EPISODE 1519

[INTRODUCTION]

[00:00:00] ANNOUNCER: This episode is hosted by Sean Falconer. Sean's been an academic founder and Googler. He has published works covering a wide range of topics from information visualization to quantum computing. Currently, Sean is Head of Developer Relations and Product Marketing at Skyflow, and host of the podcast Partiality Redacted, a podcast about privacy and security engineering.

[INTERVIEW]

[00:00:27] SF: Torsten, welcome to the show.

[00:00:29] TG: Thank you so much, Sean. Glad to be here.

[00:00:31] SF: Awesome. Yeah. I've been really looking forward to this conversation. You know, I have friends at Snowflake. It seems like a great company. I'm actually hosting a panel later this week with your community and developer relations lead. And even though it feels like it's been exploding for the last several years in terms of growth and brand recognition, it also, in a lot of ways – Being at Snowflake Summit back in June, in a lot of ways, feels like you're just getting started, which I'd love to hear your thoughts on. But before we get to all that, can you start off by introducing yourself and share a bit about your background for how you ended up where you are today?

[00:01:06] TG: Yeah, happy too. First of all, thanks for having me on today. Appreciate it. My name is Torsten Grabs. I'm the product manager here on the Snowflake product team. I cover a couple of areas. They include Snowpark, among others, and data lake, data engineering, and data science and machine learning.

I've been with Snowflake for about five and a half years. And I started working on a project, my first project was to bring Snowflake to Microsoft Azure as our second cloud provider. And the reason I joined Snowflake was I brought a little bit of experience about Microsoft and Azure

because I was working on one of the Azure teams. Prior to joining Snowflake, I've actually been with Microsoft in the broader kind of data platform business over at Microsoft for 10 plus years prior to joining Snowflake.

[00:01:54] SF: Amazing. That's a quite a journey. It sounds like you're covering a lot of stuff at Snowflake. I think anyone that's kind of working in the data space is going to have probably some familiarity and understanding of what Snowflake does and is. But if you're a frontend engineer or maybe you're working on specific technologies outside of the data cloud world, maybe not. Can you perhaps just start with a description of what Snowflake is? What it does? Why and how people actually use it?

[00:02:22] TG: Yeah. And maybe the best way to describe it these days is Snowflake is your data cloud. And it covers all your data types and all the workloads that you can think of. And maybe workloads is something that's the most approachable for folks here no matter whether you have a traditional data warehousing workload, whether you're a data engineer working on data pipelines, whether you want to bridge transactional with analytical workloads, whether you're in the data science or machine learning business, all of those are very data-intensive workloads. And that's what we are focusing on with Snowflake. And we're aiming to cover all of them really well.

[00:03:01] SF: Mm-hmm. Obviously, data storage processing is not a new idea, like ETL, extract, transform, load, has been around for a long time since like the 1970s. And various storage technologies have been around even longer. And the idea of moving data around, transforming it for analytics, reporting, isn't something new. What was it that made sort of Snowflake unique? And what product gap did it fulfill or a problem that it solved that one wasn't being satisfied by sort of the incumbent players in the market?

[00:03:30] TG: I think what really changed the game here for everyone was the public clouds that started to emerge and become popular. And that really shifted a lot of the fundamental assumptions that folks in the data space have been making so far. One of them was – And that had a lot of architectural consequences then for the products that were available back then, that you always bought compute and storage together.

And I think Snowflake was one of the first to do in a really disciplined way is to challenge that assumption and say, "Hey, in the public cloud, we can now actually buy, and provision, and scale compute independent of storage." And isn't that actually great because it allows us then to scale these two dimensions independently of each other?

And one particular workload, coming back to the workloads that we just talked about, some workloads might be very, very compute-intensive. For instance, if you think about a data science machine learning workload, that may at times be really, really compute intensive. May still go over a lot of data but require tons of compute. And wouldn't it be great if you actually can buy all of that compute without having to pay for a bunch of spindles that you're never going to use, right?

