EPISODE 1116

[INTRODUCTION]

[00:00:00] JMS: Business intelligence tooling allows analysts to see large quantities of data presented to them in a flexible interface, including charts, graphs and other visualizations. BI tools have been around for decades. And as the world moves towards increased open source software, the business intelligence layer if following that trend.

Metabase is an open source business intelligence system that has been widely adapted by enterprises. It includes all the common tools that are expected from a business intelligence system. Large scale data ingestion, visualization software and a flexible user interface.

Sameer Al-Sakran is the CEO of Metabase and joins the show to talk about Metabase's engineering design and usage.

We are looking for a writer and researcher to help with the preparation for these shows, and also to write articles. If you're interested, send me an email, jeff@softwareengineeringdaily.com. Also, if you are building a software company. I'd love to hear from you. I'm an investor, and you can send me an email at jeff@softwareengineeringdaily.com. Thanks.

[SPONSOR MESSAGE]

[00:01:09] JMS: Operations teams can find themselves choosing between two options, either take full control over the infrastructure yourself or give all developers permission to access production. The first approach increases the operations teams workload, which often results in overwhelming situations and ops becoming a bottleneck. And the second approach allows for rapid deployment of changes to production, but it causes a serious risk to infrastructure uptime.

With Octopus Deploy, operations teams have a third option. The Octopus platform can be authorized to run approved steps and play an intermediary role. Octopus delivers self-service without sacrificing control over production, and it also provides a comprehensive audit log of the changes that are being made. Developers can enable self-service with automation. By

© 2020 Software Engineering Daily

automating the processes that are forming a bottleneck, developers can free themselves from the waiting game.

You can check out the most frequently requested run book templates by going to octopus.com/selfservice. You can find those run book templates at octopus.com/selfservice. And I want to thank Octopus for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:02:24] JMS: Sameer, welcome to the show.

[00:02:26] SAS: Thank you. Thank you. Pleasure to be here.

[00:02:28] JMS: You're the CEO of Metabase. Metabase is an open source BI system. Why does the world need another BI tool? There are so many of them.

[00:02:38] SAS: I mean, I'm not sure it really does. I think one of the things about Metabase that's been interesting has been that we're trying to tackle – I mean, use us for BI, certainly. I think one of the things where kind of Metabase shines is this idea of a lightweight way to just create dashboards, share datasets and have to go work on them. I think if you had a pretty good idea of what you want to measure, who's going to consume it and why? There have been really great tools for about 50 years. I think the main space that we're trying to kind of clean up or make easier to use is this idea that I don't necessarily know what I'm measuring yet. I start an app. I need a "dashboard", but I don't know really know what's in there. And that honestly, what's in there is going to change overtime.

A lot of the way we think about Metabase is not so much creating a crisper set of artifacts people consume. But more creating a system, let's them poke and prod and just explore. And the BI use case falls out of that.

[00:03:44] JMS: There are these generations of BI tools. Like every 5 or 10 years, there's a new generation of BI tool. And as you said, you're not exactly a BI tool, but could you give me some of the context of the modern age that makes Metabase useful?

[00:04:06] SAS: Yeah. There are a couple pieces to it. One of them has been this last generation of BI tools has largely been set around the idea of decoupling from data warehousing. Back in the 90s, you'd buy a big data warehouse in Teradata or Oracle, wherever else. You'd have another equally big monolithic just business intelligence suite. Then came kind of the first wave of cloud BI with good data, clear story and kind of that generation where there was data warehouse. You threw bits up into the sky. You move things around. You had ingest. You had transformations. And then you also had visualization.

I think what happened with, I'd say, the rise of Red Shift or BigQuery is that you're not going to write a better data warehouse than one of them, or Snowflake. And so in general, there's been this decoupling. Every once in a while you see the pressures of kind of customer demand push people back into an in-memory data store or something else like that. But for the most part, there's been a strong decoupling. That's one piece of it where Metabase is just a data viz layer on top of whatever data you bring to it and we try to be fairly database agnostic.

The other piece of it is just this idea that software wants to be consumed in an open source fashion. And so I think for many people, they start their software search with GitHub and not with vendor calls. And I think that's becoming a bigger and bigger way that people both discover what to use and make choices. So I think one of the big pieces of Metabase has been we're trying to create something where if you're trying to set up just data infrastructure and you need the next step. It will be up and running in a few minutes. A lot of the kind of promise of Metabase is we're the laziest possible option, and that if you don't necessarily, again, know precisely what you're looking for quite yet, you could still set us up. We're not something that you add when everyone needs a tool. Where that tool you add before, there's really a tool needed.

[00:06:07] JMS: Okay. As you said, data warehouses are great these days. We've got the data warehouse technology that we need. And you're focused on the presentation layer. Let's say I plug in my data warehouse into Metabase. What happens?

