EPISODE 1241

[INTRODUCTION]

[00:00:00] JM: A data warehouse is a data management system that contains large amounts of historical data and is used for business intelligence activities like analytics. A data warehouse centralizes customer data from multiple sources to be an organization's single source of truth. Getting the data from your data warehouse into the different applications used by your organization can be difficult. The company Census simplifies syncing your data warehouse with other applications. Census works on top of existing infrastructure and lets users pick destination apps, map the data and then handles maintaining live in-sync metrics. Because Census runs inside your data warehouse, data remains secure and is not stored on their servers.

In today's episode we speak with Boris Jabes, CEO of Census. He was previously a managing partner at Polynome and a senior director at LogMeIn. We discussed the complexity of data warehouses and how the field of data analytics has grown over the years and just how Census works.

[INTERVIEW]

[00:00:55] JM: Boris, welcome to the show.

[00:00:57] BJ: Hi. Thanks for having me.

[00:00:59] JM: You work on Census, and we'll get into what Census does, but first I want to talk a little bit about the state of data engineering and data platforms. And a lot of work in data engineering has started to consolidate around the data warehouse. And if you look back in history of how data engineering evolved, it didn't have to go this way. There were all kinds of things that could have happened. You had all these streaming systems. You had data lakes, which you still have. But why did we get to a point where so much of the work is being done by the data warehouse and so much consolidation is around the data warehouse?

[00:01:40] BJ: I think people, really, deep down we're craving a way to simplify, and I think the modern cloud data warehouse especially has really nailed – It doesn't truly solve every problem. There're still needs for all sorts of specialized technology. But for a lot of the general cases, it gives a single tool set that covers small companies, startups even that are just kind of scaling up, as well as massive sized companies. So the first thing I think that they're able to do is to say, "You don't have to have a breakage for various levels of scale and you don't have to invest up front," and we shouldn't underestimate how big that is.

And then the other reason I think people really started gravitating towards just do everything in the cloud data warehouse is it means learning like a lot less learning something bespoke. They are effectively –They're not perfectly standard, but they're a lot closer to standard SQL, and I think that's another thing people were really craving was a way to operate on their data without having to learn something proprietary or something kind of complex. And so there's still room for those kinds of technologies, but I think those are the two reasons I would say why a lot of the gravity now just resides on the cloud data warehouse.

[00:03:06] JM: And today, what's the typical path for data into the data warehouse? Is it directly off of applications or does the data typically go into Kafka and then goes into the data warehouse? Or does it go into a data lake and then to the data warehouse? How is the data warehouse aggregating its data?

[00:03:22] BJ: Yeah, it's a great question. I think I see a couple of patterns out there that are starting to become common. So the first is getting application data into a warehouse especially when it's born out of kind of let's call it a finance BI need. So a lot of the initial use cases can be just pulling data down from your sales tools and finance tools so that you can do better reporting. And for that I think, really, we've reached the point of maturity now for tools like Fivetran or Stitch such that people really just use these very, very easy ELT tools to drop that data into the warehouse. And you're basically at the point where you don't even need what I would call like a data engineer to own that piece.

And then on the product data side, or let's call it the kind of live sides of the business, I see two different solutions based on where you are in terms of performance and scale and latency needs at your company. I think there's actually a ton of people that do very casual 24-hour kind

of data dumps from their production databases into a warehouse, and they're getting away with that because they're comfortable with kind of that latency on the warehouse side, and it's easy. Again, it's a simple solution.

You do see people connecting their event streams to a warehouse as well. Again, the way I see it is not everyone's on Kafka, but you eventually graduate to something like that. A lot of the people who are not on that stack, it's because they're actually still able to do everything in Segment or Snowplow and just have those natively push the data into the warehouse, and that way you don't have to kind of invest in your own streaming architecture, but I think eventually you do. And so I would say, broadly speaking, the most mature organizations have some kind of streaming setup on a Kafka that their service architecture is depending on and then the warehouse is a sync off of that. And then all of the business applications are going directly to the warehouse. And then that combination becomes your source input for all your analysis.

[00:05:38] JM: What's the role of the data lake in this model?