And I think Snowflake was one of the first to really explore that separation in a disciplined way. And it led to a number of architectural choices that the founders made, which then also are now paving the way for benefits for all of these other workloads as well, right? That you benefit from scaling this independently. And some other principles that we're closing out on the founders as well are on ease of use, simplicity, and avoiding knobs, removing limits wherever possible. I mean, those shine to the product experience in all various places.

[00:05:40] SF: That's amazing. Like, it in a lot of ways, this idea of separating compute and storage, like, retroactively seems like such a simple idea. But for a very long time, no one was really challenging that assumption. And it sounds like, in many ways, Snowflake, or the founders, took the sort of first principles approach. Maybe you can speak to where the idea originated from. But I imagine they had a lot of experience in the space and kind of recognize that the industry had this false assumption about kind of pairing those things together.

[00:06:09] TG: I don't necessarily think it was a false assumption up to the point when actually the public cloud emerged, when they started offering this as separate resources that you could buy separately, right? I, myself, worked on data warehousing appliances prior to joining Snowflake in my career. And it was, literally, the best that you could do is you buy off the shelf computer boxes and you just put one next to the other. You wire them up with a reasonably powerful network connection. And then you had scale out. But since you were buying individual

boxes that had a certain amount of compute and storage, you were always buying these together. It was state of the art.

And I think that also then heard a number of the incumbents, because the hardware that you ran on then reflected in some of the choices that you also made in your software. And those were not necessarily always beneficial. I think that was one of the key advantages for us with Snowflake, is that based on that separation, we also started to think about the software stack on top of storage in a very, very different way. And that created a lot of latitude for us, right?

For instance, the whole idea about workload isolation that you can actually bring up a separate compute cluster for, let's say, data ingestion separate from the analytics and reporting workload in a separate compute environment, but pointing to the same data in a shared storage place. That was very novel. And it was made possible, again, by the separation of compute and storage. And it had tremendous benefits because you were now able to scale on provision these independently.

If, for instance, people were coming in on Monday morning, you could actually provide additional capacity for your reporting cluster because everybody was pulling up their reports at 9am Monday morning, right? And you could cater to that, right? And also, you were avoiding noisy neighbor problems. If at some point you had a larger than usual amount of data to load into your warehouse, you could independently scale your data ingestion compute cluster and bring all that into the shared data space without having any impact whatsoever onto your BI reporting workloads, right? Those are just some examples here on how this actually changed the game and led to completely new and better customer experiences.

[00:08:44] SF: Right. It's almost like the idea of breaking apart the monolith, where you're creating this like separation of concerns. So, you can independently scale specific workloads or specific use cases based on the requirements of the service at a given time of day even.

One of the products that you mentioned that you oversee is Snowpark, which I believe was first introduced in 2021. Can you explain a little bit about what Snowpark is? And why is Snowpark exciting to the data engineering and machine learning communities?

[00:09:14] TG: Yeah, great question. The more we have started to embrace workloads, such as the ones that you mentioned; data engineering, data science, but also working more closely with developers actually building applications on top of Snowflake or with Snowflake, we very soon came to realize that just having SQL as the only language that Snowflake supports is not viable, it's not tenable. Because a lot of what happens in the industry in these workloads actually has different language preferences. And maybe data science is the prime example here where most of the community today is actually working in the Python ecosystem. And that's where a lot of the Innovation happens. And that's the language of choice for data science and machine learning practitioners, right?

In order to support these folks with their data processing needs, we had to think about adding additional languages to Snowflake and support them equally well as we have been running SQL as our first programming language. And that really led us to start thinking about extensibility more holistically. And that is really what Snowpark is about. It is about extensibility on two levels, if you want.

The first level is more kind of traditional database extensibility, if you want, by adding additional language runtimes to the core database engine to essentially take the compute engine for Snowflake. And that's kind of the first major infrastructure investment that we have made for Snowpark, is to build out an infrastructure on the server-side that allows us to hold most additional language runtimes in sandboxes which are highly secure so that you cannot do anything nefarious. Because you might be processing mission-critical data in there.

