EPISODE 1159

[INTRODUCTION]

[00:00:00] JM: Fivetran is a company that builds data integration infrastructure. If your company is performing ELT or ETL jobs to move data from one place to another, Fivetran can help with that data movement from source to destination. Once the data is moved into a data warehouse, a tool called DBT, or data build tool, can be used to transform that data more effectively. We've done shows previously about Fivetran and DBT, and in today's show, George Fraser of Fivetran returns to discuss the cross section of these two technologies and what his company is doing around that integration point. I hope you enjoy the episode.

[INTERVIEW]

[00:00:42] JM: George, welcome to the show.

[00:00:43] GF: Thanks for having me.

[00:00:45] JM: Last time we talked about Fivetran in some detail, we covered the basics of the company. I think today we're going through more of an advanced deep dive into what Fivetran has been up to since then. But can you just start by giving an overview of Fivetran and data integrations in general? What is it that your company does?

[00:01:03] GF: Yeah. So Fivetran provides automated data integration. That means we connect to – For each of our customers, we connect to their business systems. So that can mean the production database that powers their app. It could mean their Salesforce that their sales team is using every day. It could be their support system, their Jira issue tracker, their ad networks, you name it. All of the systems that you use to operate your business. We connect to those as sources and then we replicate the data that lives in those systems automatically with zero configuration into a data warehouse that's also going to be the customer's data warehouse. And so that would be Snowflake, or BigQuery, or Red Shift, or Databricks, or one of the other destinations that we support.

And once the customer has all of their data in a single database, their analysts can write SQL

queries that produce BI dashboards, that help them figure out what is going on in their business that power production systems. At the end of the day, data warehouses are databases. So really, you can do anything with that data. And our mission is to solve this foundational problem of getting all your data in one place in an automated zero configuration way so that the data is always up-to-date, it's always correct, and you as the user don't have to worry about the details of how that happens.

[00:02:32] JM: And again, last time we talked about the basics of the data integration story and how one of these problems that looks very small upfront, but then as you dive deeper into it, it becomes more and more complex. Tell me about some of the complexities in the advanced problems that have come up as you made it further and further along the Fivetran journey.

[00:02:53] GF: Sure. So this is a space that's undergone a lot of change in the last few years. The data integration problem has been around forever. Since there were two computers, there was the first data integration problem. And so companies have been doing this for a long time. Companies have been doing data warehousing for decades. They've been doing BI and analytics for decades. What has really changed in the last few years, and that's primarily what we talked about last time, is the emergence of these superfast, infinitely scalable cloud data warehouses. And the emergence of these tools, like Snowflake, like Red Shift, like BigQuery has caused the whole ecosystem around them to shuffle. Their sort of a game of musical chairs were changing the way that we build data systems. And we're changing the order in which we do things.

And I encourage anyone who wants to hear more about that to listen to our last interview where we get into detail. And I'll just say that the principle change that we're seeing as we're learning on these data warehouses more and more as they get better.

So since we last talked, there has been particularly a big change in the transformation layer of this stack. So if you're a user and you're using Fivetran with maybe Snowflake and a BI tool and a bunch of other workloads that you're running against that data warehouse, you're going to have this problem, which is transforming the data that Fivetran delivers. So when we deliver the data to your warehouse, it's going to arrive in a normalized schema. It's going to be a reasonable schema that reflects the nature of that source, but it's not going to reflect the

particular problem that you are trying to solve with your data warehouse.

And so every Fivetran customer, since our first customer has had some solution for transforming the data after it arrives in the data warehouse. And so for some of them, it was as simple as writing views in SQL, or some of them, they adapted other tools like Airflow, or Matillion that focus on orchestrating those SQL transformations. But that area has been for years a little bit of the Wild West. There are a lot of different solutions floating around. Every company does it a little bit differently. It's a core problem that you have to solve if you're going to set up a modern data stack. But there're been a lot of different solutions, and for a long time, it was unclear what best practice really was there.

