EPISODE 1266

[INTRODUCTION]

[00:00:00] JM: Cloud data warehouses are databases hosted in cloud environments. They provide typical benefits of the cloud like flexible data access, scalability and performance. The company, Firebolt, provides a cloud data warehouse built for modern data environments. It decouples storage and compute to operate on top of existing data lakes like S3. It computes orders of magnitude faster performance from gigabyte to petabyte scale by using a columnar data structure, vectorized processing, just-in-time query compilation, and continuously aggregated indexing. Firebolt scales with data lakes by processing queries across clusters of nodes in parallel, providing consistently fast processing and granular control over resources. In this episode, we talk with Eldad Farkash, co-founder and CEO of Firebolt. Eldad was previously a venture partner at Angular Ventures, and a founder CTO and board member at Sisense before that. We discuss big data, data warehouses, and the unique benefits offered by Firebolt.

A few announcements before we get started. One, if you like Clubhouse, subscribe to the Club for Software Daily on Clubhouse. It's just Software Daily. And we'll be doing some interesting Clubhouse sessions within the next few weeks. And two, if you're looking for a job, we are hiring a variety of roles. We're looking for a social media manager. We're looking for a graphic designer. And we're looking for writers. If you are interested in contributing content to Software Engineering Daily, or even if you're a podcaster, and you're curious about how to get involved, we are looking for people with interesting backgrounds who can contribute to Software Engineering Daily. Again, mostly we're looking for social media help and design help. But if you're a writer or a podcaster, we'd also love to hear from you. You can send me an email with your resume, jeff@softwareengineeringdaily.com. That's jeff@softwareengineeringdaily.com.

[INTERVIEW]

[00:01:58] JM: Eldad, welcome to the show.

[00:02:01] EF: Thanks for having me. Great being here.

[00:02:04] JM: So you work on Firebolt, and Firebolt is a cloud data warehouse. And before we get into exactly what you do, I'd like to talk about the category as a whole. We've done a number of shows about cloud data warehouses. We've covered Redshift, and BigQuery, and Snowflake, and I just like to get your perspective on why the category of cloud data warehouse has become so important.

[00:02:30] EF: Tons of reasons. But one big one is that if you look at how data is being moved and served today, every data pipeline ends up at a place where you need to analyze the data. So I think a few years back, we had to wire that stuff up. And now with cloud native data warehouses, that becomes simpler. So suddenly you see so many people who are using other things, getting in and setting up the cloud native data warehouse and just spilling the data into that. So it's oversimplification of everything we've done to get analytics and data serving. And cloud native data warehouse bring a lot of new things to the table. So I think that's why we see so much of cloud data warehousing. That's why kind of the biggest market goes to the cloud native data warehousing. If you look at Snowflake, which is crazy, but it's perfectly deserved. So yeah, so if I had kind of to pick between a key value store and a data warehouse, the data warehouse will win big time. There's a very small set of things if you deal with a key store, but there's tons of stuff to do with a data warehouse.

[00:03:46] JM: What's the typical data flow that you see these days between the transactional data layer and the data warehousing layer? Like are you seeing data being streamed from the transactional layer into the data warehouse? Are you seeing batches of data being moved from the transactional layer to the data warehouse? And what is that ETL pipeline generally look like.

[00:04:12] EF: So I think, as usual, we have kind of multiple ways to move data around. So if you're coming from legacy, I would expect you to first start with batch-y kind of way of moving the data, which is perfectly fine. Today, there are easy tools to do that. You would usually use the lake to stage the data. So no matter where the data comes from, which operational store, it’s logs, it's a key value, whatever, you'll push it into the data lake and then you'll have your periodical kind of pull from the lake into the data warehouse. This is the common way and actually the most used one by far. So even though you hear a lot about Kafka, Kafka is still just getting started when it comes to analytics.