[00:06:22] SAS: Oh, yes. Once you connect Metabase to your database or your data warehouse or even just a flat SQLite or H2 file, we will can do a pass through it and scan it. We'll try to guess what the various columns and tables represents. We will essentially create

this metadata system on top of your database and we will essentially – Well, depending on how you want to us, there's kind of infotainment mode where you just click around through our X-rays and we'll essentially generate dashboards based that we think you should look at in your database. You can use our kind of row browser, data browsing. We just like poke and prod at the various tables. Do quick aggregations or slices through them. And then if you want to actually build out dashboards or nightly reports or alerting, that's where you dig in and actually specify what you need.

So there's kind of these three different modes where there's just – I kind of want to like see what's in there and the others I want to explore. And the third is I actually have specific reports I need to build. And we kind of try to mix and match all three fluidly.

[00:07:28] JMS: So, give me like the day one experience. I plug Metabase into my data warehouse. What value am I extracting from this application on day one?

[00:07:40] SAS: I think from day one, you can say like minute 15. You have a way for other people in your company to look at the data and the database depending on who they are. They can either use SQL. They can use our non-SQL kind of GUI query builder. They can just explore by clicking around. And so within 5, or 10, or 15 minutes, you essentially have a window on to your data that anyone in your company could use.

In terms of day one, most people end up doing is they'll create a couple stock queries or questions as we call them and maybe a dashboard or two and pass those around. And I think one of the key things about Metabase is that we are designed for multiplayer mode. So it's not so much that I am landing and it's day one for me. It's more I'm landing and it's a day one for my team or my company or whatever institution I'm in. And it's the ability for people just to pass links around and to collectively explore through their data that makes us interesting.

And this is kind of – I think there's always the promise of this with all tools. And I'd say that like all tools, we've just taken it a little further. And for us, the way this manifests is that it's just very easy to create little ad hoc queries and ad hoc reports and that you can deep link them, pass them around Slack. You can save them into your own personal folder or kind of like the global

folders for everyone. In general, it's the, "Hey, look at this," kind of conversation we facilitate most.

[00:09:14] JMS: So it's like a discovery mechanism for finding interesting ideas within your data?

[00:09:21] SAS: I wouldn't even make it so high-minded as that. I think maybe one way to think about it is we're the dark matter of analytics. Where if we talk to people that have all kinds of ideas about what their data can do for them and they can monetize their data and they have like the always aspirational lofty goals. I think where we fit in is just the glue.

If you're trying to look at the rows in a table that have a timestamp that's still without a time zone, you can kind of do that. If you're looking at like – If you're trying to keep track of how far migration has run and whether it's complete or not. If you are looking at the last 10 signups and want to see if the IPs are normal, you can do that. If you want to dig in to whatever your application data from your app in your application database holds. Again, you can just kind of like do a quick sanity check. You can also do things like, "Huh. I wonder, of the people that signed up in this zip code, how many of them have actually done something on this site."

And they kind of mixes and matches this idea that you can browse through the actual data itself. You can create time series and like charting and all the things that usually define a BI product. And you can also do quick mapping and just – Again, it's the lightweight exploration that you then save as dashboards that defines Metabase.

[00:10:43] JMS: Metabase does allow you to have these different complexity levels of querying. Can you explain what that means? What are the different purposes of different types of querying in Metabase?

[00:10:55] SAS: I think the fundamental kind of query is that you're using our GUI to essentially behind-the-scenes construct a query. So there's maybe four different tick points in complexity. The most complex is you want to dive in a SQL and you want to use SQL templates and you want to essentially build fairly complicated queries for yourself or others. This is what people end up doing when they have a very, very specifically and they're trying to pull up as well as

where their data is not necessarily structured to make it easy for end users to slice and dice on their own.

There's kind of the slicing and dicing by end users where you know that you want to say look at your transactions table and see how many chargebacks happened. Then you want to say graph that by time and maybe filter that down to just the US. And within our GUI, there's also notebook mode, where if you want to construct fairly complicated queries using joins and calculations and kind of nested series of queries, you can.

And then what we call infotainment mode is when you're just using our X-rays. And so that's where you say, "I'm curious about this specific database. Can you just kind of blow it up for me and tell me what's in there?" That tends to be the more exploratory. Again, it's not necessarily things you'd use to run your business off of. But it does let you at least see some patterns in the underlying data.

[00:12:22] JMS: Take me through the life of a query in Metabase.

[00:12:26] SAS: So I'm going to use what we call the query builder as the starting point. So you'll potentially select what dataset you're interested in querying. For example, most people, it'd be a database table. So you'd pick the table. We reference tables using our metadata so we have an ID. You then – Behind-the-scenes you'd use our relatively fluid interface. But behind-the-scenes, we'd begin constructing aggregation clauses, filtering clauses as well as calculations and calculated columns and other that's behind-the-scenes. That would then take it back to the backend server where it gets expanded out. So the metadata gets injected. And then that is – Typically, that query will be in MBQL, which is our query language. That then gets sent to our query processor [inaudible 00:13:20] transpiler and we'll figure what the target database is. And essentially transpile that MBQL query into whatever dialect of SQL or non-SQL query language that database needs. It then gets sent over to the database it executes. It comes back. We do a little bit of just massaging of things as they come back and we'll analyze the results for things like trend lines or linear regression. And then we'll send it back down to the frontend client where it gets graphed or displayed in other ways.