However, over the last couple of years we've seen the emergence of this tool called DBT. DBT is an open source tool. It stands for data build tool. And what DBT does is it really brings the best practices of software engineering to analytics. So analysts, traditionally, they wrote SQL, they use BI tools, they mostly use a commercial software, and now the commercial software work differently. And they didn't really have the equivalent of like continuous integration, or testing, or infrastructure as code. All these things really hadn't crossed-over for the software engineering world to the analyst world. And the goal of DBT as a project, it was really to bring those best practices to analytics. And what we've seen at Fivetran over the last couple of years is that our smartest customers, the once who are on the leading edge, always doing the best new ideas in their analytics program started adopting DBT on mass.

And I mentioned that it's an open source tool. It's also a very active community of users who mostly collaborate in this workspace called DBT Slack. It's a Slack community where you can find significant fraction of all the people who use DBT, sharing best practices, learning from each other. And there's a company called Fishtown Analytics, which developed DBT and is the primary sponsor of it, and has a product called DBT Cloud. But the project truly is bigger than any one company. And we've seen it get bigger and bigger over the last couple of years. And as a result, this year, Fivetran has made a big bet on DBT, and that bet comes in a couple different forms, which I'll explain in a minute. But I just wanted to pause there and see if you have any more questions about what the heck is DBT? How does it fit in to the modern data stack?

[00:07:25] JM: We did have a show about DBT with Tristan that people can listen to. But yeah, I

think it's worth a refresher. Could you just explain what DBT is? What problem is solves?

[00:07:35] GF: Yeah. So fundamentally, when you set up DBT, you create a GitHub repository, and that GitHub repository is going to have a bunch of SQL queries in it, and these SQL queries are transformations that convert data from whatever format it arrives in your data warehouse. So if you're using DBT in conjunction with Fivetran, it would convert from the schema that Fivetran delivers the data in into a schema that is designed to support your particular analytics problems.

And it's not just SQL queries. There's some additional metadata you have to include to tell it things like when to schedule these queries. How to materialize them? But it's fairly lightweight the things that they add on the SQL. It's a very SQL-centric tool, which is fits with the larger trends in this industry, but it definitely is a change from the history. Most of the commercial tools that people were using 10 years ago were point-and-click UI-centric tools rather than SQL-centric tools.

And as I mentioned at the beginning, DBT, code is always going to live in a Git repository. And so that means that you're going to have version control. If you're using a tool like GitHub or GitLab, you're going to have concepts like pull requests. So you're going to be able follow the best practices of software engineering like code review, but in the analytics world. And that's very powerful. That's one of the first benefits you get when you adopt DBT.

And the other things that DBT brings to the table that are different than the way that this work has traditionally been done is they really bring some of the concepts of continuous integration into analytics. So your DBT transformations are going to be run by some kind of a continuous integration tool. You can use tools like CircleCI or GitHub Actions. You can also use DBT Cloud, which is Fishtown Analytics product. DBT cloud will do orchestration. It does a lot of other things as well. It goes beyond just the orchestrating that DBT transformations. And now you can actually run them directly with Fivetran. But the overall theme here is taking those best practices that have been proven to work well in software engineering and bringing them to analysts. And because it's code-centric and because it is an open source tool and a real community around it, one of the great benefits that I see as an outsider – I didn't create DBT. It was news Fishtown Analytics that created it, but as an outsider who's been watching it for years, one of the great

benefits of the way that they've gone about this is that if you're an analyst and you're using DBT to do your work every day at your company, you're investing your own skills in this tool, right? You are spending your own brain cells on learning and remembering how all this stuff works.

And the fact that DBT, number one, is fundamentally SQL-centric. And number two, is a true open source tool with many companies using it and a community around it, is a really great thing for those analysts. It means that they're investing their own skills in a tool that they're going to be able to take with them wherever they go. And that's part of the reason I think for the enthusiasm around it. Rather than spending their time learning a new generation of proprietary tools, they're spending their time by investing in themselves. And that is perhaps the biggest reason why I see DBT emerging as a standard for doing this kind of work.

[00:11:17] JM: And then as far as adopting it as a product within Fivetran – So Fivetran is for data integrations. Help me understand how DBT fits into your product line.

[00:11:27] GF: Yeah. So Fivetran's fundamental mission is to make access to data as simple and reliable as electricity. And so we're primarily in the data replication business, and we spend most of our time thinking about things like how do we support evermore data sources? How do we make the latency as low as possible? How do we make the reliability as high as possible? However, every single Fivetran customer has to solve the transformation problem. They have to solve the problem that DBT solves.