[00:05:43] BJ: Oh, that's a good question. I think there's like so much data being generated. What's the old adage? We're doubling the amount of data we produce every – I don't know, 18 months I guess. It's our data Moore's Law. And the rate at which companies are generating data is similarly increasing. And so I think when you're dealing in the trillions of events and the amount of scale that your product team generates, you don't necessarily want to have that go directly into your kind of high-performance data warehouse. So I think the data lake is that staging area for immediately storing all the data you have, and that way you don't have to worry in any way shape or form about the scale at which it operates and you can make decisions later on as to which parts of that data you want to bring up for analysis.

I do think the warehouses in the data lakes are trying to blend these two universes, right? You're seeing a lot of those kinds of – That kind of terminology emerge. I don't know where it's going to land honestly. But I think, again, the desire for simplicity is there. I think the way an engineer or an analyst wants to deal with data is actually in one consistent way to access it and then with just differing levels of latency based on their needs. So I think as an end user I don't really want to have to think about is the data in the data lake or is the data in a data warehouse? I want to be able to just do an analysis. And then I'd love to be able just to specify, "I'm okay with this

analysis being 24 hours behind or take like one hour to complete," versus, "I need this to be fresher data. I need this to be a faster query." And then a single system could define where data should live based on that. That's kind of an ideal outcome I think.

[00:07:36] JM: Yeah. I mean, I did an interview with somebody from Snowflake like three years ago or so, and I remember talking to him and it sounded like – I mean, Snowflake is proprietary technology. So nobody really knows what's going on under the hood. At least not anybody that is outside of Snowflake or who hasn't worked there, but it sounded like from what he described it's this tiered system where it is basically at the lowest tier something approaching a data lake where you're throwing all your data in there and it's just getting tiered into lower down the memory hierarchy or whatever and into slower and slower access storage mediums. But it sounded like the long-term goal of Snowflake is to be that unified solution, much like you have the same thing going on with the Databricks world, the Databricks Delta.

[00:08:28] BJ: Yeah. I think they're both coming at it from their own side. And let's even like step back on the Snowflake side, right? Even just separating storage from compute, we take that for granted now. It was the great innovation, and props to BigQuery for having that as well pretty early on. But once you do that, you've already created that first level of separation between thinking about how your data is stored and how performant you need that access from the computation. But I think the true blend will be specifically not just what kind of data warehouse is going to do that computation, but how easily accessible is that data. And it's hard to tell, you're right, at Snowflake, whether they're doing – Are they kind of changing how they store the data based on your access patterns? But if I were them, I would be working on that. I think that's definitely something. If I were an engineer there, that's what I'd want to pull off.

[00:09:24] JM: That's what it sounded like, and that was what kind of blew me away. It was one of the more mind-altering experiences that I've had on the podcast where I really started to understand, "Oh, this is a real company." Like they've got some real problems that they are solving here, because making that kind of tiered thing just invisible to the user is quite a challenge.

[00:09:43] BJ: It's a huge challenge. And by the way, I think it's something we've talked about, and we can get into this later. But we've talked about this idea of in our product like a magic

scheduler, because customers always want all these various schedules. There's like any ETL type product, you're going to have effectively people wanting to redefine cron. And from the day the company started, we were kind of trying to think about how do we get people away from the concept of schedules altogether and just try to – What if we could have a magic schedule? An adaptive schedule that determines when data should move based on prior patterns? And it is fantastically difficult right to do that well and do that in such a way that where everyone's happy all the time. Yeah, I think you're right. These are capital E hard engineering problems.

[00:10:33] JM: And so you're saying in your description of data warehouse versus data lake, it sounded like you were kind of saying companies that don't produce really, really high-transaction volumes. They don't need a data lake. You can just throw everything in the data warehouse. Is that the status quo there?

[00:10:49] BJ: That's how I think. Yeah. I think a lot of people, I mean, we work with customers of a decent scale and they're able to operate in that way. I think the interesting question is a lot of the people that I work with that have the two pieces of infrastructure separate, and oftentimes all in AWS. I can't tell if it's provably because they've run the numbers and this actually works out to be significantly better for them, or if it's just a remnant of, “Well, they started working on that system five years ago and you don't just throw it away.” I think most people who are building their infrastructure now can to a pretty high-degree just rely on the data warehouse exclusively, which I think is part of the reason why people are so excited.