[00:13:59] JMS: So is that to say that whatever query I issue to Metabase, there's a query translation layer that gets issued to the underlying database?

[00:14:07] SAS: Yeah. If you're using a completely vanilla SQL query, so you're doing a select one. That will bet passed through as is. We do have SQL templates, and those do get interpolated on the backend. And if you're using graphical querying tools, then you're constructing MBQL queries that then get, again, translated to whatever dialect we need, and then kicked down the database itself.

[00:14:36] JMS: And does that mean that – Well, you have to write query translations for all the different underlying data warehouse technologies and databases, right?

[00:14:46] SAS: Yeah. We have what are called database drivers, which if we're to do all over again, we'd name something else. They're effectively transpilers. So we'd go from MBQL, which is our lingua franca, and that's just what query is at rest are represented as and where they're constructed in by the frontend. So each database that we support, we will have to write a translation layer from MBQL to its dialect.

This typically – For SQL, we have a generic SQL database driver. But there's always something around time zones, casting, and various types that we have to dig into. And there's a fair amount of things we do to scan the database and just built up our metadata model that are fairly specific to each database.

[00:15:34] JMS: And tell me more about that. Like that sounds like a lot of overhead to have to maintain query translation or interfaces to all the different underlying databases.

[00:15:45] SAS: Yeah. I mean to have a graphical interface to any number of databases, you either have to speak their dialect or you have a single representation that then gets transpiled to the dialect. I think the only alternative would be just to have end-users write SQL themselves. And that is the approach used by a fair number of tools. But I think at some point, if you have non-technical end-users and you want to give them an interface to do so, you essentially have to go database by database on some level. So just concretely, MySQL has three different time

zone handling. The JDBC driver for MySQL has three different ways to deal with time zones and they're all a little nuanced and a little broken. Snowflake has their own kind of set of quirks.

So, in many ways, we are a great tool to flush out any database quirks of your SQL database. So there is just is a fair amount of like fairly low-level, fairly tedious work that we have to do to make sure that a given database driver is production grade.

[SPONSOR MESSAGE]

[00:17:00] JMS: Today's episode is brought to you by Logi Analytics, focused exclusively on helping software teams embed analytics in commercial and enterprise applications. A new innovation is called Logi Composer, which empowers developers to easily create customized and embedded analytic dashboards in their applications with complete control over the end user experience. It provides embedded self-service functionality can be tailored to match the skill level of your end-user. Logi Composer is also powered with smart data connectors to unlock data connectivity and query performance and a cloud-ready architecture for speed and scale.

If you're looking for an easy way to embed analytics in your application, download a free 14-day trial of Logi Composer at logianalytics.com/sedaily. That L-O-G-Ianalytics.com/sedaily.

Thank you to Logi Analytics for being a sponsor.

[INTERVIEW CONTINUED]

[00:17:57] JMS: Let’s go through some of the examples of the vocabulary in Metabase to help people understand what meta-base actually is. I actually like us to have some kind of working examples. Could you say like what is a typical company that has gotten a lot of value out of Metabase? And maybe you could describe from a high-level how it works and how it provides value to that company. Then we can dive into some of the vocabulary.

[00:18:24] SAS: Yeah. I mean, I'll use Gojek as an example. Gojek is – And I don’t want to butcher this, but I think they’re really – They’re the Uber of Southeast Asia. So they’ve been using us for a number of years. And I believe the initial use case was their customer-serviced

folks. And so they have ridesharing. They have a number of other services they offer. But essentially someone calls in. They have questions. The ability to look at that person's account. The ability to drill through all the information that Gojek has collected about that person and the stability to have standard reports across that use case is, I believe, what they started off doing. And these days, my understanding is they're using this companywide. There's about 8,000 people use Metabase day-to-day. Everyone from customer service reps to C-suite.

In terms of the value they get, I think it's a little tricky to define analytics and data access in like the value or the ROI perspective. I think the fundamental thing we do is make everything move more smoothly and just push data in context to all parts of the company. And so I think for most companies, a tool like us or our competitors is kind of necessary just to run the whole thing.

So you can kind of go use case by use case and describe value, but the best way to maybe orient folks is, at some point, the information your database has to come out again. And there is the application where you will present that back to users, but there's also is the internal use case where you have to like pull all these stuff and show it to people. And it's a little reductionist, but it does capture kind of the really widespread and general usage patterns that we have. So let's say that we're in most industries in most countries at this point. I don't know if this is helpful or not, but I'm happy to nudge me in the direction it's useful.

[00:20:29] JMS: Yeah. Let's continue with the Gojek example. There is something called a pulse in Metabase. Could you explain what a pulse is and maybe give an example of how that might be applicable to Gojek's usage?

[00:20:43] SAS: Yeah. A pulse is an emailed nightly report. It's meant to be this kind of lightweight that you get in the morning. You're in inbox and you see here are the people that signed up last week that never finished their shopping cart. We had to make something up for Gojek, but it's like the number of people in my territory that left one-star reviews. And so you get an email Monday morning. You take a look at the results of that, and then that gives you something to do or informs you as to how things are going in whatever area you're interested in.