So everyone who uses Fivetran somehow needs to set up a solution to this. And for a long time, we didn't really have a recommendation in this area. Every Fivetran customers was kind of on their own to figure out how to do this. But as we saw DBT emerging as the leading standard for solving this problem, we started to make a bet on DBT in our products, and that takes a couple of forms.

So number one, we started building DBT packages. One of the concepts that DBT takes from software engineering is the concept of dependencies. The idea is you can write code and publish that code so that it's reusable by others. This is very obvious to anyone who is a software engineer, but in the analytics world, this is actually new. This really has not crossed over until DBT. And so if Fivetran, one of the things we've started to do is publish DBT packages

for all of our sources. I mentioned earlier that Fivetran always delivers the data in a normalized schema that reflects the source. It's a reasonable schema, but it's not customized to any particular customer.

So that means that the schema for a given source like Jira, or like Salesforce, or like Facebook Ads, is always going to be the same except for custom fields. We'll make it slightly different. But the core schema will always be the same for every single Fivetran customer. And that means that we can write DBT packages that transform that normalized schema into a dimensional schema that's more friendly for typical analytics projects that people do with Fivetran data. We can write a prepackaged DBT package that does that that is going to be a great starting point for any customer who's trying to work with Fivetran data. And so we started to do that about six months ago. We started working our way through our connector, writing DBT packages for all of our connectors. And now we have a whole solutions team that does this full-time. Their primary goal is to get as many companies as possible using these DBT packages.

The idea is for some of the SQL that people write on top of the data Fivetran delivers, some of it is custom. It's different for every customer, but some of it is similar across all customers. And so to the extent that we can do some of this work once and then publish it as a package that your analyst can just put a dependency on, we can save everyone 30% of the work of their analytics project that starts after Fivetran delivers the data. So that's a big bet that we've made internally at Fivetran, and we've really started to see adoption takeoff, especially in the last couple months. If you're a DBT user and you're a member of the DBT Slack community, you'll see Kristin, who leads that team in there all the time talking to people about what should be built next. Is this working for you? Have we thought of all the right use cases for each of these packages? We're really excited about that. We feel it's like the second layer of Fivetran. The first layer is the data replication that puts all your data into a single database in a normalized schema. The second layer is these reusable packages that act as an accelerator for you on your analytics project that are going to hopefully take out the first 25% or 50% of the work that you have to do taking that normalized schema and turning it into a dimensional schema that is friendly to BI tools.

And then the second big bet we've made on DBT as a mechanism for orchestrating transformations is that we've added the ability to natively run DBT packages inside of Fivetran.

So if you want to run those Fivetran packages and you want to run your own DBT code that you wrote yourself, you can now do that just using Fivetran. Fivetran is acting as that CI tool for running DBT on the server side. And the reason we did this is because, as I mentioned earlier, every single customer who uses Fivetran needs to have some solution to this problem. And we wanted to be sure that every Fivetran customer, if they don't already have a solution for this, whether it's something other than DBT or its DBT Cloud, or DBT running at CircleCI, we wanted to make sure that there was a solution that basically comes in the box.

You may recall that Apple announced yesterday that they're no longer shipping power chargers with their iPhones, because there're so many power chargers out there apparently. Well, this is kind of the opposite of that. We want to make sure that you have a charger that comes with your phone.

[00:16:58] JM: I think it's worth drilling even deeper on this CI analogy. Could you describe this comparison with continues integration tools and DBT?

[00:17:09] GF: Sure. So in software engineering, your software engineers write a bunch of code, and that code might be in Java, it might be in Python, it might be in who knows what? But at the end of the day, that code sitting in your repository doesn't yet do anything. Somehow you have to build that code and deploy it to some kind of production system. And so with SQL, your analysts are the ones writing the SQL queries. They're manually running this code as they work on it and test it and look at the results against your actual database. But then at the end of this process, somehow you have to actually run all the SQL code that transforms your data from one schema into another that's then ready to be used by your BI tool or your production systems or whatever it is you intend to do with that data.