[00:11:39] JM: Gotcha. So let's start to get into what you're building with Census. So I’ve done a few shows with Fivetran, and I think of as a really easy way to move data between different places. Like get your data out of Stripe and put it into your data warehouse for example, or get your data out of Segment and put it into your data warehouse. And naively, when I look at Census, I kind of think of it as the opposite direction. Like you've got your data in your data warehouse and you're looking to get it out into something else like a Segment or a Zendesk. Is that a good way of looking at it?

[00:12:21] BJ: I think from a pure in which direction are the bits flowing? I think that's a very reasonable way to describe it and that's how our customers have kind of described it since we started in 2018. They started calling it like reverse Fivetran back then. I think what it doesn't

describe is what people are doing, right? I think when you think of Fivetran, you're pulling data down into your database so that you can do analysis. But when people use Census, they're pushing data out in order to drive a business workflow. I think that's actually a very different job to be done. So even though, yes, bits are moving and they're moving in the opposite direction, the needs of the users and what they're trying to accomplish is actually quite different.

[00:13:06] JM: Can you say more about that?

[00:13:07] BJ: Sure. When you think of ETL or ELT as the term goes nowadays, you're bringing data down from Stripe, from NetSuite, from your Zendesk and you're trying to analyze you know how much revenue did we generate in this quarter based on all these various signals so that we can get the one correct answer so that the CFO doesn't get in trouble when we present to Wall Street? With Census what you're doing is you're on the other end of the – Let's call it the data refining process. So users are generating really interesting models and insights about users, about customers, about entities in their business, and then what they want to do is use that to change the way someone operates in the company. So let's take your example of Zendesk. Everyone uses Zendesk to handle tickets from customers, and one of the things that every company is trying to get better at is provide not only better service, but better service per agent so that they can cover more users without loss of customer support experience. And the data team can have really useful information about what users are doing to the point we just described. So data analysts might compute how much revenue they're generating from one company and combine that with how large the company is and how active they are in the product and determine this is a VIP customer. And by pushing that data out to Zendesk using Census, the support team can now take advantage of that when they're prioritizing tickets and when they're responding to customers. And so it's about making the rest of the business operate better using data rather than enabling analysis. I think that's the real difference here between what people are doing, and which changes the needs in terms of latency, in terms of correctness.

[00:15:01] JM: Can you go through another workflow example?

[00:15:05] BJ: Totally.

[00:15:06] JM: How people use Census.

[00:15:07] BJ: Totally. And there's really you know dozens. I'll give you another classic example. This is a really, really common scenario. We work with a lot of freemium high-scale b2b SaaS companies. These are companies like Figma or Notion, and they have millions of users and many of them have the product for free. Some of them are on like a pro version and then there's an enterprise version. And you can't call everyone. That's not only is it not cost effective? It's not the point. The whole point is to do scaled-out software. But when you're dealing with the go to market and the sales funnel for companies like that, you really want to identify great leads out of your user base. And the team that is most able to do that is the data team. They have every single signal landing in the data warehouse. They can combine that with external signals that you have about companies like the kind of company that the person works at. And then you can combine that with a regression analysis or just something more advanced on the data science side to determine what kinds of users, what kinds of companies in the user base are most likely to convert to a paid plan or to an enterprise plan? And then when you have that analysis done, instead of having a meeting and say, "Hey, here's some users I identified in my analysis," you plug that in directly into your CRM and generate the leads automatically. Census will not just sync data. It will create objects in the CRM and then the sales team can have that automatically routed to whoever's in charge and they can make calls. They save time having to do this kind of manual process between them and the data team and you get higher quality leads because the data team is uniquely positioned to have the best information to do this.

[00:17:09] JM: In that example, can you give more of a description of what the data team is actually doing? What the data team is building? Like I've got my – There's like the sales team with the CRM, and then there's like the data warehouse that's ingesting lots of events, and then the data team is responsible for building up the middleware using Census to sync data from data warehouse into the CRM. What is the data team doing?