Now, over the last few years, data warehouse vendors started to offer direct streaming support, which makes tons of sense, because why would you need to stage the data? Why do you need this kind of iterative process, which is very error prone, which is slowly responding to changes in data, which kind of users get all the data served? So, of course, training is something that everyone is looking for. And I think if you look at cloud native data warehouses going forward, this will be the standard way to do it. So you always start by actually picking the formal data source. So you won't even necessarily pick your Kafka. You'll pick your Sumo Logic. You'll pick your web servers that generate the logs. You'll pick those things, and it will just start to stream. So yeah, but we still don't see that as much as we see batch. On the other hand, when we work with kind of cloud native companies, companies that had their data born in the cloud, and most the team are data engineers, then Kafka is king. And in most cases, they immediately ask, "Can we just plug Kafka into the system?" So that's kind of what we're looking at these days.

[00:06:18] JM: In terms of the applications of the data warehouse, is it more based around reporting or interactive analytics, or machine learning applications, or maybe even real-time monitoring? What are the most popular applications that are built on the data warehouse these days?

[00:06:46] EF: So pretty much most of the stuff you've mentioned, maybe real-time monitoring would be out of the scope usually. There are better tech and products to use to do that. But I think what really changed is we kept the same patterns. Like we have reports which are batch-y, we have ad hoc dashboards, which are less predictable. We have a feature stores that serve stuff generated by AI and ML. So we have the same things. The thing that really changed is the data and kind of the requirements. So a few years back we used to split up internal versus external. We used to say, "You're building a customer-facing," something that's customer facing, or you're building internal dashboards. And it's obvious that for internal stuff it's easier, right? You're much more flexible. You can give up on speed. You can play with the metadata. People can be more understanding. But when it's customer-facing, it demands latency, concurrency, scale, consistency. So there was all of those things that made customer-facing a big deal.

Of course, from a product perspective, you would really need engineers to build up kind of a data serving solution when it comes to customer-facing. But if we fast forward today, customer-facing is as challenging as internal and vice versa, because if you look at cloud native data companies, they use the same data sets. If you look at the data lake, it's the same data. It's serving customer support. So if you're a kid and if you think that you were cheated out of the amount of credits that your gaming company gave you, you send them an email, and they reply, and they tell you how much you really need based on the amount of, I don’t know, weapons you picked up or drop or jumped. And they do that by actually running queries against the same data that the user itself sees. So you have a big data lake. And you have internal apps, customer success, support, operations, building heavy stuff that is near real-time, highly available. They demand the same thing. So I think that's the biggest change we've seeing, that because of the data, because of how the data changed, we see less offloading and more kind of using the same data lake for stuff we would use a classic legacy data warehouse for before. So kind of aggregating and trimming down resolution of data works on legacy use cases. But you'll rarely be able to see that if you're kind of working with actual big data, companies that actually have tens or hundreds of terabytes of data being served like that.

[00:09:42] JM: I did a show with the Snowflake, I think was CTO. It must have been like three years ago or something. But my understanding is that one of the key design challenges in building a data warehouse product is the tiered storage, and because the customer really does not want to worry about managing what data is faster to access and what data is slower to access. You really want a data warehouse that is basically capable of keeping all that under the hood and tearing the store. And basically, you throw everything into the data warehouse, and the data warehouse figures out what to put like in S3 style bucket storage, and what's on disk, and what's going to be like higher up the memory hierarchy. And it sounds like a really, really hard engineering problem. So I guess I'm wondering to what degree you agree with that statement that that is one of the canonical problems of designing a data warehouse and to what extent it's been something that's been hard to solve at Firebolt?

[00:10:51] EF: Absolutely. I think, yes, the Snowflake CTO was absolutely, and is still absolutely right. Getting consumption-based elasticity at scale is extremely complex if you want to oversimplify it. If you don't want to oversimplify it, then you just expose tons of parameters and wires, and you have experts doing that for you. But if you want to move data to places where they need to react fast and have no knowhow or expertise into how to really kind of perfect the experience, then it's the data warehouses job. And this is the biggest thing about data warehouse, is they oversimplify complex big stuff that we would work hard to accomplish before.

