use ClickHouse as our OLAP database under the hood. And we take advantage of a lot of the sort of real-time capabilities of ClickHouse. And we contribute to them as well.

[00:10:01] AD: Can you tell me in a second about like what is ClickHouse? Yeah, maybe just for people that aren't familiar.

[00:10:06] JS: Yeah. ClickHouse is an open source database that came out of Yandex. Many years ago. They built it to basically do website analytics. Do click house to manage, to keep track of clicks, and events, and so on. It's great for all uploads for time series. And it's designed to scale horizontally and vertically as well. So, it's really powerful.

But we always say, it's a bit like a Formula 1, in the sense that if you have a team of specialists that have been doing this for a long time and you have the Formula 1 driver, you can make the car run at 300 kilometers per hour. Tinybird is basically taking the best of ClickHouse and helping any driver drive a Formula 1, let's say,

[00:10:52] AD: Yeah, I love it. I remember when ClickHouse was kind of new, and I was using Redshift at my job for something. I was like, “Oh, we wanted something a little faster. We wanted to try it out.” We tried it out. It's was really fast. But we're just like, “There's so many moving parts and components here.” And I don't know enough about tuning it to make it work. So, awesome to see that as a hosted solution.

How did you, I guess, sort of getting involved in ClickHouse house and figured out, “Hey, I want to start this company, Tinybird, that manages it.”

[00:11:17] JS: Yeah. I should say here that I'm not the only founder. We're five-founder team. We've been working together for a long time. We used to work at a company called Carto, which is in the location intelligence space. And at the time, they use Postgres. And we were struggling to meet the demands of our customers who were coming with an order of magnitude more data every year, let's say. We started thinking about how to solve that. How to cut through all of that? A thousand times the data. And that's how we start investigating.

Carto went in a different direction, more about building on top of other databases, rather than building their own sort of platform. And we started leaving, but finding the same problems in other companies, is whenever there are huge amounts of data, and you want to build something on top, people throw cathedrals of infrastructure at it. Also, there's been this whole conditioning going on about the modern data stack and all tightly sort of divided up in verticals and components that VCs can invest in. But actually, that doesn't lend itself great to real-time. And it comes with huge costs, because every handoff, you're paying the toll. And basically, we wanted to do away with all of that. We wanted to work as developers, which is, “Hey, I love the idea of I have data. I write some SQL. And I can make something of it.” So, that's Tinybird in a nutshell. It's the desire to essentially work with any amount of data the same way that we work with smallest amount of data.

[00:12:50] AD: Awesome. And I assume a lot of people listening are familiar with Postgres, MySQL, and just SQL generally and how it works. But like you're saying, when you get up into those orders of magnitude, more scale, it doesn't work. So, what is ClickHouse doing under the hood that gives the same interface, SQL interface? But, like, under the hood, what is it doing differently than a traditional OLTP database?

[00:13:11] JS: The biggest difference is that it's a columnar database. And essentially, that means that the data in the disk is stored for columns rather than per rows. So that makes a lot of sense for analytical queries. Because if you want to add up, do a sum of numbers, you don't have to go row by row to find all the certain columns. You basically read all those numbers in one go and become – It is much, much faster.

And then it's built for scale. Like, every single analytical function, like a sum, or a count, or a unique, they're all designed specifically to scale and to work in such a way that also queries can be distributed across different nodes. So, you can have 1 billion records and you have four nodes, you can divide the data into a quarter of a billion records in each machine, calculate that, and then put it together. So, that's what can help you scale to massive amounts of data. So, the combination of those things, and a bunch of other things here and there. But those are the main architectural designs, let's say, our approach that helps you do that.

[00:14:22] AD: Yeah, absolutely. That's great. So, based on that, we have a columnar rather than a row-based one. We also are splitting and trying to paralyze this work. Are there any, like, RDBMS patterns that people are used to using in OLTP databases, but then are inefficient, don't work and ClickHouse, need to change how that works?

[00:14:39] JS: Yeah. I mean, because the data is stored in columns, that means that things like updates or deletes are harder. Because you need to specify and specifically each row in every part of the desk to get rid of it or to update it. Although it is possible to do, it can't be done at scale the same way that If you can do inserts or queries in ClickHouse.