If it's onboard customers, you could create yourself a pulse that essentially gives you a sense of the heartbeat of that and you'd say, "I want this daily, weekly, monthly." And it is meant to be

fairly lightweight. So it's not meant to be a dashboard, but it's not something you look at on your phone. And we find, I think, Gojek uses pulses very, very heavily. And I believe a lot of their workflows are handled by a pulse showing up with essentially a list of things to do.

For example, the previous example of a one-star reviews in my territory. You'd get that. You'd look at the CSV and then you'd reach out to all them potentially to understand what happened.

[00:22:03] JMS: And that's pretty useful compared to – I remember like probably 8, 10, 12 years ago when the status quo of these kind of nightly reports is like some Hadoop job. Runs nightly engines. Gives you a dump of data and then doesn't really tell you that much about what to do. I think with something like Metabase, you get a lot more fidelity into not only what has gone on over the last 24, or 48, week – 48 hours or a week. But gives you insights into what direction to go in.

[00:22:40] SAS: Yeah. And maybe like the most compelling thing I'd say from my perspective of Metabase as a user Metabase is just the ability to take the next click. And so if you are getting the results of Hadoop job at 7 AM and you had a follow-up question, well, that was too bad. You could ask someone on the data science or analytics team to build you a query. If that question or even asked before and there was something on a dashboard you could refresh, so you could go through that. But if it was a new question, you have to just wait your turn, or just not get an answer to it.

And a lot of the magic of Metabase as you start wiring it up inside the company, is that you get the pulse. You get the list of one-star reviews in your territory. You dig in and you can say, "I want to see this by time clustered in a certain power." You can slice it up by geography. You can maybe check in on given segments of that set of one-star reviews and potentially see are they related to a given driver. And all this just falls fluidly out of the pulse as supposed to being something that I'd have to go ask someone to do for you.

One way to think about Metabase is that we're automating that ad hoc query workflow, and that happens to most companies, where someone gets a dashboard and then there is like these subsequent questions. And those subsequent questions can be answered on your own with Metabase.

[00:24:10] JMS: Now, I remember we did a show with a company called outlier.ai. And Outlier does something similar to what we are trying both these pulses, where it gets an understanding of what data is being looked at and what kinds of queries are important to surface. And then it surfaces all kinds of information about that for you. I'm curious about what your technique is for surfacing what's useful. I mean, do you analyze the queries that are coming through the query layer and just kind of supervise that and then use that to inform what kinds of emails you should send to people in the morning? Tell me about what is the anatomy of a pulse.

[00:24:52] SAS: I think one of the most important things to point out is that pulses are made by humans. There isn't really any kind of automated generational pulses. I think, at some point, a human said, “I'm really curious about one-star reviews.” And so we’re not trying to infer user intent here. We’re just giving them a super lightweight tool such that anyone regardless of their sophistication can create a pulse on their own.

So the magic is not so much in us guessing what you’d want, as it is in letting anyone in your company do it themselves. Again, to take that second step themselves. And so the easiest way to think about Metabase is that we are a way for the 80% of your company that's a little intimidated by XL and pivot tables to ask your own questions and to, again, create their own dashboards or create their own pulses.

[00:25:45] JMS: There's also the ability to view visualizations and saved questions from Metabase. And I just like to get a better understanding for some of the other ways in which people use Metabase and what impact it has on an organization.

[00:26:02] SAS: One of my favorite examples of this was there was – I believe it’s 1800FLOWERS in Britain, and they’re the largest independent flower shop chain in Britain. They were using this on the factory floor to help figure out how to pack boxes of flowers. And one of the ways in which we were helpful to them is the folks on the factory floor started to see certain coupon codes over and over again. And so they sound the alarm and it turns out that people were fraudulently abusing one of their coupon codes. And so that was an example where someone who otherwise might not have had the ability to look up, “Oh, how many packages

have this coupon code in the moment?" And then realizing based on their contextual knowledge of their own workflow that this seemed off, and then bubbling up the chain.

And so I think there's a lot of benefits that come to an organization when people up and down the org chart have access to data on their own. And it lets you harvest kind of the creativity and just the awareness of their own work that folks like all through an authorization have as supposed to a world where only analysts write queries. And if an analyst didn't dream it up, no one is going to measure it.

It creates this decentralized, almost kind of sensor network where everyone just has access to the information relative into their jobs. And that infuses an organization. Makes it a little more efficient, a little more crisp, and it will catch mistakes like the coupon code abuse that I mentioned at the moment when it's happening as supposed to three months later when the fraud team takes a look at it. Again, if they've had classifiers or some sort of fraud detection mechanism, they could've set that up. But this is just the flower shop.

[00:27:52] JMS: I'd like to continue going through some of the features of Metabase. And then we can talk a little bit about the engineering under the hood. There's also this term called an X-ray. An X-ray is an automatic exploration of data. How does an X-ray work?