And at the core of that is this simple, yet trivial, yet so hard to achieve feature, which is, "I want to run something. Get me the resource to run it as fast as you can. I want to pay only for using that resource for what I'm doing." So it can be specific query. It can be a pinned isolated resource running dashboard. But the point is that I want my resource to be available fast. I want to pay for consumption, meaning only when I'm using the resource. And I want the resource to be optimized. What does it mean? It means that it needs to do stuff behind the scenes so that the experience gets better over time and that everything falls in place. And you've mentioned it, mixing between tiered storage. More specifically, mixing between cold storage like S3, and hot storage like local cache, and super-hot storage like RAM and CPU caches is extremely complex. And this makes a cloud native data warehouse unique. And Firebolt – Snowflake's design I think is revolutionary when it came out. For us, it was obvious that if you want to do cloud native elasticity, it needs to support those principles. So yeah, of course there is consistency. How do you apply transactions and metadata changes across multiple clusters, which belong to the same logical database? Because with disaggregated storage, there is no database. The database is just a logical concept. There is storage, S3, and then there is a schema running a query at a compute block. The compute block can be a core sitting in a farm of servers. It can be a dedicated cluster. It can be a GPU instance. So it's not just about kind of providing clusters. It's also about decoupling cost from compute. So you get a consistent cost experience. You pay X-amount of dollars for a consistent experience. Yet, behind the scenes, crazy stuff happens all the time. Things change all the time. But for you as a user, yeah, it feels like magic. And it's extremely hard to achieve.

[00:13:53] JM: The consistency question, how important is consistency in a data warehouse? Like I think of a data warehouse, the term that was used for data warehouse type analytics, or applications, at least like maybe four years ago, was OLAP, online analytic processing. And the sense I always got with OLAP was, "Okay. It's okay if OLAP is out of date a little bit. It's okay if OLAP data is a little bit messy," because you're usually doing aggregations, you're usually doing just like large scale analytics where it's not super sensitive. You're doing machine learning off of this, and it's okay if a data point or two is out of date or copied or something like that. How consistent do you need this data warehouse to be if you're doing aggregations across data where like some of the data is in S3 and some of the data is in like a in-memory storage system and you've done these rights at different times and maybe you've made some mistakes over time. Is consistency an issue?

[00:15:00] EF: So first, yeah, consistency is a big term. So, consistency, when defined, is fresh. How fresh is my data is not the thing here. I was actually referring more to consistency in terms of transactional and distributed consistency. So like a resource is like a thread. It comes and goes. It serves a part of specific query or workload, yet you have many of those running behind the scenes, and users change metadata and modify data all the time. But you have multiple compute resources serving the same table across kind of different use cases. So you need to make sure that if one compute resources, modifying something, that modification is safe. So it's like there's a transaction applying that, and it will be consistent across all compute resources for the same database. And if you're running one server, or if you're running kind of this big, shared nothing cluster, you don't have that problem. That problem exists when you switch to native elasticity where isolated compute resources are working together under the same metadata under the same data. So if two compute nodes will return the same kind of result, you will not have changing results based on the table or based on the resource that just serve you the result. And that is extremely complex, especially when you just spin up and close a resource sporadically. And in many cases, it happens kind of in a transient way. It just removes itself when your query is over. So that is kind of what I meant was part of the thing.

Like if you think FoundationdB, or if you think kind of all of those solutions and technologies, this is a native part of the cloud native data warehouse. So it's not just a columnar storage. It's not just a tiered storage. It's not just a vectorized, or just-in-time compilation. It's also kind of how you manage metadata, where metadata is also the data itself. Snowflake, they call it micropartition. Firebolt has different concepts. But in many ways, those are huge challenges that every cloud native data warehouse deals with.

[00:17:26] JM: So let's start to get a little bit into what you're building at Firebolt specifically. So you started this company in the midst of this growing popularity of cloud data warehouses. And I imagine it's really hard to stand out in the market. I just like to know a little bit about the business ideation process, how you came upon building a new cloud data Warehouse when there's already some really big players and what you were trying to do differently.

[00:18:01] EF: So first, I think that if you're at a market it's being fully, fully, fully owned by those huge dinosaurs, that's kind of the perfect market to open a startup. It means a lot. It means that the market is not innovating anymore. Now Snowflake is an exception, right? It went from being a startup to being a giant in no time. But I've been with data warehousing. I've been building databases like for ages, since I was 16. I've been in the stage. All my startups were around that. And when we started Firebolt, we wanted to solve what we believe was a unique and kind of growing problem, which didn't really concern cloud data warehousing, which was efficiency. And people often get confused between efficiency and elasticity, or fine-grained elasticity. So being able to pay only for what you consume has nothing to do with efficiency.