[00:17:35] BJ: Yeah. I think they're doing a bunch of different things, right? All of which I put under the umbrella of productizing their data. The first thing is they have to model the data correctly, and I think of that as coming up with what are the unique entities that matter in the business. So potentially that's users. Potentially that's companies. It could be other kinds of

objects too. It could be a unification of invoice data for example. And then once they have those unified models, they have to start – They structure them such that they can get all of the columns that you need to drive a business workflow, and don't have to solve that all in one giant table, but we're increasingly I think seeing that as a pattern of let me create out of my entire kind of data transformation pipeline. Let me finish with a table that represents here are all the users that we care about.

And then on that table you're going to affix columns around how active they are. What are the various members of that workspace? How many workspaces they have? And then you might have interesting scoring attributes. In that example about lead qualification, you're going to run a query or it could be the output of an internal ML process too, but let's keep it simple. And so you might have a query that uses as signals. Like I said, how big a company is? How much they've been using the product? Have they hit this event or that event? And then combine that into a score, make that a column, and then create a mapping from users that have a certain score into the CRM such that that generates leads. And the rest is kind of schedule, hit, go, and then maintain the pipeline.

And so in that process they've gone from kind of design and coding part of this, which is really where they should spend most of their time and how they model users. Should two users that have the same email be part of the same company? If you have a workspace with 10 different emails of different like users like that don't all share the same domain, who should we attribute this workspace to? There's a lot of hard work to be done there. How do I stitch anonymous user data with identified user data? All of that is work that the data team has to do. And then when they're ready to deploy the results of their analysis into a live workflow, that's where Census comes in, and they can be sure that the data is valid before it goes out, because the last thing you want to do is push invalid data into production systems because the hit, if you screw this up, is much worse than if you screw up a dashboard. And then have a tool like Census there to monitor how it's going. Make sure that when the sales team decides they're changing their data models that your pipelines break and everyone can kind of adapt. And that's kind of a day in the life.

[00:20:34] JM: And the alternative to this would be that the sales team is literally periodically going to the data team and asking for like a report and then they have to like manually create the leads in the CRM?

[00:20:50] BJ: There are only two alternatives I think to our world. One is you get your engineering team to build these pipelines, and we're back into your original question about what's happening in data engineering, and these are non-trivial pipelines to build because all the usual reasons. The APIs change. And if the pipeline breaks, you're breaking a live workflow. You're not breaking a reporting system. You're breaking a live workflow. You might have live emails go out based on this data. It's really, really important to have these pipelines be up to date and correct.

And the other alternative is your sales team goes to your BI tool. They'll go to Looker or some tool like Mode and download a CSV and upload it into their CRM. This happens all the time. And that comes with all sorts of problems. One, it's not incremental. It doesn't have any kind of ability to iterate. It's just a random report they're going to download and then it creates a whole bunch of duplicates because a manual upload isn't really aware of what's happened and the data team is disconnected. That's really the most frustrating part of what people are doing here, is that they're moving data kind of independently. And the point of Census is to say, “Not only should we make your data warehouse an operational hub. Not only should we make your data warehouse be the source of truth for these systems,” but you want all of these pipelines kind of live connected to the source so that the people responsible for generating that information are aware when like where they're used in the leaf nodes of the business.

[00:22:27] JM: And so let's talk a little bit more about it from your point of view. When you're building infrastructure to facilitate this kind of export from data warehouse to CRM, what's actually going on in your infrastructure? What have you built?

[00:22:49] BJ: Yeah. There's a lot going on. Let's see if I can summarize it. The first thing is we connect to our customers data warehouses. We have a credential to the data warehouse just like a BI tool. And that credential will execute a query. It will execute the query that says here are the – Let's call it sales qualified users, and it will then do a comparison with the last time data was synced to determine what has changed. Here's a query that is like all the sales

qualified users and we don't want to resync the 100,000 we synced last week. Let's only sync the new 10,000 that occurred. That's the first piece of interesting code in our product is like generating that diff, and the way we do that is really nice. We use our customer's data warehouse to do that. None of the data has to leave our customers premises for this. And then from the resulting data set we have to prepare that for upload into the CRM. That involves unloading it from the warehouse and stitching it together such that we can batch it into the destination system. You're probably aware that every single SaaS product is different. They have different kinds of APIs, different kinds of rate limits and different kinds of batch APIs, which is what we try to use exclusively. And so for certain products that means we have to take the data, determine what fits in. Whether that's a 10 megabyte file batch or a number of record batch and then kind of have the job run so that it can bring that data all into the destination system, extract from the destination system everything that went wrong. Again, this is a distributed system and these third-party apps are fairly unreliable. They fail in a number of different ways. Some of which the user can't do anything about. Some of which is in the user's control. A system might reject an update for having a field that's too long, stuff like that. So we bring all those errors back and record kind of how the sync was run. Like just kind of what was the final tally of records that were created, updated and rejected, and all of that kind of comes back into your warehouse and then we go quiet again and until we run the next time. I think that's probably in a nutshell what Census is doing here.

