SACRA

EXPERT INTERVIEW

UPDATED
09/12/2022

Julia Schottenstein, Product Manager at dbt Labs, on the business model of open source

TEAM

Jan-Erik Asplund
Co-Founder
jan@sacra.com

Published on **Sep 12th, 2022**

Julia Schottenstein, Product Manager at dbt Labs, on the business model of open source

By **Walter Chen**

EXPERT INTERVIEW

Julia Schottenstein

Product Manager
dbt Labs

Background

Julia Schottenstein is a product manager at dbt Labs. We talked to Julia to get a better understanding of dbt's open source business model, the modern data stack thesis, the competitive dynamics between dbt and companies like Snowflake and Fivetran, and dbt Labs' big upside vision around data transformation.

Interview

Tell us a bit about your background, especially in venture. What inspired you to join dbt Labs?

I spent a number of years at a venture capital firm called NEA investing in open source, dev tools, data, and infrastructure companies. I first discovered dbt in early 2020, made by what was then a data consultancy called Fishtown Analytics. I flew

to Philly, got to know Tristan (Handy) a little better, and thought, "This is really interesting."
dbt was at the intersection of a lot of big trends that I was seeing – more data moving to cloud warehouses, empowering a different persona to do more software engineering-like work, and a whole ecosystem being built around data assets.

I had a lot of conviction in what the team at dbt Labs (then called Fishtown Analytics) was building, and at first, I wanted to get involved as an investor. I had invested in a lot of open source businesses, and I saw excitement and enthusiasm for dbt in a way that I had never seen for a different product.

After a year of watching the company closely, my conviction only grew. I ended up calling Tristan in early 2021 and asked to join full-time. I'm now a part of the product team.

What's your take on the composition of data teams? In particular, who makes buying decisions around dbt and transformation vs the data stack and the data pipeline? How is that different for an early-stage company vs people who are migrating later?

I think titles for data roles can sometimes be challenging because they're not descriptive enough.

One of the main personas is a data engineer, someone who thinks about data pipelines, builds the data platform, and is more familiar with traditional software engineering practices.

Then, there are the data analysts who usually do analysis in SQL—their language of choice. They help answer questions for the business.

We have a persona that we like to target—the analytics engineer – which is a hybrid of the two. They understand the business context really well and work closely with their business stakeholders. They primarily work in SQL, but they also want to be the ones who create clean data assets in their production warehouses, a task previously owned by data engineers.

Who do we target?

Customers of dbt Cloud sit on a wide spectrum and look really different. If you have a cloud data warehouse, then you're a good candidate to use dbt. We have teams of 1 that like dbt, going all the way up to teams of 1000+ analysts.

For the majority of people who use dbt, it's a bottoms-up and self-serve go-to-market. They gravitate to the tool and get up and running very, very quickly.

However, in larger teams, the decision makers are often heads of a data platform team who sponsor dbt for their org. They roll it out to all the data teams within their organization, and help each of them succeed by platforming on dbt.

Each team member has different priorities. For larger companies, where we're very successful, a head of data is thinking about, "How do I get 1000 analytics engineers productive and successful when I have all these security and environment requirements?" and "How do I think about data quality at scale?" and "How do I think about velocity in contention with data quality?" They're thinking hard about how to create a system to make their teams successful.

The practitioners on the team, or analytics engineers, are the users of the product and usually are our champions and motivate broader adoption in their orgs.

Can you briefly talk about what dbt Labs prides itself on being the best at?

The coolest thing for me, it's why I joined dbt Labs, is the sheer enthusiasm that people have for the product. dbt is not just software in the traditional sense. It changes an entire workflow of how people do their day-to-day jobs.

It's very common for people to say, "I won't work at a company that doesn't use dbt," because what it means for them is, "If your employer doesn't invest in dbt, your day-to-day is going to be hard, and you're going to have all these downstream data quality and reliability problems."

People can describe their business transformation logic and their metrics in dbt code and get the benefits of built in documentation, testing, version control, which makes actual data table creation a lot safer and the quality a lot higher.

You can try to do this in GUI based transformation tools but it's just not as powerful or descriptive as you'd like, or you can pick another tool like Airflow, but that’s maybe overkill for what you're trying to do.

Where I think dbt is exceptional is this Goldilocks approach—we say, "If you know SQL, we can empower you to do all these other amazing things and help you earn the right to create clean tables in your production warehouse." That was really never done before dbt.

You've implied that people have this "a-ha" moment when they start using dbt and that is one of the reasons why people are so passionate about it. Can you give us a historical perspective on the world pre and post dbt?