Yes, of course, if you run resources and you're not using them, you're just burning money. But efficiency means how fast, for how much you return a result to a user. It can be an insert, a huge Spark job. It can be a Looker dashboard serving hundreds of users. It can be data scientists building a feature. Building a feature today is actually running thousands of SQL queries. You run those queries. When you build a feature, it can take you up to eight months to kind of go from a feature to a data-driven feature.

So Firebolt’s vision and kind of origin comes from building something that fits data engineering more and less kind of a more of a legacy strategic thing mindset around data warehousing. We look less at the data warehouse as a product and more about how you use SQL and data to build features. And as I said, it can be internal and external. Both are getting as challenging as another, but that's going to the purpose of Firbolt, to be extremely efficient at scale and it's still for data engineers.

And to your previous question, how do you stand out? The truth is it's crazy. So the traction is unbelievable. And I think we touched a very painful point in this market, a big frustration around working so hard yet, ending up with Tableau extract, with a huge aggregation, or denormalization, or all of those things we've been doing for 20 years, because by the end of the day, you still pay 100k to run one terabyte of data. And with analytics, unlike other things, it's extremely challenging, because there's so many little things you need to solve to get it really fast. And that's what Firebolt is all about. Make it seamless. Get at data lakes scale, real data lake scale. So 10 terabytes is really not a lot. And you want to crunch these 10 terabytes as if it was less than one terabyte. And you want to make it efficient so you can have your data engineering team working on it together, versus having one person kind of writing an Excel, asking for a list of SQLs to run, getting permission. Because it's so expensive, constantly trying to figure out the efficient resource for the task. Yeah, getting data to be valuable at scale and be efficient is extremely, extremely challenging today. So Firebolt makes that simple. And that's why it resonates so well with so many use cases.

So people don't consider us as a rip and replace option. They shouldn't. Working along most companies we talk with, they either use or have been using multiple cloud native data warehouse for a good reason. It's super easy to trial it. There is no risk. You pay for consumption. They're all the same in terms of how you move the data in. It’s SQL. You do a copy into. You create an external table. You just move the data into your data warehouse, and they play with it. So that's kind of how Firebolt ended up getting all the traction we got just because we've focused solely on making efficiency at scale super simple.

[00:22:27] JM: And to compare it to the elephant in the room, Snowflake, what's the most driving difference between Snowflake and Firebolt? Like from the onboarding experience, to the pricing, or the efficiency? Like what are the major differences?

[00:22:47] EF: I think the first difference is that it's perfectly fine to ingest much more than you thought and slice it and work it just as if it was your kind of fully aggregated data. So raw data is something that your data warehouse can work with. The second thing is that many customers don't refer to us as a data warehouse. They say data warehouse is something that IT is using. Snowflake is something that you give it to offload their data in. They don't really connect to the data lake. And yes, it costs millions of dollars to build something that is really low latency. And those requirements for them have been everything for the last 10 years. So they're looking for something that fits that kind of mindset. And Firebolt feels like that. Just returns fast. It's efficient. Your first glimpse, you say, “How do they make money?” And then after a few months you pay maybe half of what you used to pay, but you do 5x more use cases on what you did and you have many more data engineers using it. Adoption is everything. And with consumption, with efficiency, you can predict adoption. And what you see is that it gets all over the place. So it's more distributed in terms of how you use it. And it's really fits a developer mindset.

[00:24:10] JM: Tell me a little bit about the engineering decisions behind Firebolt. Like what are some of the – Maybe you could just describe the architecture and then we can get into some of the engineering design decisions.

[00:24:25] EF: So as I've mentioned before, this is a disaggregated design, meaning S3 decoupled from compute, but also that you can spin up isolated clusters on the same data running concurrently. At its core, we have our own file format. It's a proprietary file format called FFF. It's like Parquet or Arrow binary. Of course, it's totally different. It was designed to serve two things. One is our own query engine. And two, efficiency. So indexing, sparse indexing, encoding and decoding, ordering of the data. Tons of stuff that is happening behind the scene for you is done through the FFF format. The goal of FFF is, for each file, to have a primary index. That's kind of the root of everything. You define a primary index on your table. Meaning that the fields and expressions that you put into the primary index will define how your data is ordered and how your sparse indexes will be generated behind the scenes.