In Tinybird, we've solved that. We've added a sort of a layer on top of Clickhouse to the product that enables you to do large replacements and deletes more from the point of view of mistakes are made often. Like, I've been ingesting the wrong data. I want to replace all of this data. And so, that’s very easy to do in Tinybird.

And then from the point of view of designs and sort of the difference between RDBMS, things like that are trivial or part of the normal way of working. Like, doing upserts, for instance, or deduplicating stuff. That's not trivial to do in ClickHouse. Depending on the use case, you might want to do differently. It's possible, but you have to solve it at query time more often than not. It has some ways of doing it. But it doesn't guarantee deduplication. So, we help a lot of our customers to find the right strategy depending on the right sort of on the volumes and the use case.

[SPONSOR MESSAGE]

[00:16:08] JM: Here's a puzzle. What do products like Dropbox, Slack, Zoom and Asana all have in common? The answer is they were all successful because they became enterprise-ready. Becoming enterprise-ready means adding security and compliance features required by enterprise IT admins. When you add these features, enterprise users can buy your product, and they'll buy a lot. These features unlock larger deals and faster growth. But enterprise features are super complex to build. They have lots of weird edge cases, and they typically require months or years of precious engineering time. Thankfully, there's now a better solution.

WorkOS is a developer platform to make your app enterprise-ready. With a few simple API's, you can immediately add common enterprise features like single sign-on, SAML, SCIM user provisioning and more. Developers will find beautiful docs and SDKs that make integration a breeze. WorkOS is trying to be like Stripe for enterprise features.

WorkOS powers apps like Webflow, Hopin, Vercel, and more than 100 others. The platform is rock solid, fully SOC-two compliant, and ready for even the largest enterprise environments. So, what are you waiting for? Integrate WorkOS today and make your app enterprise-ready.

To learn more and get started, go to softwareengineeringdaily.com/workos. That softwareengineeringdaily.com/workos.

[SPONSOR MESSAGE]

[00:17:51] JM: Are you struggling with broken pipelines, or missing data, or stale dashboards? If this resonates with you, you're not alone. Data engineer struggling with unreliable data need look no further than Monte Carlo, the world's first end-to-end, fully automated data observability platform. In the same way that New Relic and Datadog ensure reliable software and keep application downtime at bay, Monte Carlo solves the costly problem of broken data pipelines.

Monte Carlo monitors and alerts for data issues across your data warehouses, data lakes, ETL, and business intelligence, reducing time to detection and resolution from weeks or days to just minutes. Start trusting your data with Monte Carlo today. Visit softwareengineeringdaily.com/montecarlo data to learn more. That softwareengineeringdaily.com/montecarlo data.

[INTERVIEW CONTINUED]

[00:18:54] AD: Also, I know like a lot of OLTP people. They're like, "Hey, if I need an access pattern that I want to make faster. Or I add an index," things like that. I mean, does ClickHouse, Tinybird, does it have the notion of indexes? And if so, why not?

[00:19:06] JS: Yeah, basically, it boils down to how you sort the data. There's a sorting key in ClickHouse that you can specify. That's really important when you have huge volumes, because, for instance, if you're building analytics for your users, you have a SaaS product, for instance, you're going to have a company ID or a customer ID. And then you're going to have some dates. And then you're going to have some events that you want to query.

So, if you sort it by company ID, and by date, whenever you query, ClickHouse is going to find exactly –Even if you have billions of rows, it will only read the rows that refer to those companies. So, the way you sort the data can be important. And then that becomes especially important as you iterate. And that's why at Tinybird we also spend a lot of – Invest a lot in helping users iterate. Because it's not easy to iterate when you have billions of rows and data coming in all the time and queries happening against your table and so on. So being able to iterate on those as you learn what you need, it's really important as well.

[00:20:08] AD: As I was looking at Tinybird, I saw this concept of pipes as I'm sort of getting data in. What are pipes? Why are they important? How do I use them? How does that work?