Imagine that before you could do any of your work, someone gave you a tool that was really not that great for your job—for instance, something legacy like Talend or Informatica where it's always broken. Or maybe you had to go and ask a data engineer who held the keys to production to create the data assets you needed in your warehouse for you.

That was frustrating, because when you needed to work, you had this big dependency on a different team to help get your work done. There was a lot of gatekeeping around letting analysts contribute to prod.

And pre-dbt that was the norm for data transformation. You need to have proper software engineering best practices—like version control, documentation, reviews on your pull requests, CI checks, and tests—and without that, you *shouldn't* be allowed to push code to prod.

What dbt did was create a path forward such that people who were close to the business logic could also be the ones to deploy their pipelines and create their data tables. dbt created an easy way to collaborate and productionize data transformation work that was accessible to more people.

Is there a core way that you think about your business model as an open source company? Can you cite any analogy to help us understand the distinction between what customers pay for and what is open source?

So dbt Core is a language or framework. It's SQL-based with some extra bells and whistles, like configuration. We have Jinja and macros so that you can be more productive with SQL than you could otherwise.

We’ve built a compiler that translates your dbt SQL code into instructions for your data warehouse to create data assets. That's dbt Core. It's open source and will always be so, because it's a language, and it's really valuable for companies to have a place to describe something as critical as their business logic in a framework that's not proprietary. dbt is a standard, and it will always be freely available for our users.

On the cloud side, we build a lot of proprietary software that complements the dbt workflow that helps customers accelerate their time to getting up and running and improves their data posture and velocity.

The cloud product has an interactive development environment, so you can create your dbt models more easily. It has CI checks, so, when you're trying to promote your code from development into production, you get a staging environment that automatically tests your dbt code and data quality.

We have a scheduler that powers building models so that your tables can remain up to date and fresh, and you don't have to maintain that infrastructure yourself.

We’re launching a semantic layer which will power your business queries, and allow you to have good governance so that you can write your business logic once and have consistency wherever it's queried or consumed.

We also host your documentation for you. The cloud product, what we sell, supports the full workflow around the analytics engineering practice, supercharging the open source language and framework.

Is GitHub to Git a helpful analogy? How do you think about that?

It's a great analogy. dbt is more similar to GitLab’s approach where Git is the open source framework for how you do version control, and GitLab sells all of these products around the software development lifecycle. They sell products such as CI, roadmapping tools, and security products that are proprietary, but complement the main version control, open source technology.

Can you talk about how the team at dbt thinks about building and scaling community? Is their approach essentially the same as other open source community building efforts, or is it differentiated in some way?

The founding story of dbt Labs is well-documented. We were a consultancy before we were a software company and our founders, Tristan, Drew, and Connor, created the Slack community to talk with their clients. It was a forum where they could have conversations around the practice of data analytics, or be friendly and chat about some of the things that they were working on.

It slowly started to grow as this hub where people came to get help, learn, teach, socialize, share. It grew over time in a really authentic way. If you go into the dbt community Slack today, you see a lot of people asking questions to get help with dbt. But more than that, people are having conversations about the industry, new technologies, their work, how to manage teams. It's become this watering hole where people go to learn, ask questions, and even just catch up with their friends.

A majority of conversations happen in DMs. I have this direct line to pretty much anyone in the data community where if they want to get in touch with me, they can, not on LinkedIn, but on dbt Slack. They'll just send me a DM and I meet a lot of people that way.

We now have over 30,000 people in dbt Slack. But we work hard to make it feel small so that you can get the same interesting conversations that you could with a smaller audience. We do a lot to keep that spark. We organize channels around topics of conversation. We have champions of channels, sometimes light moderation, and we try to keep the general tone and feeling of the community very positive and optimistic, because that's what we stand for. And we also have other outlets for the community that don't evolve around Slack—It's meetups, conferences, and Twitter communities too —all about dbt.

dbt Labs is a very long term-focused company, so, we want to make sure that it doesn't ever feel like you go to that Slack and are being sold something, either by us or by other vendors that are in the Slack group as well. We hope that if we can continue to create a great place for people to talk about data and deliver great products, then it will all compound over the long-term.

Can you talk about pricing and packaging, especially with respect to retaining and expanding customers as they reach enterprise scale? How do you think about customer success, sales teams, and that aspect of the business?

We have three plans today: Developer, Team, and Enterprise, and it’s a seats based pricing model.

The developer plan is for individuals who want to get started for free and learn what dbt does.

The team plan is better for collaboration and has additional functionality like access to our API as well as higher build concurrency.

Our enterprise plan is aimed at our larger customers that have more security, compliance, and governance needs unique to bigger companies, and it’s a sales led motion.