[00:28:08] SAS: Behind-the-scenes, we have table types or entity types, which are guesses as to what the rows in a given table represent. For example, we have table types of people, of the events, of transactions, of photos. And then each of the columns also has a metadata type attached to it.

For a given entity type, we have guesses as to how you want to explore it. And so there's an underlying YAML-based language that describes X-ray definitions for specific data types. And then we'll also try to guess which dimensions are interesting. And so all this is encoded up as a set of weights. And will do a couple passes through your metadata. Through the metadata for either the table or the column or this subset of data or even the cell of results that you're looking at. So you can X-ray either a table. You can X-ray a segment of a table. And you can also X-ray the specific cell of a query. And that will essentially expand all that out and it will look for metrics of interest or generate metrics of interest. It will also look for specifically high-value dimensions

that we discovered. And it will then just create a dashboard based on those metrics and dimensions. Again, we have a waiting – A set of waiting functions, and then a ranker that will pick the most interesting metrics and dimensions pairs and display them to you.

[00:29:43] JMS: And would be an example of when an X-ray would be useful?

[00:29:48] SAS: Typically, it's useful in the exploratory phase of either analytics or data science. So you get a new dataset that you started collecting and you just want to understand the shape of it. Which dimensions are interesting? Which metrics seem to be kind of correlated? What are the ranges of certain fields? Do you have like a normal distribution over some dimension? Or is it otherwise – Are things trending higher or lower over time? And the ability to iteratively just slice an X-ray lets you build up your own understanding of how things are shaped. Then based on that model of how things are shaping and then go in and do whatever work you're trying to do. But it's that initial orientation that x-rays are supposed to provide.

[00:30:40] JMS: Tell me a little bit more about Metabase as a monitoring platform. Because you've described it as something that provides kind of nightly updates and insights that are useful on an ad hoc, or I guess on a scheduled basis. But what about as an ad hoc monitoring platform for surfacing problems that might be going on across the infrastructure?

[00:31:06] SAS: I think once you move to infrastructure, it gets a little less useful. So we don't to real-time pulling of databases. And so, inherently, kind of the monitoring cadence recycle time that you'd have is on the order of hours and not milliseconds. And so if you're looking at infrastructural issues where the time granularity needed is like milliseconds. We don't work as well.

I think where we work really well is a modeling layer for the business. And so with that, the cycle times or cadences of interest are more daily. And so it's like what happened yesterday? Did anything strange – Or anything strange happened? Did users behave a certain way? Did our transactions falloff? Did more orders – Did fewer orders get completed yesterday than the previous three days?

And so when you have a kind of time cycle of days, we're very, very useful. I think as you compress the granularity down to kind of machine speed, just because we are largely federating queries down to data warehouse and waiting at them to come back, we're a little less useful. But people still definitely use us for that.

[00:32:25] JMS: Okay. Let's talk a little bit about the engineering. The deployment mechanism of Metabase, I could self-deploy it because it's open source. But tell me about how Metabase is typically used or deployed.

[00:32:41] SAS: I'd say at this, 99.999% of our users self-host. We've tried to make it very, very easy for companies to just spin us up. However they're in Docker or if they're JVM shop, however they run jars. We have some recipes for Heroku and AWS and some other places. And so that's the primary deployment model. So I think the usual pattern is, "Hey, this is kind of cool. I pull it down. I run a Docker image. Huh! This is really cool." You show it someone. And then at that point, people start being invited to the instance.

We are in like the worst kept secret ever in a dark beta for a host of Metabase offering. So we are actually starting to offer kind of a, "Hey, I want you to host it does for me," story for Metabase. But I'd say like even going forward, we expect self-hosting to be the primary way that people interact with Metabase.

[00:33:34] JMS: is there something that's been particularly difficult about creating a hosted version of the product?

[00:33:41] SAS: I mean, I think offering up hosted services at a consumer grade level is always really hard. And so I think that in our case, there is a lot of billing that we need to take care of. We have enterprise edition of Metabase. We also have support. I think one of the things that we've taken very, very seriously to the entire process is just security and privacy. So we have – The maintainable challenges in billing out a hosting product have revolved around just isolation. And so having there be a fairly hard barrier between different customers data. And then just kind of having the whole thing be largely self-healing and toaster-like. So we've always been a small team. Kind of way – Or punches way above our weight. And with hosting, we don't necessarily

want to just hire up a huge team to run servers. And so we've been automating things a lot, and it's just been the gradual processes of sanding things down and making them bulletproof.

[00:34:41] JMS: Did you use the word toaster-like?

[00:34:45] SAS: Yes. It's been one of our aspirational values on the engineering team, which is toasters just work. You put bread in. You press the button and you get a toast. And so one of the things we aspire in the systems that we build is just to have that be that straightforward. There are gazillion knobs. Things kind of just work. I think that's been one of the core strengths of Metabase, the product, where it is a pretty complicated product in a very complicated space. But we've tried to retain kind of the idea like there's a fundamental thing that you do, which is look at your data or chart it. And it's very easy to get there and it doesn't require a lot of fanfare.