Somehow, this code has got to get run. And so that's where the analogy to continuous integration comes in. So when you use DBT, you can run it locally to get up and running, but then you're going to want to actually have a server somewhere that is going to run your DBT transformations every time you make a change to the code and every time the underlying data that's getting transformed gets updated. You're going to want to rerun those transformations, and that needs to happen somewhere. And so you can do that with conventional CI tools. You can do that with Airflow. You can do that with DBT Cloud. And now you can do that with

Fivetran.

That's kind of the build and deploy analogy with CI. The other part is the test analogy. And so testing, it's funny has, is very underdeveloped in the analytics world. A lot of big data warehousing projects basically have no testing. Now, SQL being a declarative language is less error-prone. So that's how you can survive without testing in the world of SQL. But you can still make mistakes in a declarative language, and it's a tough problem. The lack of testing in the analytics world, the lack of automated testing, that is, as supposed to manual testing is a big pain. And one of the things the DBT does is they do introduce some testing capabilities into the SQL transformation problems. So they have the ability to track things like uniqueness of columns. They can run. You can write queries that act as like unit tests for your data that verify custom invariance that you know should be true of your data after it's transformed. I think this is an area where this will continue to develop over time. But even now, this is one of the marquee features of DBT. And one of the big things that people get out of it is the ability to run basic automated tests against their data. And again, that's a feature that is going to get invoked when you run DBT on the server-side, whether you're doing that with Fivetran or something else.

[00:20:17] JM: Let's talk through a holistic picture a little bit more and revisit some of the topics we discussed in last episode. So Fivetran sits in between your various data sources and whatever database or data warehouse you're doing to do the big analytic queries. Give me a little bit more context in terms of what Fivetran is doing in terms of the ETL process from one source into the data warehouse.

[00:20:50] GF: Yeah. So we view our job as fundamentally to create a perfect replica of the date that lives in your source, whether that source is a database like MySQL or an API like Jira. Our job is to create a perfect copy of all that data in a reasonable schema and then to keep that data up-to-date as the data gets changed in the source and as the configuration of the source changes. That means if you add a new column to your data source, it's part of Fivetran job to go and create a new column in your data warehouse. If you change a type, we have to propagate that type change. And all of this needs to be automated.

Historically, a lot of these processes have been human-powered, and at Fivetran we are absolutely religiously opposed to that. Fivetran is all about automation. The only way to make

systems that will actually be correct over long periods of time is to build automated systems. So that's our job, is to create that copy of the data that lives in your data warehouse, that's always up to date, modulo some latency. And we're always trying to drive that latency down. Today, about the best we can do is one minute. And if all the stars align, we can potentially get one minute. But it's a never-ending project. I mean, for our first customer, years ago, our latency was 24 hours was the best we can do. So there's always that next frontier of reducing latency.

And then once the data is in your data warehouse, as I said, it's in a schema that's reasonable, but it's not really optimized for any particular use case. The most common thing people do with data warehouses is business intelligence. And for a BI tool, what you really want is a de-normalized dimensional schema. If you're not familiar with that term, a dimensional schema is basically just simplified view of the data that is easier to understand, that has fewer tables, fewer larger tables. Now when I describe it that way, that makes it sound like it's just better and you might ask, "Why don't we just always use dimensional schemas? Why doesn't Fivetran deliver the data in a dimensional schema?" The reason is when you convert data to a dimensional schema, it's a lossy conversion. Certain questions then become impossible to answer.

And so you don't want to do those kinds of irreversible changes to the original data. It's important that you preserve that original normalized copy of the data so that if you ever change your mind about what kinds of questions you want to ask, what kind of dimensional schema you need, it will be able to do that. But that is the most common scenario for using Fivetran. You want to convert into a dimensional schema and then you're going to connect BI tools to that. An and that is where you're going to – That is where DBT has that role of orchestrating those transformations, which are either coming from packages that Fivetran wrote for you, or your own SQL queries that your analytics team ran the convert that data from one schema into another.

And then the last step of this, there's this like last mile problem. You have your data all in one place in your data warehouse. You've transformed it into a schema that makes sense for the problem you're trying to solve. Whether that's creating a dashboard or powering some kind of production system, but then there's this last mile where you actually have to do something with the data, right? Like it's sitting there in the database, but it still hasn't really done anything.