And now, this infrastructure then has allowed us to first slot in Java virtual machines as the first language extension that we provided there. And more recently, the Python runtime as well. But the infrastructure is general enough so that we could actually, in the future, if customers asked us to do that, we could add additional language runtimes into the mix here. That is kind of the bottom layer, the server layer, if you want.

On top of that, we have also started to think about more modern, more fluent developer experiences that resonate concepts and abstractions that resonate with modern software development. And something that has become state of the art is very popular particularly in the context of data processing is what people call data frames. And they've become popular through

the likes of Pandas or Spark. And particularly for large-scale data processing, they are a very familiar programming concept for data engineers or for data scientists.

In addition to the server-side extensions that I just explained, we also started investing into client-side SDK experiences to provide language integration into these host languages and offer the familiar data frame abstraction for developers. Now, what is neat is those two layers, they don't act in isolation. They're actually tightly connected. And what happens is that you can write a piece of custom code on the client side, invoke it within one of the data frame transformations that you're working on. And when you execute this, we will do the heavy lifting of figuring out what's the piece of custom code that you've written on the client side? Pushing that down to the server-side, registering that as a piece of custom code in the server-side language extensions. And then, essentially, composing that with the overall data flow program that's executing on the server side. And that is actually leaning on the same operators that we're also using for SQL processing.

But then in those places where it needs to invoke the custom code, it will escape into your piece of custom code in the language runtime in those sandboxes and then do whatever you've told your custom code to do. That's really important, because for all those operations in the data frame program that you've written on the client side, everything that translates into the regular Snowflake operators, that will actually execute and process at regular Snowflake speed. Although, you're using a very, very different programming surface with the data frame APIs.

And that, in a nutshell, is what Snowpark is about. And it has allowed us now to add support for languages such as Java and Scala, which are based on the Java virtual machine. And then, also, Python, with the Python runtime both with data frame extractions on the client side in addition to the server set extensions.

[00:14:14] SF: Yeah. It sounds like, essentially, someone is writing custom code and then running it directly within Snowflake within the sandbox environments. I guess, how does that differ from a traditional data engineering process and the tooling that they would be used to using? And what is the advantage that working with Snowpark essentially running those loads directly within Snowflake provide the data engineers of the world?

[00:14:42] TG: Yeah. I think one of the immediate benefits is that you don't have to worry about loading your data into a separate processing environment. Everything stays in the same familiar place, which is Snowflake, right? You don't have to extract your data into, let's say, a VM that hosts your data integration or ETL tool and does the processing there. And then you have to worry about pushing the results of the transformation back in. And how do you make sure that there's no data left on the virtual machine that hosts your tools, right?

All of this just happens within Snowflake in exactly the same compute infrastructure, the same compute clusters with the same governance and security concepts and primitives that you're already familiar with. And I think that's a big advantage, first of all.

In addition to that, it's much easier to scale and even scale elastically, because you are not in the business of managing individual VMS and managing individual capacity for your data pipeline. You can lean on familiar Snowflake scaling and performance concepts, right? Our virtual warehouse concept is where Snowpark programs run, right?

It's the same compute infrastructure, the same t-shirt sizes for your virtual warehouse that you can also use to speed up your data pipeline if you're a data engineer, or your data science work if you are a malpractician.

[00:16:15] SF: Mm-hmm. With the support now of Python within Snowpark, I'm sure that is a big win for the data scientists of the world that are looking to do machine learning models and so on with Snowflake. But let's say that I have data in Snowflake and then I had built a machine learning model through some traditional methods. How do I actually go about moving that model over to using Snowpark so I can take advantage of some of these things?

[00:16:40] TG: The simplest way to think about this is if you have already a readily trained machine learning model, you probably know which function you invoke on that model to invoke inference or prediction on that machine learning model.

The easiest probably is to just refer to that function, invoke that function, from a Snowpark data frame program and just inject that somewhere wherever appropriate into your data frame API calls that you could, for instance, be – that you have a protection list that you apply to the

underlying raw data that you want to score or apply the predictions to. All that is readily done through a data frame API call or a set of API calls.