Sparse indexes are hidden for you, but they're really, really important. They allow us to decouple access to ranges from the data itself. Meaning that when you run the query, we don't just do the normal partition, micro-partition and file pruning, we do a range-based pruning. Meaning we calculate the query over the sparse index. The sparse index, which is loaded in RAM on the compute layer returns ranges, not data points. Those ranges are being merged and scanned over S3. The result is that you have a polymorphic file, meaning you have an over-compacted, over-merged file on S3, which means because it's ordered, it is highly, highly compressed and efficient. On the other hand, when you scan the data, you don't download the file. Never. You download specific ranges, which can be tiny or huge. Whether you're doing a point query or huge scan, those ranges will represent that. And we download the ranges and we store them in a unique way within the compute layer. So we're moving away from an architecture that generate files and needs to kind of compromise between the size of the file on S3 versus our ability to update it. So we have this predefined size. Those files today are being downloaded, mostly pruned by partition or micro-partition, which is far, far, far from what we really need.

When you're scanning or using those products, you're scanning much more data than you actually need. So what Firebolt does is figure out giving the predicate you're using. So your where statements and joins inside your query. How to extract the exact the granular ranges, and scan only those over S3? So the result is orders of magnitude of data is being pruned out of scanning, okay? So your cold queries returned only what they actually need. And then it goes up into an HPC kind of second kind of part chapter where it's all about memory bandwidth, CPU caches, just-in-time compilation and authorization, which is kind of a whole different domain.

So on one end, we have storage, FFF, sitting in S3, growing and being compacted as time goes on. On the other hand, you have the compute layer that scan ranges within those files to avoid downloading files as they are. And this is a huge, huge change. It requires you to do a few very different things. So you need to order your data. If you don't order your data, you can’t create sparse indexes. So, for us, ordering the data is kind of a tool. It has two big things. One, is about compression, okay? And the second one is we want to join those sparse indexes. So we use the primary index to support that. The outcome is amazing. And the query engine is totally redesigned. Optimizations are totally redesigned to support kind of a core change in how we store data in a data warehouse. So think about it as having those files. And each one of them is ordered by one, can be 100 columns. It doesn't matter. But that's the core thing within Firebolt. I hope I kind of managed to explain it in a simple way.

[00:29:16] JM: No. You did. I think I follow you. Maybe we could go through the read path and the write path of data. So maybe you could talk about like a batch ingestion and then a query against the storage system.

[00:29:30] EF: So, input comes at changing frequencies. On one hand, you get this consistent tickling of streams, pushing data in. On the other hand, you have those batch S3 inputs. Each needs its own unique way. If your compute resource goes down when you ingest Kafka, it's a whole different challenge than when a compute resource goes down while you copy into from a data lake.

So kind of the first part is making sure your system can handle different types of ingest pattern. If you're adding a single row or a million rows, it will affect the FFF files that are being generated. We are consistent in terms that when you insert, it's immediately available even before S3 commit. So you don't need to wait for S3 to commit a data to query it. This is important for kind of a niche operational near real-time use case. But it's important kind of to distinguish between a Firebolt compute and other compute. We don't need S3 to expose the data. It’s already exposed. The microsecond you insert, files are being generated and are being compacted.

Our goal during the ingest, so everything that's happening behind an insert or copied into, is to take files, to take FFF files, and to regenerate them as bigger and more compact files. And we can commit those files to S3, depending on how you want to expose the data. So sometimes you want to say, “Well, I want to aggressively compact the data. It's a copy into statement, I don't care about near real-time.” On the other hand, some other cases, you want to say, “Well, the data is – Actually, I want to analyze daily data and history at the same kind of swish, yet they get constantly updated every five minutes, every minute. And I want to have a descent experience of both things at the same table.” So you need to go back and re-compact files. This is kind of similar to a data lake, of course, 100 times more efficient than a data lake. If you would build an LSM to merge and compact data today, it would look different than the one you would use with RocksDB or something similar 10 years ago, but the concept is the same. We can merge in compact data, okay? It's being done when data gets committed. It's being done over time over your S3, and it's crucial.