And for that last mile, if the goal is to create a dashboard, you're going to use a BI tool. There're

lots of great BI tools. That's a well-known area. If your goal is to take action automatically based on that data, you're going to need to take that data out of the data warehouse and go do something with it. Put it into, say, Marketo. Maybe the result of all of this work is lead scoring. That's a common scenario. You're saying which prospects are the most likely to buy my product? And so you want to take those lead scores and feed them back into one of your sales and marketing tools. There's a company called Census. I don't know if you've ever talked to them, but they focus on that problem. I'm getting the data out of the data warehouse into production systems. And then sometimes people build their own production systems, so their software engineering team will just query the data warehouse directly. So we have lots of customers who do that, who will just operate production systems on top of the data warehouse. And those use cases are sort of hard to describe in a summary view, because they're all kind of unique to those companies. But those are really some of the coolest use cases that people do with the date that Fivetran delivers. I mean, we have customers who use data Fivetran delivers to run billing, who use data Fivetran delivers to run payroll. I mean, really, anything is possible in that scenario. So the relationship of Fivetran as a tool and as a company and DBT as a tool and Fishtown Analytics as company in terms of technology is we're both part of this pipeline from data all the way to some kind of insight or action.

And then we also have an interesting commercial relationship as vendors. We're partners. We sell into a lot of the same accounts. So we have tons of shared customers. We actually have some shared investors. Andreessen Horwitz is an investor in both companies. So we're commonly helping to solve this problem together, us and Fishtown Analytics, the primary sponsor of DBT. And then we're also both participants in this DBT ecosystem.

There is an interesting overlap now in what we do. So Fivetran can run your DBT transformations. That's also one of the features of DBT Cloud, which is Fishtown Analytics product. And so there's actually some overlap despite us being friends and partners. There's overlap and what our products do. And we actually recently published a blog post about how we think about this. So we sort of wrote up a conversation that Kristin and I had over Slack about how we're thinking about the relationship of the companies going forward. And I encourage everyone to read it. But long story short, we really think there's a great opportunity here to create a vendor ecosystem around an open source tool that is collaborative and that is customer-centric that creates a good experience for the users of this tool.

I think there's a long history with open-source ecosystems getting divided by vendors, especially if you look at like a Hadoop ecosystem, for example, or if you look at like MongoDB with AWS making incompatible fork for their own offering. There's a long history of open source tools getting really fragmented by vendors each trying to pull the tool in a different direction. And I really see and I know Fishtown Analytics sees DBT as a standard as a way, as an open-source standard for orchestrating transformations in a data warehouse. That's the future we all want to have. And we are thinking a lot about how to create an ecosystem where the various companies that participate in that ecosystem really collaborate in a way that's best for the user rather than trying to play tug-of-war with the same rope. And we'll have more coming out in the future about that, but it's something we all care about and are really trying hard to navigate that well.

[00:28:45] JM: Could you tell me a little bit more about what you see is the future of this data ecosystem? Like what are the missing pieces in the puzzle?

[00:28:53] GF: The last mile piece is still – It's funny. It kind of goes from bottom to top. I mean, we kind of started at one end with the data warehouse and the big changes to the ETL process that I think Fivetran pioneered. And now we're seeing people really coalesce around DBT as a way to orchestrate transformations. And kind of the next stop is that last mile. How do you do something with that data? You can have great data. You can have it all in one place. You can have I t transformed into a wonderful format. But at that point, it's still just a bunch of tables sitting in a database that hasn't actually done anything yet. And I think the BI dashboard, that's pretty well-trod territory, but that's just the beginning of what you can do with these systems. These massively parallel data warehouses are so useful for so many different things, and we're seeing our customers discover and invent all these other use cases. I mentioned a couple earlier.

There's this real Cambrian explosion of what you can do with these data warehouses when you really have all your data in one place. And I think that's some of the most exciting territory in this ecosystem. There's a couple different categories that we're seeing emerge. So one of them is people building vertically integrated solutions around the data warehouse. So we're seeing other vendors who take data warehouse like Snowflake or Red Shift or BigQuery and they embed Fivetran. So you actually can embed Fivetran similar to how you would embed Stripe. We call

that powered by Fivetran, and there's a whole API for doing that. And then they embed some kind of usually data visualization layer, and they build a vertically-integrated end-to-end solution around a particular use case like marketing, or customer management, or you name it. And I think the thing that's really great about that category is it has the ability to deliver whole lot of value very quickly with very little effort from the user, and I think it's going to allow companies that don't have a really strong internal analytics or data science capability to tackle some of those problems by using these vertically integrated tools. So that's one category.