The way that we grow is we'll often land with one team in a large enterprise. But there might be a dozen or a couple dozen data teams in a large organization. If we’re successful with one team, we have a natural expansion path to other teams as well. Or we might start with the Snowflake business, and we expand to get the Databricks business too. At large companies, there’s often more than one warehouse in use.

SQL has been the lingua franca for the modern data stack. What are the advantages and drawbacks of having that as the standard from dbt's perspective? Can you talk about the trajectory in which the product is going? What can writing Python unlock?

I think the choice to be SQL-based was a huge advantage for dbt because it is so accessible to lots and lots of people. It has this staying power that's unlike any other language in analytics because you have relational data warehouses. SQL is the main way people do their dbt transformations.

We are introducing Python as a transformation language as well because we view SQL and Python as two languages that are better at different things. We always advocate for people to use the right tool for the job, and we're increasingly seeing polyglot teams.

It's really common for many data science or machine learning workflows to consume from cleaned, prepped data tables that were produced with SQL-based dbt transformations. Now with the introduction of Python models in the broader dbt DAG, we make the whole workflow a little bit smoother. You can have better handoffs between different teams that actually rely on the same underlying data.

How do you think about the balance between capturing value and creating it? Is this a part of dbt's philosophy?

We have a value at dbt Labs that “profits are exhaust”, and I really believe in this value. You have to build something that people care about or see enough value in that they're willing to pay for it. Without a great product that improves the way teams work, they won't exchange money for it.

We want dbt to be a standard. Our users see a lot of benefit of dbt being open source forever. If you’re going to spend the time to write all of your transformations and business metrics in dbt, you want to be really confident that you own that. That's yours. That's what open source enables us to do. It gives people the confidence and comfort that they are the custodians of that logic whether they decide to pay for dbt Cloud or not.

Then, it's our job to build enough proprietary software that helps complement the open source framework such that you're better off running dbt on our cloud products, on our infrastructure, than you would be otherwise.

There are many companies that have different needs that are very happy using open source only. And we're happy with that outcome. But for many companies that don't have unlimited engineering resources and value their time, they see that it's a lot faster to use our cloud products. They're looking for reliability and durability, and they don't want to have to maintain dbt themselves. For them, dbt Cloud's a great option.

What do you make of the vision that every B2B SaaS app will be built on top of the customer's data warehouse as the data layer? What must happen in terms of latency in the whole data pipeline that needs to be built for this vision to be realized? Where does dbt fit in when speaking of the modern data stack?

In a lot of ways, it's already happening. We see customers of dbt running dbt transformations at increments of 15 minutes or faster, which means that they have some kind of real-time data need. I'm constantly surprised, in a good way, at how people stretch dbt in their warehouses to power production software.

We have one customer, a corporate training company, which powers their customer facing admin dashboards that give the details on which employees have completed which certifications or training, with dbt in near real-time.

I think most people would've thought that this experience was built by a software engineering team, but actually, we've empowered an analytics team to build customer-facing production software, which is exciting.

I don't subscribe to the belief that "We will have one data store that can do both analytical and transactional work equally well." Software engineers will continue to pick the best database for their needs, and almost always, warehouses are not a good choice to build production applications on top of.

However, on the fringes, we will see more and more examples where warehouses will be pushed to near real-time use cases to power what people are calling data applications, and dbt will have a very central role in that shift because we describe the underlying logic of what the application needs to do.

Pre-dbt, people couldn't even have this conversation because data wasn't tested or the quality was so low that it couldn't be pushed directly to customers. However, with proper guardrails and checks in place, you can now have data that lives in your warehouse, and is prepped with dbt, power all sorts of use cases.

What excites you about new things that are being built on top of modern data stacks in dbt that couldn't have been done before?

It's not any one application that I'm most excited about. It's more that we are increasingly breaking down data silos so that data can be used in many contexts. This is hard to do right because centralizing data in one storage layer is never really possible for large companies, but increasingly we see data tools being composable so that you can be sure you're getting the same data information anywhere you view it.

Then, data conversations become a lot more interesting, because you don't have to constantly question, "Is this data right? Can I trust this data?" High quality data can be pushed to more real-time use cases, like a production application. Or, it can be pushed into systems of action like Salesforce where I trust this data so completely that I can go and send a marketing campaign email off of it without human intervention.

That, I think, is really exciting. It's more centralization, consistency of business logic that people can depend on across different data stores for different use cases.

There's a distinction made between transformed data and metrics. There's also some discussion about whether metrics should be a separate category in modern data stacks. How do you see some of these competing visions?

This is a big, compelling reason why I joined dbt Labs . I believe so strongly that metrics belong with dbt.