And the interesting thing that we've started to add into that is if you think of Census as this last mile, we're basically pushing data out to these systems of action. As the last mile, like this is really where the data has to be correct. And a lot of what we've seen with our customers over the last few years is the more they use Census, the more they are empowered, but also scared that they're going to break systems. And so we've started adding into Census built-in validation so that you're at least some level confident that your data on the way out is not malformed and is not going to break things.

A good example of that, I'll give you a really classic, simple thing that Census can always validate because we understand the destination systems, is something like duplicate records. You might be able to sync into something like Salesforce duplicates, and Salesforce will, believe it or not, silently accept the duplicates and you won't know which one was picked as the correct answer, which is not a great experience for the user. And so those are validations that we do up

front in Census. Every time Census runs, it computes a delta, it computes which records are valid and rejects upfront records that are invalid based on the destination rules. And then it batches the data so that it can push it into a destination system at maximum speed and then brings you back all of the errors that occurred. Oh, there's also a metadata check obviously. Metadata breaks. Your columns break. Your CRM field names change or field API names change or they get deleted. And so every time of course there's also a metadata check to make sure that these pipelines can even still execute. It's a lot of work to maintain high-quality data pipelines across kind of let's say systems that are changing as well as unreliable.

[00:27:36] JM: And if they break, do you just notify the engineer somehow?

[00:27:39] BJ: Yeah, we do two things. One is I treat this like a monitoring problem. A lot of how Census is designed and a lot of I think what you see in how our product is thought out is really applying the lessons that you and I know from the world of software engineering and DevOps into the world of data. Yeah, it first and foremost will stop the sync. Send out an email to any number of users or a logging system that you might have internally and tell you what to do. Tells you, “Sync can't run. A mapped column was deleted.” Something like that. And then in addition to that Census – And this is actually pretty fundamental, and a lot of non-technical people at first are confused by this, but it's actually I think the right way to design these kinds of systems. Census is state-based. It's not an event-based system. So you're pointing census at a data model of – What did I call it earlier? I said let's call it sales qualified users. That model is not sales qualified users like events, like, “Oh, here's a new user. Here's a new user.” It's a stable table that defines or query that defines what users out of the entire user base are qualified. And that way if there's a failure, Census will, no matter what, rerun. And when it reruns, for all you know, even if you didn't do anything as an engineer to fix it, maybe the problem was legitimately transient, then Census will heal. When the next run occurs, everything that was rejected, everything that failed last time is automatically rerun. And so I think that is where people get a lot of benefit relative to hand-rolled solutions. It's all of this kind of tracking of the failures and retrying of the failures, because this is going to be eventually consistent.

[00:29:40] JM: Zooming out a little bit. Is one way to look at this like a Zapier for your data warehouse? Because you think about all the N-by-N problems to be solved and all the different integrations to write, do you think the analogy to Zapier is appropriate?

[00:29:59] BJ: It absolutely is actually. I think the reason it's appropriate is if you think about people use tools like that in lots of different ways. They're very broad kind of user base. But it's a lot of operationalizing and automating some business workflow. And the way Census designed is to solve those same problems, but instead of using a point-to-point, N-by-N as you said, integration model and an event-based integration model where, "When something happens here, do something there."

Census is designed around this idea that build a unified correct data model. So think of it more like functional programming. And tell us where it should be put and we will make sure it arrives there. And that way you don't have to think about event replay. You don't have to think about, "Well, what action triggered that outcome?" You can define the query. You can write the functional program that says, "Hey, what inputs lead to what outputs?" So what rules define whether a user is sales qualified? And if those rules change, then great. Then all of the old users that were not qualified before will now become qualified and Census will sync them, which is something that is extremely difficult, if not impossible to do in these event-based kind of connector tools.