So there are many things that you need compaction and kind of merging for. But it also gives you the ability to ingest different data at different speeds and different pulses. Once the data gets ingested, it gets committed to S3, and then it's available to other compute blocks, to compute resources. So if you have three use cases running over the same table, all of them will immediately get kind of the same version of the same data once the data gets committed to S3. And once it is committed, that's it. For you as a user, it's just like any data warehouse. The storage layer, everything I've described in terms of sparse indexing, scanned ranges and stuff like that, is happening for you. You don't need to deal with it. And that's basically it.

Of course, over time, there is the dup, there's upcert, there is merging, there're tons of stuff, but it's all being driven by the same concepts behind the scenes. So you write a new file that represents a delta, yet you need to replace and merge other files to keep things efficient and to make sure everything is consistent. In other words, you don't read about it in the documentation in your data warehouse. You would usually read about it in your Spark deployment. You would need to know about it. But the personal data warehouse is exactly that, is to oversimplify stuff you’re worried with legacy systems.

[00:33:33] JM: You mentioned spark there. My sense is that there's kind of a dichotomy in the direction that people are going with their rapid access data management, and one direction is people going all in on Spark. And then the other direction is people going all in on snowflake, or name your data warehouse. Do you have that sense as well? Is the commitment to Spark in an organization mutually exclusive with the commitment to a data warehouse? Or do these serve kind of heterogeneous purposes?

[00:34:08] EF: So a few years ago, I would say if you're coming from data engineering, you would use Spark no matter what. And if you're coming from Vertica, or Oracle, you would use Snowflake. That was a few years ago. Today, everything is mixed. Today, they're competing on the same use cases. So up until recently, you would need Spark to drive Snowflake. You would use Spark to simplify your workloads, aggregate them, filter them, do some calculations over them. This would be the data engineer’s job. And then they would offload that into an S3 bucket, which would end up in a Snowflake, which is used by your marketing, and finance, and BI and more traditional teams. In the same company today, you will say their engineering teams using Snowflake, running DBT, completely replacing Spark. By the way, DBT this is one of this huge things that are changing everything. With DBT and a cloud native data warehouse, you don't need to run a Spark anymore.

Yes, of course, we're talking about maybe 80%, 70% of what you're using Spark for. There are many things you need Spark that you can’t use your data warehouse for. But if it comes to data serving, so, aggregating, filtering, joining data, doing kind of ELT stuff, why use a Spark? If you just use a DBT, it can be used by so many people in your org, and you just pick your data warehouse. You can switch between the data warehouse. You don't change your DBT script. So that is also a huge thing that's happening. Tools that simplify, use, exploit data warehouses, cloud native data warehouses, to simplify stuff that you needed a whole team to do a few years back.

So to your question, I think everything is getting mixed up. Databricks will be 100% SQL over the next five years. Snowflake is SQL, will be SQL, but will try to do more of a Spark-ish use cases. And yeah, we'll end up like fashion, it repeats itself. Your data warehouse, your big vendors will add more and more stuff. They’ll add BI. They will add ELT. It will become this huge thing that you can use for tons of stuff, which is amazing for many companies. But I think it's really about self-service. I think that what's happening now is about making data movement, data serving, data analytics more self-service. And that's why cloud native data warehouses are so important, because they're simpler.

[00:37:03] JM: And in terms of the market, I guess, it's just not zero-sum, because there's room for all these different players to grow. There's so much legacy. So many legacy companies that basically have nothing in the way of data infrastructure relative to the amount of data that they have. There're lots of new companies that are getting started that just have – From very early days, they'll be generating terabytes of data just because of the nature of their business. So I guess these are all just gigantic growth industries that probably even today look smaller than they truly are.