[00:35:26] JMS: The server-side of meta-base is mostly written in Clojure. And at the beginning, you started off with Python and Jango. You switch to Clojure later on. Tell me about the decisions in programming languages.

[00:35:40] SAS: I think, in general, we've always had a pretty pragmatic engineering culture in the team. I think I actually wrote the Python version, I guess 6 or 7 years ago now while I was at Expa. And it was just something that I both wanted, and I think the companies that we were supporting got some value out of. And so that was just the language that I was most fluent in that time. So I kind of use what I knew.

And running it for a while, I think the main reason we switched over to something other than Python was the installation story. So I kind of mentioned earlier that we're trying to make it really easy for people to self-host. And the decision to move off with Python was largely driven by that, which is we wanted the ability to create a dirt simple, very robust installation and setup process, and we wanted to not have the all the pip hell that Python projects involve when people are running them.

If you're a Python developer, it's pretty easy to set up your dependencies. It's pretty easy to deal with like the inevitable weird croft that happens. You probably have a virtual environment story that's kind of muscle memory for you at this point. For folks that aren't heavy into the Python

world, it's kind of a pain. So our general footprint that we were looking for was it's one file or one image. It's up or down, and it comes with everything within it. And the other part was we wanted pretty robust database drivers. And so we poked around and ended up essentially wanting the database drivers and the threading model of the JVM.

We started off trying to port it to Scale and realized that because what we're doing with the database drivers, we are constantly going against the grain of what the Scala database ecosystem wanted to do. And so eventually switched gears and rewrote it in Clojure. That ended up being a great decision. I think we're all pretty happy with it at this point.

[00:37:42] JMS: Metabase can be run from a jar, can be run from a Java jar. I think that's because Clojure – Clojure gets trance piled into Java, right? Or compiled down to Java?

[00:37:55] SAS: Yup. Exactly.

[00:37:56] JMS: Yeah. So are there advantages, disadvantages to running on the JVM?

[00:38:03] SAS: Yes, very much so. I think the advantages are it's all pretty durable and it all – It's pretty well understood. The JVM has a lot of warts, but a lot of folks are very conversant in it. One of the biggest advantages was just we can build one binary. And there is literally a single jar, and that jar works wherever. And the fact it is a single file is very useful. And the fact that it's a single file is very useful, the fact that there is just single command and get things going is very useful. The ability to embed a database alongside it is very useful.

And the other thing, which I think is underappreciated is that the database drivers just have been banged on for a number of years. And I think that we, Metabase application, tends to be borderline abusive towards a JDBC support of a given driver. So that was one of the big components of all this, which is we wanted robust, mature drivers we could depend on. And so things – Like the node ecosystem at the time was still kind of coming up and we were a lot less certain. Oh, and just like the fact that there is very, very wide support for all kinds of databases with the JVM.

Downsides have largely kind of centered around memory consumption and spin up time. There a lot of constraints imposed on ourselves due to the fact that it takes about a minute to 90 seconds for the JVM and for all the Clojure classes to get pulled in. And so that has been a bit of a drag. And I think – Net-net is definitely been worth it. I think I don't believe that we would've been as successful as we were had we gone another route.

[00:39:44] JMS: Had there been any particularly difficult areas of engineering that would be worth us to exploring in more detail?

[00:39:55] SAS: We've done lots of things that are “hard”. And so there’s like ML algorithms that are through there. We have a compiler. I think the source of the greatest pain was really time zones. And it sounds very mundane. And I think there's a lot of folks that have kind of a heuristic towards how to deal with time zones. But when you have to support arbitrary databases with arbitrary implicit time zones running arbitrary versions of databases, database drivers, the time zone situation has been kind of nightmarish. I think that's been one of the longest running, just almost trench warfare that we've been engaged in.

And I think the sheer number of quirks between databases is staggering, and this has been especially problematic for us, because when you look at a chart and the numbers are wrong, then you lose faith in the tool. For us, there’re often time zones where it's not that there is just everything has shifted by a couple hours, but things like something revenue by day. It turns out, it matters a lot what your day definitions are.

Again, if folks look at a chart of something like revenue, which they understand very, very well and the numbers are off, then they’re like, “Why is this tool broken?” Ao things like having to dig into how timestamps are serialized and do a serialized within JDBC themselves. How they’re serialized, de-serialized on various databases? How is the wire? And then just having to just do the almost chain of custody for a given timestamp all across the layers of database, JDBC connection on the database side, the transmission, JDBC on our backend server. Whatever Clojure implicit time some conversions happen. And then also having it be pushed down to the client has been really, really fun.

[SPONSOR MESSAGE]

[00:42:02] JMS: Code doesn't always behave as you expect, and your team might be wasting time trying to understand why. You can empower your engineers with Rookout. Get the data that you need from live systems instantly and start shipping software better, faster and more reliably. You can visit rookout.com/sedaily today. That's rookout.com/se daily.