[00:31:30] JM: I see. Can you talk a little bit more about how you keep up with all the integrations? I guess I'm a little unclear about that because you have so many of them.

[00:31:40] BJ: Yeah. That is a lion's share of what the team does obviously, and my team has years of experience with this. This is not our first experience building these kinds of integrations. Today I think Census has over 30, like probably nearing 35, and that's going to grow to probably 100 this year. And so to build those, there're two things. First, the beauty of census is one side of the equation is fairly fixed. We support five or six data warehouses, not N-by-N. So we don't have to support every combination of input to every combination of output. We define the input as this hub. Like the data warehouse is the hub. And our goal is to sync data out from a warehouse. So it's Redshift, and Snowflake, and Databricks. These are our bread and butter.

And on the output side, first of all, some of these integrations are very much follow a power law in terms of like popularity. That's one thing. The top 10 of our integrations get a lot of play, and they're the most robust and they actually have been kind of battle tested in production for years.

And then the long tail, we just have a lot of internal systems to make buildings integrations effective. There's, first of all, like internal generic components that we can reuse for any destination system. So things around retries, things around rate limits, all of that luckily is shared logic in Census. When a new service comes out, and we just added one the other day that has unbelievably low rate limit. Instead of having to adapt to their weird amount, like this is all just plugs into our existing infrastructure. That's one. There's there are things that we can reuse across every integration, which is really nice.

The second thing, there's a lot of institutional experience, so the batch APIs. We're at the point now where for most of our integrations we're actually giving them the feedback on how they need to fix them because most companies really implement poor batch APIs, which is something that Census really thrives on, is to try to move these SaaS applications to take data in batch. And then the last thing that really helps is number of years in production. Like one of the unfortunate things about this space is unlike – To rewind to what you asked at the beginning. Unlike – Let's call it Fivetran, where you're bringing in raw data from a Salesforce instance down into your warehouse. Census is dealing with bespoke user models into potentially very bespoke objects in a CRM or in a support tool or a marketing tool or any of those things. And so a lot of this just means you just need years of customer use to see all the patterns that can emerge. There are just error messages that you would never be able to simulate on your site is the really honest answer. I refuse to believe that there's anyone who could generate every single way that – Let's call it a third-party Marketo instance could fail. You do as much testing as you can internally and then the rest is being in production for a few years.

[00:34:56] JM: Are there any big variations between the data warehouses like when you interact with the various data warehouses like Snowflake versus Redshift versus BigQuery?

[00:35:11] BJ: Yeah, there's definitely some variants. They're not as – I mean, as any customer who's tried to port from one to the other will tell you these are not turnkey migrations. Let's call it the SQL is 80% the same, but given the ways in which we twist SQL here to do things, it is actually different on all of our warehouses including, let me be clear, between Postgres and Redshift. Those are also not the same despite the fact that Redshift will tell you it is. Yeah, I would say all of our integrations with warehouses are unique. I'll give you – What's a good example? The way you unload data in Databricks is very different than the way you unload data

in Snowflake. And then the formats in which they dump data, even though it's CSV, there can be some mild differences between them. And then the limits, we are generating kind of – Because we're – Remember I told you earlier we do this delta of what records changed between each run of our system. So we do that by doing – It's actually a kind of a SQL accept query in your warehouse, and that's a very long generated query that's doing it against a snapshot that has a GUID in your warehouse, and those don't always just work between two warehouses. There's some customization there between how it works in, let's say Snowflake versus BigQuery. And then finally authentication is slightly different, and permissions are slightly different on every system. Yeah, I would say the warehouse connectors are not the same.

[00:36:50] JM: I'd like to zoom out a little bit and talk about some other subjects in the engineering landscape. First, DBT, we've done some shows on DBT, but maybe you could briefly talk about – just give an overview for what DBT is and why has DBT had such an impact on the world of data engineering, and particularly your business.