Data transformations are batched and pre-computed, so you know exactly what that table will look like. Metrics are business definitions like CAC or lifetime value of a customer that are very bespoke to your business. They're often computed in real-time at the moment of query, so the user can pick over what time period they care about this metric to be calculated for.

You can't pre-compute these metrics in pre-transformed tables, because it'd be too expensive to do so. It needs to be computed on-the-fly when the person asks the question, "What does this metric, over this period of time look like?" At dbt, we believe that metrics should be tightly coupled with your transformation logic. You don't want to describe these things in a million spots.

Traditionally, people describe their metrics in their BI layer, so it's tightly coupled with Looker as an example. LookML is a

proprietary query language built for Looker. It's a lot of work to get that semantic layer up and running. But, what happens when you want that metric to show up in some place other than Looker? You've got to describe and define it again.
The beauty and power of metrics being coupled at the layer at which you do your transformations is, you can have lots of consumers that read from the very same metric, whether it's a notebook, a BI dashboard, or even, a data catalog. You want metrics to be consistent no matter what context they show up in.

It's valuable because it's not tightly integrated with your warehouse or your consumption layer, so we can serve consistent information across any context.

What about positioning this metrics layer as something completely independent between BI and transformation?

We don't need another layer. In fact, that's the last thing we need. I think adding metrics to the transformation layer is very natural. Doing dbt transformations and separately doing your LookML metric definitions in two different tools is a lot of work. It'd be far easier to have one centralized place alongside your transformations where you write metrics once and never have to think about it again.

What prevents a data warehouse company like a Snowflake from getting into this, on the one hand, vs an ELT company like Fivetran?

Nothing prevents warehouses or Fivetran from offering dbt or other transformation technologies. They already do so. dbt is open source, so they can offer simple dbt runners.

For our users, many don’t want the business logic of transformations and metrics to be as tightly coupled to the warehouse or BI layer. Many customers have multiple warehouse vendors or multiple BI tools, so you don’t want to reproduce the logic in different storage and consumption layers. In a lot of ways, our abstraction and vendor neutrality makes dbt so powerful and useful to consumers.

The other reason why people buy dbt Cloud instead of using a dbt runner in a Fivetran or another EL product is because we offer a far richer overall product experience. The people who do transformation work are often different from those who do set up the extract load flow. So, it's a separation of tools that match a separation of responsibilities. The transformation layer needs to be closer to the business needs versus the pipes of data movement, which is what Fivetran or the EL provider owns.

Is there anything in particular that dbt has learned from the success of other open source companies that have preceded it? How do you position dbt vis a vis incoming cloud companies—Amazon, Google, Microsoft?

Clouds like to sell compute, storage, and infrastructure, and while dbt is infrastructure, it’s also a workflow tool which is harder to copy. We invest in building productivity software for analytics engineers with our purpose-built IDE and CI checks, alongside the platform parts of our offering like the semantic layer and scheduler.

From the beginning, we've always built proprietary products that complement our open source offering which helps protect us from competition. We also have a loyal community which is an important moat for our business.

What are the key tailwinds behind modern data stacks? How is dbt positioning itself to capitalize on that?

The only reason we can have this conversation is because we're 10 years out from when cloud warehouses were created and launched. Moving lots and lots of data to BigQuery, Snowflake, Databricks, Redshift, and others is the norm.

We continue to believe that more and more data will be stored in the cloud and that dbt is the layer that will help people make sense of that data and serve it to their businesses in useful ways. Most analytical workloads are for reporting and visibility purposes, but we'll see new use cases emerge as well and dbt will evolve to help serve and expand the market beyond where it is today.

dbt recently raised $400 million. It was last valued at over $4 billion. What is the upside from there? Is it more a thesis around the big category expansion of transformation, or does dbt need to go out and launch its own data warehouse/ELT product to live up to that valuation?

The cloud warehouse market is enormous. The combined market cap of the cloud warehouse players is probably close to $200 billion today, and we believe, transformation and the semantic layer can easily be at least 15% of warehouse spend. We're vendor-agnostic, so that's a $30 billion opportunity right there. And that's just with our core competency, today.

We don't have plans to build a warehouse. We don't have plans to build an EL product either. Those are extremely hard to do. I have a lot of respect for people who build those products, but for us, there's tremendous value being Switzerland and working with these vendors as partners. We're excited to be executing right where we are.

In five years, if everything goes right for dbt Labs, what does it become? How will the world change as a result?

I think the success of dbt could change the world in really big ways. People are very excited about the future of data apps. I am too, but I'm probably more excited about a mundane future where people just trust that their data is on-time and correct and, little-by-little, they use data in their lives to make big and small decisions.

Just like I don't get into my car without using navigation, I expect that more and more people will depend on dbt prepped data in lots of different contexts in their lives.

Disclaimers