And then one of these API calls you just call out into the inference function of your machine learning model. Obviously, you will have that imported into your Snowpark client-side code. And once that data frame program that then executes and gets compiled down into the compute that runs on Snowflake, we'll then push the machine learning model into Snowflake on the server-side and then host the model in Snowpark in the server-side extensions and wherever we need to escape into that and treat it as a piece of custom code.

[00:18:01] SF: I see. And then Snowflake also now supports the native application framework, which allows one to build essentially an application on Snowflake's data cloud and then distribute it in the Snowflake marketplace. And it, again, sounds like it's another sort of system for bringing the application closer to the data rather than sort of the other way around.

As a developer, what can I do with these applications? From the marketplace, as well as like developing things for the marketplace?

[00:18:29] TG: What I would call out first there is that we started the marketplace as a data marketplace. But we soon came to realize that most of the value that you get out of your data actually comes out of it when you apply a piece of code to the data. And oftentimes, the data and the corresponding code go hand in hand. And we wanted to help our customers with that juxtaposition, if you want, between data and code by giving customers and developers the ability to publish, and distribute, and eventually monetize their code through our marketplace. It's no longer just a data-only marketplace. It's a marketplace for data and the code that knows how to derive value from your data.

And that I think is making it easier now for developers on Snowflake to actually bring their code to prospects, to customers. Give them the ability to easily try out the code, deploy that into the customers', the consumers' Snowflake account. And then if the customer is happy, just have them bill and monetize the code and the application through the same infrastructure that we've built out for the data marketplace.

[00:19:46] SF: Yeah. I imagine, historically, for someone building sort of data-intensive applications for customers, the process of actually sharing that application with the customer is probably quite a headache. You're going to run into the, "But it works on my machine," type of problems. And it sounds like by leveraging the marketplace, you are eliminating that issue. Because, essentially, you have cloud development, cloud deployment, an easy way of sharing the code and having someone install it directly against their own data. Is that right?

[00:20:14] TG: That is correct. Right. And we're currently running on the three major cloud providers across a number of different regions on each of them. And that already helps us getting the application code for a Snowflake native application distributed across the globe into your Snowflake cloud provider and your Snowflake cloud provider region, right? The code will be readily available there. You don't have to worry about like where do I download this from? You can actually trust the download location that I've been given, right? It will help also with upgrades and version management.

I think, really, the inspiration here comes from the mobile phone marketplaces and app stores, if you want, and provide a similarly seamless experience now for a completely different domain, which is data processing and data applications.

[00:21:05] SF: Are you seeing people, I guess in the data engineering, data science, space that are essentially solving problems using this method that were maybe not possible to solve before or just difficult to do?

[00:21:18] TG: I mean, one example, that comes to mind is, for example, Capital One, one of our really large customers. And they've built a number of tools that originally came from their own work with the Snowflake internally. But they found them valuable. And others have also shared that sentiment and said, "Hey, I would actually love to use that for my Snowflake account and for myself Snowflake work. Can you provide me the code for that?" And that essentially let them down the native application's path, where their slingshot project now is using Snowflake native applications to distribute something that started as an in-house project at Capital One, but now is essentially distributing that into the data cloud to all the Snowflake's customers. And gives them the ability to benefit from the same capabilities that Capital One is using in-house to manage their Snowflake so that those customers can also use the same tools and techniques.

[00:22:14] SF: Okay. Yeah, that's very cool. I think it makes a lot of sense to move applications closer to the data. I think computer scientists have kind of been moving stuff closer together to optimize for speed for a long time. But in terms of data space, where maybe people have always thought about sort of pulling data into their applications, have you encountered like an educational hurdle that needs to be overcome to teach the market that this is possible. And that this paradigm of moving applications closer to data is like the right model?

[00:22:45] TG: I think that the industry is starting to understand this better. And I think a key driver there is that people are realizing that the mental overhead of managing a large fractured stack with a lot of very different components that all deal with certain slices of the data processing problem can quickly become a nightmare if you don't manage it really, really well. And there's obviously just concept count, because every slice, if you have a convoluted data stack like that, will trade pieces differently. So, you'll have different teams with different skills and expertise that you need to build around that.