[00:37:12] BJ: Absolutely. It's without a doubt kind of the most significant change in the data landscape over the last year or two. I think I told you this earlier. The way I see everything that we do and most of the things that are happening in the realm of data as the application of software engineering and DevOps principles into the realm of data. You can think about that at every stage. Census almost acts in a way as a CI/CD system. It is a deployment mechanism for your data to get out to where it's used. And DBT really created two pretty huge, yet simple, but huge innovations. The first thing is it gave people a macro language around SQL to reference two different SQL queries so that you can start to decompose your transform logic into a series of SQL queries and data models. And the beauty of this is they're connected via the logic, not via some ordering of events. So it's not like, "Do this query. Then do this query." It's, "This query depends on that query." You get a distributed acyclic graph of dependencies and that way you get a separation of concerns in the way you do your data modeling.

DBT gives users that, which is a huge improvement over just writing a SQL query, which is what most analysts were doing. And then, second, it allows you to use Git to kind of store and version that code. And that's kind of the baseline for doing all of engineering. It allows you to treat your SQL analyses and your SQL data models as a piece of versionable code, and that was a huge change for people in the data world. Now you can have versioned data models that can be

tested, that can be code reviewed by others, that you can put them in a pull request, you can have different branches, all the things that you and I take for granted. And then Census actually natively connects to your DBT repository so that you can pull out all of that information, all those data models directly out of your DBT repository and then you can decide which DBT models, which branch, which version of a model is the one you want to sync, and then Census acts as, like I said, kind of a CI/CD and deployment system. So it will validate that data, push that data into a destination system and report back to you with basic monitoring of how that's going.

The existence of DBT has really changed our business mostly because it finally gave people the building block they needed to start treating their data not as a one-off. I think, really, everything we're trying to do here as a company is about helping data practitioners, data analyst, data teams treat their job and their output as a product so that they're not just reacting and doing, “Oh, let me write another query for you. Let me fix this report for you.” But actually saying, “How do we start to get leverage out of our data investment? We spent all this money on the infrastructure, but how do I leverage up kind of all of the analyses that we're doing?” And that is what DBT has really made possible here. And Census, as soon as we kind of – I guess we came across DBT a few years ago, and as soon as it was possible to start integrating with it natively, we did.

[00:40:46] JM: Great analysis of it. Another category is the workflow scheduling tool, notably, Airflow. And then you have up and comers, Dagster and Prefect. What are the roles of the workflow scheduler and why is Airflow insufficient? Why are there competitors to Airflow?

[00:41:09] BJ: I think you're going to start to notice a pattern in my responses. If you, once again, go back to our trusty handbook of software engineering, then we had code and versioned code for decades before we had Terraform. You can think of DBT as really having laid the groundwork to give you versioning for the code, which is the data models. And what Airflow and Dagster and Prefect and all these orchestration tools are doing are saying, “How do I bring the logic of how all this moves around and the state that you intend to exist across all your systems into code as well?” And Airflow was born before DBT, right? Of course Airflow is also solving a lot of the problems that DBT was solving around the dependency graph of SQL queries. People were doing that as a series of Airflow jobs, which some of that has been simplified. But at the end of the day you know you are eventually dealing with heterogeneous systems even inside

your warehouse. You've got maybe an ML process happening with a totally different tool set. You've got some SQL happening in DBT. You might even have sufficiently large team that there're different teams doing that stuff. And there're higher level dependencies that you need to orchestrate. That's what I think those tools are there for.

To your question about why do we still see innovation there and why is Airflow not enough. I think that honestly comes down to the audience for these tools is bit by bit, and I think we're really at the beginning of a 10-year kind of journey here becoming more sophisticated, and I think they're becoming frustrated with tools that are not as great as they can be in every category. They got fed up with doing SQL in a worksheet and they moved to DBT. And I think even the people who are using Airflow are frustrated with some of the limitations in our – Or the user experience, and they're trying to find something better. And so that's my high-level answer for you as to why there's innovation in that space. I think deep down this audience is just learning about these techniques, and as they become sophisticated they want better and better tools.

[00:43:30] JM: Predictions for how data engineering is going to change in the next five, ten years.

[00:43:40] BJ: I think the first self-serving answer I think is a lot of the simple-ish tasks in data engineering will become automated using technology like Census. If all you're doing as a data engineer is connecting Salesforce down into Snowflake, like that's no longer a good use of your time when there's tools like Fivetran around. And vice versa, if you're there just to try to create a pipeline that you maintain between Snowflake and Salesforce the other direction, that's also not a great use of your time, and there're products like Census to take care of that.