And then there's like a related category of companies that are building tools that operate directly on top of the data warehouse. A good example of this is Narrator, which is a tool for understanding the people that you're interacting with on your website, on your app, in your Salesforce, you name it. And the way Narrator works is it's built on the assumption that you're going to have a data warehouse that has the data from all these systems in it. So you're going to be using something like Fivetran. You're going to be – And then it's going to connect to that data warehouse and actually operate directly on top of it. This is a really cool implication of the fact that all the new data warehouses are cloud-based. They're all in the public cloud. So your data warehouse is no longer a box in your basement. It's a system running in the public cloud, and that means it's possible for other tools to sit on top of it and operate directly against it. So that is a really interesting use case that we're seeing emerge and we're seeing more companies build tools that work that way. And then the last category that I see a lot of these days is this let's get the data out of the data warehouse and back into production systems.

So how do we get the result of this chain of analysis and transformation and take action on that automatically? So instead of a human being sitting there and looking at a dashboard saying, "Okay, these are the five customers who are most in danger of churning. I'm going to open a ticket with each of them to talk to them about this." You can just open that ticket automatically. And again, it's like every single use case of this is very unique. So it's hard to describe, in general, what this looks like. But the thing that they all have in common is that you have all this data in one place. You can find out all of these interesting things about it. And then instead of having a person look at the results and take action on it, you actually close the loop. And so you go through the data warehouse and then back out to the world and take action on this automatically.

So there are vendors that are working on tools around us, but then there're also just a lot of customers who are building this themselves, who they're engineering teams, create a process that pulls that data out of the data warehouse and goes and takes those actions on it. And that is a huge vector all of its own. That's going on a lot right now.

[00:34:02] JM: I think you've talked about people using these data warehouses in production as like kind of production databases. That sounds very foreign to me, because I thought usually you're using like an OLTP database, like a SQL database, or a Mongo for your OLTP transactions. You're not using a data warehouse, I thought.

[00:34:20] GF: Yeah, that's right. It was weird to me to when I started to see customers doing this. I thought, "What the heck are you doing? This is not how data warehouses work. These things are supposed to support analytical workloads." And it's still true that analytical workloads and OLTP workloads are very different.

And, for example, if you are selling a product that people are sitting there clicking on all day and updating data in your database, you are going to want an OLTP database to support that. Postgres is not going anywhere. However, one of the things we see is that there's actually a very blurry boundary between analytical workloads and transaction OLTP, record-oriented workloads is how I like to talk about that. They're sort of the record oriented-workload world. And then there's the scan-oriented world.

There're a lot of companies that have workloads that sit somewhere in between. So it's a production workload. However, it leans less towards updates and more towards scans. It isn't doing 10,000 tiny things per second. It's doing 10 big things per second. And we see a lot of people doing this kind of work, particular with Snowflake, because they do a really good job with support for concurrency. We see a lot of customers running these sort of in-between hybrid workloads that are kind of analytical, but kind of productional workloads.

[00:35:53] JM: Do the costs add up if you try to use a data warehouse like that for those kinds of online, like user engagement queries?

[00:36:02] GF: They sure can. You need to be very careful when you run these kinds of

workloads and how you do it. And sometimes you need to do optimizations. Like you build a dataset in your data warehouse that is ideal for some production workload you want to do and you prove that it works, but then as a cost optimization, you need to create a replica of that dataset in a row store like Postgres, because the read workload you want to do is like one row at a time. So it can definitely get complicated, and it's definitely tricky to make sure you don't blow up your costs, because at the end of the day, data warehouses, they don't have indexes that are optimized for scans. And that means if you go and start looking up one row at a time, it can get very expensive very fast.

But as I said, there's this is blurry continuum between classic record-oriented production workloads and classic sort of BI dashboards. We're always scanning the entire dataset. There's a lot in between. And I think some of the most interesting things that are going on in the data warehousing world are actually in that in-between zone.