[INTERVIEW CONTINUED]

[00:42:32] JMS: It's been a while since I worked as a software engineer, but I think anybody who has worked in enterprise software for a long enough time has encountered difficulties with times zones. I think it's hard, because one of those things it doesn't exactly surface until it's too late if you ship time zone problems to production. I can understand how that will be really difficult. So I guess do write a lot of unit tests or integration tests around time zones to future proof yourself?

[00:43:08] SAS: Yes. As a project, we're pretty serious about our testing story. I think we've had a lot of different testing process, I say, like frameworks over the years. And so we're big into integration tests. And specifically with time zones, you can't really unit test it. Or you can, but it's not as useful, because so much of where bugs emerge is at the interfaces. And so we do a lot of end-to-end testing. We run our entire test suite on every database that we support. And we're constantly adding to our test suite. And as we discover new strange edge cases with time zone support specifically, we fix them. We wrap it all on tests and keep going.

I'd I don't remember off the top of my head exactly how many tests we have overall, but there's a fair mount pain. Yes. I have to get back to you as far as actual numbers, but yeah, we do test very, very aggressively. And it's been one of the few ways we've had to keep everything the same.

[00:44:09] JMS: What about the frontend? Tell me the architecture of the frontend.

[00:44:13] SAS: So it's a React and Redux JavaScript application. I think we have a couple interesting things going on there. One of them was we split off our kind of Metabase-specific craziness into its own JavaScript library that didn't really interact with Raact or Redux at all, but

it was meant to be a place where MBQL manipulation, parameter coercion and just things that are specific to Metabase and the Metabase source complexity lives, and where someone could dive in and start to use that without having to know much past, just vanilla JavaScript.

We have a fairly complicated single-page app. We use TCGS to visualize things. And so that's its own separate layer. And then in kind of the interaction layer is where the React and Redux code all lives. And so we've tried to split things up into layers that let someone dive in and be helpful and/or productive without having to learn all of technology stuff we use in the front end.

We spent a fair amount of time trying to make the frontend development experience really easy. And so both are Metabase lib, and then something that's called our entity loader framework, which lets you just do things like get me all the dashboards. And then the ability to use dashboards as objects that have methods has been very, very useful.

And so our actual kind of interaction code isn't littered with a lot of fetching or just, again, Metabase specific complexity. And for the most part, it seemed to work really well in entity loader specifically we like a lot. And so entity loaders really just do things like get me – For this specific type that I'm interested in, get me a filtered list of them, or give me this instance, or update this instance. I think that if we had doubled down on GraphQL years ago, we probably wouldn't be using them. But I think they are a very slick and very just – The ergonomics feel pretty good when you use them.

[00:46:21] JMS: When you look at the open source BI landscape, there is not only Metabase. The other thing that comes to mind is preset. It's a more recent open source B tool. Do you see this is a multi-BI tool market? I mean, there are lots of different applications that we would think of as BI. Do you think of this is a multiplayer market or do you expect the average company to really pick one BEI tool and go deep on it?

[00:46:53] SAS: A couple things. One of them is just that analytics has always been crowded. I don't think there's ever been a time when there was one tool for analytics. I joke that, yes, there are hundred competitors. And you'd found there'd be hundred competitors. In general, we don't presume that weird, anything resembling a greenfield world where we've created this industry

and we're the only player in it. Analytics was vibrant and had great solutions before we ever showed up or thought about it, and it deftly has other people doing great things now.

I think there's a cup – In terms of how companies use BI tool specifically, I think many companies use a lot of different ones. And I think they're not always couched as BI tools, but they kind of serve the same function. And so even if there is a centralized tool that people have paid for and decided as the mandated way to do analytics, there's a lot of other things floating around the company as well. And often especially as you get into larger companies, different teams use different tools. And so within one company, there can be 20 different BI tools that are being used across different divisions. So we definitely don't think that we're the only game in town, and there's definite a lot of really smart people doing a lot of really great things in this space.

[00:48:14] JMS: Have there been any kinds of queries or other types of bottlenecks that you've had to iron out from the presentation layer/querying layer to the underlying database getting actually queried against? Is there any particular bottlenecks you can point out where you've really had to figure out how to optimize it?

[00:48:39] SAS: There are a couple pieces to out optimization story. I think one of them has always been the app itself. And we you have definitely worked on making the app and the metadata system and the query generation system responsive. I think in terms of user-facing or user-perceived performance, the single biggest factor is the data warehouse query itself. Typically, everything Metabase does happens on the order of a couple hundred milliseconds. And then the query gets sent out and it can take anywhere from sub-second to tens of minutes. So when you look in a dashboard and it's slow, it's really unlikely to be "Metabase's fault".

And so there's a couple pieces to data warehousing performance. One of them is we try to just generate better, more optimized queries. And so we've definitely worked a lot on being smarter and how our nested queries interact with indexes. We tried pretty hard to work on casting appropriately date columns in a way that respects indexes on different databases. And so there's kind of the N database drivers all have their own performance optimization work.