[00:37:42] EF: Absolutely, med tech, FinTech, EdTech, cyber, all of those industries are reinventing legacy with data basically. So if you look at those types of companies, every company will be this type of company one way or another way forward. The way they operate on data is just different. It's not reactive. It starts with the data. They build everything around it. It's not a side, an afterthought. It's not something we connect to our company so you can build something. It's something that we built our company on. And yeah, it drives different culture. It builds different – Built differently. I think the market is going to explode like crazy. Everything is everything needs to analyze the data. No matter what you're going to build, you'll need something to crunch your data. And we will see more companies, much more competition, much more innovation. There are amazing companies and startups out there. Definitely, I think we're super lucky to have been kind of building what we're building at this period. It's exciting. We've been building this company remotely. Everything is distributed, from the way your culture works. We have five offices. We're just 70 people. Your product is distributed. It's on the cloud. And yeah, as we see, the market really needs more, more of that. So yes, we're also lucky to be building what we're building at these times. Doing that 10 years ago would be very boring.

By the way, my previous startup, Sisense, was also started as a database company, an HPC database. Nobody even thought about buying the database from us. They looked at us in a weird way. It was 20 years ago. And it was just different. Today, you go online, you talk to the team, to the data engineering team. They're able to make a decision and try you out after 40 minutes. There's nothing else to do. It's consumption. There's no risk. So they can use their own budgets, and they can just move forward quicker. You use Slack to support customers immediately. Everything is different. And SQL drives most of it, and will drive most of it going forward.

[00:40:06] JM: What do you expect the competitive dynamics to be like as you get to a place where you are more and more in competition with a Snowflake? Like how do you expect to – Just in terms of like the macro competitive dynamics, like in terms of marketing and winning the big customer contracts? I'm just curious, considering you've run a database company before. I'd love to know what it looks like when it gets to the point where you really need to get big, big numbers in order to make the company grow and you really start to be colliding with the gigantic companies in the area.

[00:40:48] EF: So first, we qualify only for prospects that are using Snowflake, Athena, Redshift, and BigQuery. We will not speak with someone who has not been using one of those. Meaning that 100% of our business is coming from replacing use cases running on the best of breed, the modern cloud native data warehouse. If you're using Vertica, we will kindly ask you to try Snowflake first. If you're just starting with Snowflake, we'll wish you good luck. And we'll talk to you in a few months. And you’ll love snowflake.

We talked to many prospects, they love Snowflake, yet, Snowflake can’t solve all of their problems. So 10%, 15% of their workload, they need something else. So they look for it. We don't need to make a big marketing statement other than we can show you that it takes a few clicks to get your data into Firebolt. You'll see how to do that it's SQL, 100% SQL. And you can compare apples to apples. In fact, we only show kind of apples to apples. We give you the result. And you see the results and you compare it to what you're using side by side. That is very different from how we used to compare and evaluated warehouse before, okay?

And I think the reason is because we go after the data engineering teams that are just moving so fast, and they're not really connected to this kind of a strategic long-term unified data warehouse strategy, which exists in other industries. And therefore we are not selling to the financial industry. We do sell to cloud-driven financial product. So if you're in FinTech, you would use Firebolt easily. We run on AWS only. And we’re amazed to see that what we would consider to be big, huge enterprise deal a few years back is now a budget for data warehouse that an engineering team is running. And they don't need 50 people to approve anything. It's a 5, 10, 20, 30 people team, and they can make all the decisions themselves. It's crazy fast. And it fits to where we work with data now versus before. So yeah, we went online a few months ago. We went out in stealth, and we immediately started to sell. So you don't do a six month trial.

I think another thing that helps is ecosystem. Many times, it's being overlooked by database companies. And if you're into analytics, then most of your customers will use some kind of tool to speak to you. So Looker, Tableau, DBT. You need to be really able to support those. You need to have a very robust sequel layer to support those. So there're tons, tons, tons of investment that you need to make to call yourself a data warehouse. But once you pass that critical point, it's happening, actually happening all the time now. We're growing like crazy. And it's consumption. So Snowflake wrote the playbook for us, and we thank them for that.

[00:44:14] JM: I still want to understand like the market point of view more deeply. Like is there a market from the past that you can compare this to? Like, I don't know the database market in some time period, or the BI market in some time period where you have this set of really dominant players already, Redshift, and Snowflake, and BigQuery, which some people really like, where there is still room for a player in what some people might consider later on in the game.

[00:44:48] EF: I think actually, database market is one of those places where you constantly see huge dinosaurs being completely eaten by small fish. Snowflake is the latest one. But Redshift was before. Redshift was nothing, was this tiny experiment. And boom! Nobody's using Oracle for analytics anymore.