[00:37:11] JM: Can you give a little bit more context as to what exists in that in-between zone?

[00:37:16] GF: Sure. I gave a couple examples earlier, like running billing out of your data warehouse. Saying you get all the data about what all your customers are doing, what your agreements are that your sales team has made with them. How they've used your product. All that data exists in your data warehouse if it's a halfway decent data warehouse, and it can sometimes end up being a really great place to figure out how much to bill everyone. And so we've seen that done. And that's a good example of you're not needing to update that data 10,000 times a second. That's a workload that you if you choose to, you can just run that directly on top of your data warehouse.

Another similar example is running payroll. For some companies, Payroll is really complicated. There're a lot of different inputs to that decision about what someone gets paid for the work they did this month, particularly if you're like employing a lot of contractors. And so the data warehouse can end up being a convenient place to do that. And payroll is not something you run a thousand times a second, hopefully. And so it can be totally economic to do that.

We also see a lot of examples in marketing. So you see companies trying to identify customers to give offers to things in the like on-demand apps. You'll sometimes offers. We've seen

examples of customers powering some of those offers out of the data warehouse, because, again, the data warehouse is a great place. You have all this data about what everyone is doing. You can write a SQL query that can encode that decision about who to make that offer to. And that can be a great workflow. That's where you can you can start to run into trouble with costs with that kind of example. So you can find yourself running those queries so frequently that it can really add up and you have to start to think about a more complicated infrastructure to support that workload. But we've seen some really cool use cases there.

[00:39:16] JM: We're talking that these databases, the data warehouses in the abstract. They do vary in some subtle ways. Can you tell me about the 2020 view perspective on how these data warehouses differ from one another and any kinds of insights you've had from surveying the landscape of Snowflake and Red Shift and so on?

[00:39:39] GF: Yeah. So they all have a lot in common. They all contain some of the same big ideas that really emerged in the 2000's, first Macadamia, and then they were commercialized in various companies. So they all do massively parallel processing. So that means they're designed natively to run across many nodes and to split the work that they do across those nodes, which is very different than the classic row store like, say, Postgres, which fundamentally is designed to operate on a single node. So they all do massively parallel processing. They're all column stores, which means that they fundamentally store the data in a different way. In fact, the opposite way of your typical production databases, and that's part of what makes them so fast doing those scans at reading like an entire table. It's because of the way they store the data.

It also ironically is what makes them slower and more expensive if you just want to read one row. And then they'll have, for example, vectorized query processors. I suspect. We don't know this. And they don't reveal these particular details about how they work internally. But based on what we know from the academic research, we can be pretty certain that all the major data warehouses have vectorize execution engines, which basically is the in-memory version of row store versus column store.

So it's a bunch of these fundamentals that they'll do the same way. And so as a result of that, one of the things you'll see if you benchmark them as we do every year against a typical

analytical benchmark like TPCDS. It's a benchmark design for data warehouses that's been around for years. They all are pretty similar in their performance. And if you just give them that sort of straight down the middle analytical workload, you're going to see very similar performance across all the data warehouses, because they're probably doing a lot of the same big ideas in terms of how do you build a fast parallel SQL execution engine.

Where they really start to differ is in the pricing model and the user experience. So they all have a different history. They were all built on different infrastructure. And so that's where you really start to see some real differences. For example, when you set up BigQuery, in its default configuration, BigQuery is totally serverless. You just pay per query one query at a time, and that has pros and cons. The Pro is you only pay for what you use. If you don't use it very much, BigQuery is incredibly cheap. The con is it's easy to accidentally spend a lot of money, because it will just give you more – The bigger the query you give it, the more resources it will give you. So there's kind of a cost control tradeoff there.

Whereas Snowflake is designed around this concept of virtual warehouses. You create one or more data warehouses. Unlike a conventional data warehouse, a virtual warehouse points at the same dataset. They're all pointed at the same storage. That's kind of Snowflake's original marquee feature, separation of compute from storage. And so that means that you can change the size really easily. You can set up different data warehouses for different workloads all pointed at the same physical storage layer. And it's not it's not quite as serverless as that experience I just described with BigQuery. But it's a lot more serverless than, say, a traditional on-prem data warehouse.