[00:20:17] JS: Yeah. So, when we started TinyBird, the first iteration of it, it was not like a data – Like a sequel editor. And that was it. The problem is, as things get complicated, you start getting queries that are like this big. Really long queries that are hard to parse, that even for the people that have created them, the next time they come around, you need to sort of figure out what is this doing kind of thing.

So, with pipes, it's a way to chain queries. So, you can – For instance, let's say you first want to filter data. Well, you filter the data first in one of the nodes of the pipe. And then in the next node, you can query the results of the filter that you've done before and start aggregating. And then maybe you want to do – In a different node, you want to query some other data source, and you want to join the result in a different node. So, you can do joins between nodes. You can do unions. And that these pipes are bit like Jupyter Notebooks, but in SQL. So, you can comment as you go. You have a way to enter comments and to document what you're doing, which we found really helps with development and with bringing the rest of the team along to what you're doing, and so on. So, it's become a great way to develop, and explore, and work together in data projects.

[00:21:33] AD: Gotcha. Okay, so pipes are – They're a read time thing. They're not changing how I'm ingesting or like organizing the data differently – It’s all happening at real-time going through those pipes?

[00:21:43] JS: I should have started there. Pipes, you can use them for different things. One of the things is creating API endpoints. So, you can write a query, click a button, it becomes an API that you can start integrating. But you can also use them to materialize data. So, instead of saying, “Hey, the output of this query, I want it to be an API.” What you do is, the output of this query I want to materialize into another data source or a data set, that you can then query from a different pipe, for instance.

So, pipes are very flexible in that sense. And you can use them for these things. And we're working on other things like using the result of those pipes to send them somewhere else. Not even to materialize, but to send them to a S3 bucket or a Kafka topic. So, as you're getting data, you can transform it and send it somewhere else as well.

[00:22:27] AD: As I'm working with ClickHouse, and especially thinking about joins, does it make more sense to sort of do joins at read time? Or should I be doing that beforehand joining, denormalizing, having a wider table that can filter directly on that? Like, what makes sense there?

[00:22:41] JS: We have a bunch of rules. There's a bit of a mindset you need to get when working with – In general, if you want to scale queries, not just in ClickHouse, but in columnar databases. But with Tinybird, we always sort of encourage our users to think in a particular way. One of them is, first, if you're going to filter, filter first. Because apart from the sorting key we're talking about earlier, if you're not going to be using some data, just filter it out as soon as possible. Then if you're going to do joins, join as late as you can.

So, if you can filter out the data, then do your aggregation and then join, then it's going to be much faster than when you join first and then aggregate, because you will be doing it with smaller amounts of data. So, it's not so much that you can't do joins. You can do joins. No problem. It's how do you make them fast? And there's a bunch of rules that we are sort of first educating our users on, but also bringing into the product in terms of paradigms and guidelines that you should follow to make your API's fast, basically.

[00:23:47] AD: Yeah. Interesting. Do you have like query planner type statistics or different things where I can see, "Hey, I ran this query." And like you're saying, maybe I'm using a join inefficiently. How can I debug that? Or what does that look like for a user?

[00:23:59] JS: Yeah, yeah. We invested a lot in providing observability on top of what you're doing. So, in the very same pipe, when you run a query, you can see how much data you're scanning. How long did it take? How many cores did it use? To see if it's being distributed or not. And then you can – Also, once you publish your API's – By the way, all the exploration queries in Tinybird are free. So, we're not going to charge you for while you're developing, let's say. So, in the pipes, all of that is free.

But when you expose an API and then you start using it, you're going to want to know how much data am I processing? How fastest is going? how many requests do I have? So, all of that data, in real-time, we dump it back into your account so you can query it with pipes. So, in the same way you query your own data, you can query the data that your account is producing. So, it's very meta. Everything you produce, we enable it for you to query as well.

And people are doing amazing stuff. Like, for instance, they're building API's over those data sources and then integrating with Datadog to monitor their usage or to monitor performance and things like that. So, it's become super – Like, the probability layer is a very important part of our product.