And then we had self-service in-memory, which replaced kind of the legacy way to do analytics. So I think, actually, with databases, not to mention open source and everything that's happening over there. But the database market is constantly evolving. It's constantly changing. It's amazing. And if you look today, being able to exploit the cloud, at an intimate deep level, makes the data warehouse even more fascinating. And that's why you constantly see innovation here. And it's happening. It's been happening all the time.

[00:45:48] JM: So okay, a few more things, a few more areas I want to cover. So let's talk about DBT a little bit. You mentioned DBT a couple times. And I think there are people who still don't quite understand DBT and why it is so transformative. Could you give your elevator pitch for DBT and how it's changed the market for data analytics?

[00:46:10] EF: So I will give mine and kind of the way I look at DBT through my experience with Firebolt, because DBT is huge and people do tons of stuff with it. But, for me, in parable, DBT allows people to simplify their pipelines dramatically and go from an engineering effort into this ad hoc, like LookML, like really simple and easy way to manage your data cadence, to version your data, to understand how your metadata is being applied and by whom. And DBT allows you to do that at scale as well, because it runs on cloud native data warehouses.

So that makes DBT a superpower for an analyst that depended on it, the Spark team, yesterday to prepare the data for him. And the data scientist, she had to wait for a week for the Spark team to kind of get her task prioritized. And now what they're doing is they're doing it themselves. They're depending on DBT and they're using the data warehouse in so many interesting ways behind the scenes to accomplish big, big tasks they wouldn't dream of doing before DBT. So it's not just DBT. It's the fact DBT runs on cloud native data warehouse, this decoupling makes DBT extremely dangerous for stuff we would consider complex before.

[00:47:39] JM: Okay, to close off any perspective on the future? I guess I'd love to know the future in terms of what you're anticipating in terms of the market, and also engineering decisions and company direction within Firebolt. What do you think is going to be built in the next 5, 10 years?

[00:48:02] EF: So I think one of the biggest things that will come in the next few years is hardware changes, actually, for a change. We've been looking at hardware on the cloud for the last five years at least. Nothing happened. FPGAs, and more importantly, GPUs, or more precisely, kind of the future of how GPUs are going to be built is going to change analytics as well. GPUs weren't really valuable for data warehousing for many technical reasons. And if you look enough, the grace architecture, for example, it's coming in, I don't know, a year or two years from NVIDIA. This is completely new. This gives us memory bandwidth to use the GPU for stuff that we hadn't been able to use before in our domain, analytics, OLAP, data warehousing.

A hash lookup, which basically is kind of half of your challenge when you run a query. So a lot of technical stuff that can be done much better on a GPU will start happening. And it fits cloud native data warehouse, because you decouple the compute, kind of the physical compute for the user.

So you need to know whether you're using a GPU and when precisely. So if I'm de-serializing Parquet with a GPU or using a GPU RAM LSM to do compaction, or if I'm using my GPU to decompress a table scan and then move that to the CPU to do kind of higher high-end operators, this is the data warehouse’s job. And I think it's exciting.

Firebolt, personally, there's tremendous kind of knowhow and background around HPC, GPUs, and we've been exploiting that with Firebolt. And we plan on doing that. And I think that's kind of a big exciting thing that we haven't been used to before in the cloud. So I think big change is coming. It might take a year or two, but definitely we’ll start seeing GPUs doing a big comeback. So this is not a niche GPU-driven data warehouse anymore. Like if you look at the benchmark, right? But it's really, really something else. This is one thing.

I think the other thing is self-service of data. Really taking those eight months you need to go to production and be data-driven and turn that into a two-week sprint. Make it efficient. Make it simple. I think that's something everyone appreciates. And we actually forgot how important it is when you're using data, data on a daily basis. So we need that a lot more. And yes, I think the ecosystem is evolving as well. So you don't need to talk to kind of those just huge legacy partners anymore. You're talking to tens of startups, whether they're kind of more mature startups or new startup like, from the Monte Carlos, the DBTs. So many interesting things happening on the ecosystem now that is exciting to see how it will evolve going forward.

[00:51:07] JM: Okay, great conversation. Congrats on the success.

[00:51:09] EF: Thanks. Thanks for having me.

[END]