That said, I think the next five to ten years for data engineering is this push towards productization. I think of traditional data team was actually split. It would have BI under the CFO and they would be given effectively a warehouse with stuff in it and they would do analyses and present it in Tableau or something. And the data engineering, insofar as you had those people on staff, lived under the engineering org. And I hope they're not living under a CFO, but it has happened based on some companies that I've talked to, which is unfortunate. The new data team I think is reorganizing itself to be, A, like its own entity in the company. I think that's the first

significant change. I think you're going to start to see more and more this idea of a chief data officer arise. And the reason for that is if data is not just an afterthought or like exhaust system. It's not an afterthought. It's probably more of a backwards looking team for a lot of people. It's like, “Hey, what happened last quarter? What happened last year? And can we run a regression on users?” and those kinds of things. But first, as AI and ML took over on the product side, and then now with tools like Census, people are productizing their analysis for everything in the business, whether that's marketing, sales, support finance. All of these organizations are being enhanced using live data. And so what you're going to see I think is a restructuring of the data team to first of all merge these various practitioners. There's going to be data engineers, analytics engineers, data scientists all living under one house. And the data engineering staff I think is going to become increasingly in charge of really high-scale production data systems. I don't think – let's say you have – To go back our way at the beginning, you have your Kafka set up for connecting various services within your live systems, like what users directly experience on your product side. That's going to be a significant part of what the data engineering team take care of. This is the pub/sub infrastructure to make sure that your Elasticsearch servers get every single update as quickly as possible from the other side of the business. That's I think where a lot of data engineering focus is going to be and potentially purchasing and owning core, core infrastructure, so buying Confluent, buying Snowflake and maintaining those systems and the permissions around those systems and all that kind of stuff.

Then you're going to have this middle layer of, let's call it analytics engineers. This is the kind of bread and butter of what DBT has championed as well as Census. These are people who have historically not been engineers, but are kind of hopefully it's becoming more evident that they are becoming like engineers. And their job is going to be to create the central models for the whole company. These are, “There's a whole transformation pipeline here to try to build these unified. What are the users we care about? What are the companies we care about?” What is a deduplicated user? What is a company? Is it an email domain? Is it actually 12 email domains combined?” all these things that you have to do, and then tie every metric to them.

And then you'll have the data science team that is there to kind of do ad-hoc experimentation, forward-deployed analysts that are working with a specific team to kind of work on a process, like you might deploy an analyst out into marketing for a project. And so to me the real shift over the next five to ten years is this kind of unification of everything that a data team does under one

umbrella, and it will likely be viewed similar to kind of a product or a technical team rather than a BI team.

[00:48:14] JM: It kind of sounds like in your vision there's a lot more stuff that can be bought or added on to enterprise products like Confluent or Elasticsearch or whatever that are going to save the time that currently is allocated to data engineers today.

[00:48:31] BJ: Yeah. I think the way I see it is as a business and as a human, you want to try to work on what is differentiated and unique to you. I think there will always be some data engineering work that is hard to buy, and some of that will just be like, “Well, it's really we have a weird database that none of the vendors want to support. So we do it ourselves.” Obviously, that's a whole chunk of the world right there. But for core things, I think you're not going to build your own Snowflake and you're not going to build your own Kafka. There's very little reason not to buy off the shelf there. That said, as an engineer, you will still find lots of ways to add I think scaled out value, and the emphasis should be on what is unique to your business. Same for an analyst, right? Like the data engineer and the data analysts are both trying to find unique insights in their business. There's still a long way to go. I think just connecting production systems will still be a lot of work for data engineers. And look at how much work they're still involved in deploying an AI model directly into a product, right? It's still kind of a mess. I think just running on the product side, there's a lot of work for data engineers. I haven't looked at the stats, but I think the number of job titles with the word data engineer is going up, and that's a good sign that I think everyone is trying to invest in this.

[00:49:54] JM: All right. Well, Boris, it’s been a pleasure talking to you. It seems like a good place to wrap up. And thanks for coming on the show.

[00:50:00] BJ: Thanks for having me.

[END]