[00:25:15] AD: Awesome. I love it. I think it's so interesting to learn more about like columnar databases and how they're different than the row-based stuff, and how it really twists your mind. Like, a bunch of things you mentioned. One other thing that always stands out to me, if you have like a sharded database, whether that's a NoSQL one or even like Vitess, or something like that. Often, you want to try and hit exactly one shard. Find the exact item you want and pull that record back. But like you're saying, ClickHouse is also sharding, but you sort of want to use up those – Like, make sure each shard is doing fourth of the work, if you have four nodes. Like you were saying, easier. So, just like some of the paradigm shifts, moving between OLTP, OLAP is interesting. And I think educational work is just such a big part of what you'll have to do, and just teaching people how to use those things.

[00:25:56] JS: And it's a deep topic. Like, it's fractal. The more you learn, the more you realize there's more to learn. But at the same time, there's a set of guidelines, again, that once you sort of master and you start thinking like that, you get really good performance, at least with Tinybird. I can judge for other – But you get really good performance out of the gate.

And then for bigger accounts and so on, we have different plans. But for big accounts, we also enable sort of services. As in, we call it Premier Support, which is, essentially, there's a name data engineer you can talk to in Slack that basically knows your use case, but also knows the product inside-out. Because we want to ensure that you can – We never want someone to be annoyed because they're paying too much because the queries are consuming their own data. It doesn't make a difference when you're dealing with small amounts of data. It won't move the needle. But if you have billions of rows, you're going to want to optimize and so on.

And then now we're also doing something we call the Jumpstart Package, which is for people that just want to try. And basically, it comes with a data engineer as well to help you get to production. And then you can stay on that package or move on to other packages as long as you want. Because I think there's people that sort of – Because they don't control the medium, just having someone that says, “Hey, look, try this. Try this. Start with this start kit.” It really helps a lot and gives people a lot of confidence.

[00:27:31] AD: Yeah, I love it. On that same note, you talked about packages, plans. Like, what does pricing look like for Tinybird? Is it usage based? Is it monthly? Or how does that work?

[00:27:39] JS: There's a free plan. You can start on a free plan. And basically, it includes a thousand requests a day. So, as long as you build something small and you want to – For personal use and things like that, very rare that you'll hit more than a thousand requests a day. And then you want to remove the limits. Basically, you put your credit card in. And then we charge by two levers, let's say. One is processed data, which means how much they do scanning or writing. And also, storage. Although, storage is something that over time normally is the smaller part of the bill, let's say. But we're sort of working towards trying to remove that entirely to make storage as cheap as possible, so that you only have to think about what is it you’re building? How much you're using the system?

[00:28:26] AD: In terms of architecture, I mean, it is like a multi-tenant system that you're running? Or are people running on shared machines? If someone provisions a new instance of Tinybird, you're going to set up four nodes somewhere for them. And they're just for them. What's that look like?

[00:28:40] JS: We have multi-tenant infrastructure out of the gate. So, if you sign up in multi-tenant infrastructure. But then for bigger deals, let’s say, sort of an enterprise deals, then we have a lot of dedicated infrastructure provisioned. But it is completely transparent to you. As in, you don't have to do anything. Basically, as part of becoming one of those customers, you go into dedicated infrastructure. And then you're only sharing, let's say, the front-end, let's say, and then everything else, it's your database resources and such that you don't have noisy neighbors or anything like that.

[00:29:18] AD: Yeah, absolutely.

[SPONSOR MESSAGE]

[00:29:36] ANNOUNCER: Did you know that 17 out of 20 people stop watching a video because of stalling and rebuffering? Don't let your users click away to a competitor site. If your business lies in online media, rely on CDN77 to deliver a seamless online experience to your audience.

CDN 77 is one of the leading global providers of content delivery network services. They power the world's most popular websites and apps, such as Udemy, ESL gaming, live sports TVs, or social media platforms. Aside from their massive and redundant global network, you're going to love their no bullshit attitude and skilled team of engineers ready to help 24/7 No chat bots. No tickets bouncing around unresolved for days. Just people who know your use case, and can immediately help you pinpoint and fix the problems. Don't wait until your users run out of patience. Go to cdn77.com/sedaily and ask for a free trial with no duration or traffic limits.