And I could go on and on. There are a lot of differences. Some of them subtle, some of them obvious in the user experience of these different data warehouses. But really, the best way to assess this is just to try them out. What I always tell people is that the performance at the classic analytical workloads is quite similar across all the data warehouses. The differences are in that user experience. The differences are very obvious. You will notice them when you try them out. Just set it up. Play around with them. And if you can't manage to get it set up, that's probably a sign that the user experience is not going to be great for you. So that's what I always advice people, is just to demo them and test them out for their own workloads. And they will immediately see how the different data warehouses have made different choices, and you may

like one way more than another. There're definitely some user profile things there where certain kinds of users find certain things familiar about Snowflake or about Red Shift, or about BigQuery, or unfamiliar. And so they can figure out quickly what's right for them.

[00:44:23] JM: As we begin to wind down, any perspective you offer on the future of data connectors and data warehousing?

[00:44:31] GF: Well, I think this trend of the data warehouse being the center of your data universe is just going to continue. So I think we're going to see people use data warehouses to do more and more things. One of the most important trends that I see happening is the use of the data warehouse to support machine learning workloads. This is kind of the frontier. Doing classic Bi and analytics, using SQL is pretty well-understood. I mean, there're been a lot of these changes in workflow and a lot of productivity improvements over the last few years as I've been describing. But the basics of how you accomplish that are I think well-understood. The machine learning world is a lot more influx. We see companies like Google with infinite resources able to do amazing things. But if you're not Google, it can be pretty hard to get started. And I think the solution to that problem is really going to involve making the data warehouse central to the machine learning workloads.

Now, machine learning workloads, they're not based on SQL. Machine learning workloads are fundamentally all about linear algebra, and that means you're going to use something like Spark or like Python with the numerics libraries that you can get with Python in order to do those workloads. But I think we're going to see more and more people using those tools on top of a conventional relational data warehouse. This claim, some people find very surprising. This is maybe one of my more contrarian views. But I think with the plummeting cost of storage in the data warehouse and with their great flexibility and ease-of-use that they bring, the last problem that really needs to get solved is the connection between the machine learning ecosystem and the relational database ecosystem. And there is this project called Apache Arrow, which solves that problem. It's a protocol for these different tools to communicate big datasets between them fast. And it's brilliantly designed, and it's getting adopted by all the tools in this ecosystem. And I think once all the various readers and writers support that protocol, what you're going to see is there is going to be absolutely no reason anymore. It's not just run your machine learning workloads directly on top your data warehouse.

And I think that is actually going to be a huge productivity breakthrough for those teams, because they won't have to manage a whole other data stack to support this other workload. They'll just be able to tap in to the same data stack that the analyst teams are using for this machine learning workload.

[00:47:13] JM: One more quick question. In terms of the ecosystem, how does the usage breakdown differ between data warehouses and Spark?

[00:47:22] GF: Will, first of all, those aren't necessarily two different things. For example, my opinion is that the Delta Lake, which is a storage layer that is natively designed for Spark, really is a data warehouse. I mean, you can call it a data lake. But from my point of view, when you have an integrated query engine that supports SQL and a storage engine that has a relational model, that's a data warehouse. So some of this I think is just terminology, like I view Databricks as fundamentally being in the same category as these data warehousing tools. They do some other things as well, but I do think they meet the definition. They meet my definition of a data warehouse.

And I think we're going to see them converge more and more over time. So I think you're going to see the support for running non-SQL workloads on top of relational data warehouses like Snowflake and BigQuery. I think that we are seeing the support for SQL in the Spark ecosystem getting better and better. We're seeing a built-in storage layer in the form of Delta Lake. And so you're seeing the convergence of these tools. They're each kind of filling in the gaps on either side. And so I think they're going to get more and more similar over time.

[00:48:41] JM: Well, George, it sounds like a good place to stop. Anything else you want to add last-minute?

[00:48:46] GF: Nope. It's been great to talk again.

[00:48:49] JM: Yeah, absolutely. Okay. Well, George, thanks for coming on the show, and congrats on all the success.

[00:48:53] GF: Thanks very much.

[END]