The other thing is we do try to help our users construct better queries and better schemas, and often the most important determinant of performance is how well your schema matches your query patterns. And so things like pre-computation, things like changing the shape of data at rest are all the things you end up having to do to grind down performance from like a 10-minute query to a 10-second query. And those happen outside of Metabase proper, but they're still an important step in making users happy with us.

[00:50:28] JMS: You've worked in the data landscape for pretty long time. You've been founder and CTO it multiple different companies. How have you seen the data and analytics field change in the past few years?

[00:50:43] SAS: I mean, one of the most interesting kind of paths was the rise and fall of NoSQL Hadoop. I think for a couple years, their SQL was dead and everything was either longer. Hadoop is something else. I think SQL is definitely back and I think has cemented its place. I think that's been one of the most interesting shifts.

I think the degree to which data warehousing as a commodity is very interesting and has been a pretty tectonic shift. So the ability to go to Google [inaudible 00:51:16] credit card and potentially scale up to hundreds of terabytes without any drama is a pretty big deal. Did at Red Shift, did at Snowflake. I think the overall access to top of the food chain data warehouse technology that anyone has just by handing over a credit card is pretty incredible. I think the degree to which data warehousing is an impulse buy, where it does take you 20 seconds or so to provision in a Red Shift cluster. But it all happens pretty fast and it's all self-service. And that's where – Whereas in the battle days, it'd be like I'm going to set up a Hadoop cluster. I'm going to install software on the nodes. I'm going to the end like have a care and feeding this cluster. Nowadays, it's just like, "I just press a button and a Red Shift cluster pops up." That's been pretty incredible to see happen in my professional life.

[00:52:07] JMS: Why didn't the NoSQL era workout?

[00:52:13] SAS: I think it was always too many things under one tent. And so if you think about Mongo, I'd say Mongo did work out. But Mongo's primary place in my opinion was just that it

had an incredibly low cognitive impedance between a developer's mental model of how data structures worked and how to persist them on disk.

I think Hadoop was different story and you can kind of separate out the HDFS story from the MapReduce, and then Hive and Spark story. Arguably, it did work out, and Spark is still a really big deal. But I do think that SQL as a querying language is kind of all right. SQL isn't the greatest thing ever, but it's pretty solid. It's pretty easy to learn. There's a ton of resources for people to learn it. There is a ton of tools that speak it natively. I think, in general, a lot of us underappreciated the degree to which the SQL ecosystem had answers for a lot of the problems that we're solving and this ad hoc fashion in the Hadoop world.

I think the writing kind of was on the wall from the Impala and Presto days when it was just like, "Okay. Well, I guess what we're really doing is writing a storage layer for the a massively parallel SQL databases now.

[00:53:34] JMS: What are the biggest engineering challenges in the near future for Metabase? What are you focused on the most?

[00:53:40] SAS: I think performance, especially again with the data warehousing performance. Our ability to preemptively translate or transform data on behalf of users, the ability for us to better give them feedback and suggestions as to how to improve performance. We're trying to consistently trying to do more around our metadata ontology and our ability to move from infotainment mode on the automated X-rays to something that's actually useful day-to-day. We are trying to clean up our embedding story. And so having there be this crisper and more customizable embeds.

And then we're always just trying to get a better handle on the database driver world and both in terms of reliability and durability and, again, making it as much the toaster as possible. So there's a mixture of like the very boring and like day-to-day cleanup bugs, write better testing, as well as the more aspirational kind of taking X-rays to the next level that we're also excited to do.

[00:54:48] JMS: As a parting thought, if there's a company out there that's listening to this and they're thinking about using Metabase. Their curiosity is now piqued. Would you have any suggestions to kind of prod them over the edge to trying it out?

[00:55:04] SAS: I'd say it's very low-cost decision. You go to metabase.com [inaudible 00:55:11] Docker image. Kind of t each your database to just see if you like what's there. We've made it very easy for folks to download us and set us up. And I would encourage everyone to view it as just something to try out as supposed to this big, heavy's decision to make. And we generally found that once people see what access to data we provide, that they generally like us and keep us around. Yeah, go out, try it out. It's super easy to at least see what you do. And hopefully you'll get value from it.

[00:55:42] JMS: Okay. Well, Sameer, it's been really great talking to you about Metabase and quite a great modern data engineering product. So, congrats, and I look forward to falling in the future.

[00:55:53] SAS: Thank you. Much appreciated.

[END OF INTERVIEW]

[00:56:03] JMS: Today's episode of Software Engineering Daily is sponsored by Datadog, a monitoring platform for cloud scale infrastructure and applications. Datadog provides dashboarding, alerting, application performance monitoring and log management in one tightly integrated platform so you can get end-to-end visibility quickly, and it integrates seamlessly with AWS so you can start monitoring EC2, RDS, ECS and all of your other AWS services in minutes.

Visualize key metrics, set alerts to identify anomalies, and collaborate with your team to troubleshoot and fix issues fast. Try it yourself by starting a free 14-day trial today. Listeners of this podcast will also receive a free Datadog T-shirt. Go to softwareengineeringdaily.com/datadog to get that T-shirt. That's softwareengineeringdaily.com/datadog

[END]