[00:30:54] ANNOUNCER: To cybercriminal, the most desirable credentials are the ones with full access to an organization's most sensitive data. Yet these highly sensitive credentials are almost always hard-coded into source code, CI/CD systems, or config files. Keeper Secrets Manager is a fully-managed cloud-based and zero-knowledge platform for securing infrastructure secrets, such as API keys, database passwords, access keys, certificates, and confidential data. Organizations can get Keeper Secrets Manager up and running in about 20 minutes, which is less than the time it would take to just download traditional solutions appliances. Manage your infrastructure secrets without any added complexity, maintenance or security issues. Try Keeper Secrets manager for free today. Get started by visiting keepersecurity.com/se daily.

[INTERVIEW CONTINUED]

[00:32:03] AD: As a developer, I love the trend towards more like fully-managed databases and stuff. And it's great. And you've been doing this now a couple of companies, both Carto and here. What's hard about running a managed database solution? What are the difficult parts that you've sort of learned?

[00:32:17] JS: The first two things that come to mind are, first, that when you put a platform out there with a set of API's and a database and tell people, “Hey, use it however you want.” People do the craziest things, and use it in ways that you didn't even imagine it could be used. So, that generates a lot of tension, because, suddenly, you had designed your architecture thinking of certain use cases, and suddenly you're hitting limits and problems that you haven’t sort of designed for. And that's constant. That happened sort of the first day. We had a customer and continues to happen to this day. Because the more customers you have, the more things they'll try to do and more pressure to bring to bear, let's say. So, that's something you sort of have to manage and sort of start to put rails for people to try to go through without losing the flexibility that we want to provide. So, certain limits in certain types of operations to ensure that things remain within reason and so on.

The other part, also, is there's a part of the market that is – As you were saying, like you're excited about things becoming fully-managed, and serverless, and so on. And there's a whole – We see a big trend in that direction. But some companies, and for policy reasons, and for – They want to have it sort of hosted and so on. So, that's things we also, from a go to market point of view, we come across. But anyway, it's all part of – It's not that we didn't expect this. But we know it from Carto. Same problems. But that is something you have to also think through. And what's your strategy there? And so on. So, those are the things that come to mind.

[00:34:00] AD: Yeah, awesome. We talk a lot about ingesting data, querying data, things like that. Where does the data come from? Like, how do people get data into – I imagine they're just not hitting an API one by one with 250,000 rows a second. So, how do they get that data to Tinybird?

[00:34:14] JS: You'd be surprised. But, well, for streaming data, the way we started was with CSVs. We had an API. You'd send a CSV. That was like three years ago. The reason we did that is because CSV is the universal currency for databases. Any database can produce a CSV and so on. So, we thought, “Well, it's always going to be a good default.”

But very soon, we saw that the push was towards streaming. And we built a Kafka connector. That Kafka connector is through which we receive our biggest loads at the moment. But also, we found that there are people that don't just – They don't want to go through the hassle of either hosting Kafka or have the added expense of hosting the Kafka somewhere else. So, we've built a streaming endpoint. We call it the events API. And essentially, it's an endpoint that you can send one or more NDJSON objects in every request. So, you can send maybe 2000 objects every request, or one every request, however your use case is. And we ingest that in with very high frequency at an amazing scale. And a lot of our users are using that interestingly. Because even if sometimes they're using Kafka or Kinesis, actually, it solves a lot of problems. It simplifies setups a lot. It's just a trivial posed to us. And that's really it. That's another great way.

And then we have connectors to other data warehouses, like BigQuery, or snowflake, and so on. And those are more to bring dimensional data, because customers already have data sitting in databases, like a products table, or customer plans, or whatever. And then, with those connectors, we just keep the data synchronized so that you can do joins, and you can enrich in real-time when you ingest, and things like that.

[00:36:05] AD: Very cool. What about ingesting – Like, tailing from OLTP sources? Whether it's MySQL replication log, or DynamoDB streams, or some like that? Is that something you all are looking at? Or what's that look like?

[00:36:17] JS: We did our own first implementation of the CDC connector as well. But it's working for a couple of customers, but we're not going to continue that. For CDC approach, we're going with Debezium. Because it's battle-tested already. It can go through Kafka. We can do sort of different approaches there.

The problem with CDC and scaling CDC is not so much – The Binlog problem, let's say, it solved this. Debezium does it really well and others. The problem is because the data that you’re getting from Binlog is not just inserts. It's also updates and so on, deletes. It’s is how do you deal with that in the OLAP site. So, it's very easy to – If you have an append only table, super easy to replicate and set up. It's harder if you have updates and so on, and you want to keep like an always up to date version of the data. And that's something that we're working on to facilitate as well, so that it's transparent, and so on.

We do it now following some like sort of micro batching and sort of materializing in batches and so on. So, we can solve the problem right now. But it's not ideal. And that's something we're working on as well towards trying to solve in a more transparent way. Because it’s – I mean, if you think about it, you wouldn't want that. It’s like connecting to having a scalable replica for analytical queries or whatever you're doing in your database.

But we also use tailors for other things like logs and things like that. We have a bunch of use cases building real-time analytics over logs. We have a tailor that we open source called tbtail that you can connect to any log and just send to our events API, so that you can build a web application firewall or charts over your logs and so on and connect it to other data sources and things like that. So, pretty useful as well.

[00:38:17] AD: Yeah, very cool. I love it. Okay, I want to close off with just a little bit about the company, the future, things like that. So, tell me about Tinybird. Like, how long have y'all been around? How big is the company? Where y'all base? What's that look like?

[00:38:28] JS: Yeah. So, we officially started sort of by mid-2019. So, three years ago. And we're now almost 60 people. We've raised a bunch of money with CRV, and Crane, and singular, and which amazing investors have been great to us. And we are – Now, we're a US-based company. We started in – We're all Spanish founders, but the headquarters is in the US. And the company is growing a lot here. And I'm living in the US now. And that's where we're at. And we're seeing huge opportunities and huge interest in sort of real-time data storage and processing and so on. And yeah, we're here to capture all that as much as we can.

[00:39:12] AD: Yeah. Well, at three years, y'all have some good logos on the site. And that's a lot of just like core infrastructure that you need to build up before you can even start taking on users. So, y'all have done a great job there. Last thing I want to ask you is like what future improvements? Any target areas you're looking at, or features, or different things? What's on your roadmap?

[00:39:30] JS: Yeah. There's a bunch of things. One, obviously, one of them is we want to be really cloud-agnostic. And we started in one cloud, which is Google Cloud. And now we're – Very soon will announce another cloud. And we basically want to go for all of them. Because people have very strong requirements about that. Like, they want to be in one cloud or other than want to be paying egress costs moving data around, things like that. That makes complete sense. And we don't really care. We want to be where our customers are.

We're also lowering the barrier of entry and making sort of connectors to pretty much anything that people use is also really interesting to us. Because, again, the experience that we want to provide is that you can build an API literally in a couple of minutes. And part of that is saying, "Okay, my data is here. How do I post it here?" So, we are investing in that a lot.

And then another constant source of improvements for us is in the iteration of – and how you build data products as a team with tests and the same way that you build software development projects anywhere. And if you think something like Rails, for instance. Like, Rails has migrations. When you want to put something in production, there's a way to do those things. With analytical data, it's not as straightforward, because maybe a Rails application database is, I don't know, one gigabyte if – and I'm pushing very high. But analytical databases can be terabytes of data or more. So, when you want to make changes, it's not as simple as, "Hey, I'm just going to run a query that is going to turn this column into something else." It can be very complex. So, we're investing a lot in making that trivial, safe and easy.

So, that's a big, big technical challenge. And we're obviously hiring. So, anyone hearing this, we're always hiring.

[00:41:34] AD: Awesome. Well, I'm glad that you're solving these problems so that I don't have to. I can just use the cool stuff. And I really appreciate you coming on and teach us how ClickHouse and Tinybird are thinking about columnar databases and parallel databases as compared to OLTP stores. I think this was really useful to a lot of people. So, Jorge, thanks for coming on talking about Tinybird. And best of luck to y'all going forward.

[00:41:55] JS: Of course. Thank you for having me, Alex.

[END]