And then the other big driver these days is around security and governance, right? If your data moves literally across these different places in your stack, you need to make sure that you understand all them well. And you know at any point in time where a certain piece of data sits. Just mention things like GDPR, which just have "additional pressure" on this where you need to make sure that you understand where all your copies of a certain piece of data are. And that becomes more tricky if you have more components that interact in interesting ways, right? Simplicity helps big time on this run.

[00:24:05] SF: Right. Yeah, the centralization of information is going to help a lot with essentially the PII sprawl that a lot of companies are facing today, where their data is in like 56 different locations and they have no idea where and sort of what they're storing.

I want to talk a little bit about UniStore. UniStore allows organizations to store and use transactional and analytical data together in a single platform. First, what's the difference between these two forms of data?

[00:24:33] TG: The data is not necessarily that different. You'll often have situations where the transactional data is actually the origin, the source, of data that later ends up in, let's say, your analytics systems. Think about like sales reports that you do. The sales report is probably a piece of analytics that you want to run. But the actual sales are tracked somewhere in a transactional system. The traditional way how these connected were through a data pipeline through an ETL process that essentially extracted the data out of your transactional system and then dumped it into your analytics system.

And the storage organization, the physical layout, of the storage was typically quite different between them. Transactional systems usually prefer a role-oriented representation of the data at the storage layer. And the reason for that was that the workload pattern was typically such that only a few rows need to be accessed at a point in time. A transaction was typically just touching a couple handful of rows. And a row-oriented storage layout made that very, very efficient. So, response times, latencies were low because of that.

Now, conversely, on the analytical side, for your sales report, you typically want to touch a large number of rows, right? Your data set that you want to process for a single report or in a single transaction is dramatically larger than what you would process in a transactional system.

And oftentimes, you didn't need all of the values that you had present in a row. You were maybe just interested in the customer IDs, maybe the customer region and the actual sales numbers, but a lot of the other data was not relevant. And that then led to the columnar representation from a storage representation that allowed us to exclude those columns that were not relevant for the particular report and focus on those that were really needed to answer the reporting question. And that layout then led to large speedups for reporting workloads that were plowing through a very, very large number of rows but only needed slices of the columns that you had present in in your transactional table.

And it's that difference in storage layout and optimizations that were applied to cater to these different workloads, which led this to be broken apart into historically separate systems. But we were just talking about the complexity in your data stack and how easy it is to end up with redundant copies where the same data sits in multiple places and maybe sometimes you not being aware of that. This is an example of exactly that situation. Because you'll have the same

customer data likely sitting in your transactional system and maybe a slightly different representation, but in essence, the same data sitting in your analytics system and just with different storage layouts to cater to the different needs of the different workloads, right?

And obviously, there's a lot to be gained, and folks have worked on this for many years, is can you bring these two closer? And can you eventually actually eliminate them being separate systems? And can you fold them into one single system that works equally well for your transactional workloads and for your analytical workloads? And that's really what UniStore or and hybrid tables are about, is, essentially, you bring these two together and creatively bridging between different storage layouts based on the workloads that we observe, right?

And again, we started with independence of compute and storage. Again, this helps us here, is that we can actually work on that compute layer and then point to different physical storage representations, which logically just looks like the same table. And by virtue of doing that, literally bring these two, the transactional and the analytically side together in a single place.

[00:28:34] SF: Is that how, essentially, you're bringing the two worlds of like OLTP and OLAP together, is that you have kind of one common interface as a developer to interact with these? But behind the scenes, there's sort of these two maybe somewhat complementary ways of representing data?

[00:28:50] TG: That's exactly right. And we're trying to be smart and mindful about what storage representation we apply to which thesis of your data at what point in time.

[00:29:00] SF: Right. So then, I guess using UniStore then, is a developer essentially able to build both a transactional application on top of Snowflake as well as using it for traditional sort of analytics workloads?

[00:29:15] TG: That is correct, right? And maybe even in the limit, you could run those on the same table. And if you're, for whatever reason, choosing to still keep transactional data in one table and analytics data in a separate table, they're all regular Snowflake tables. And if at some point you need to run a SQL statement across both of these tables, you can totally do that. It's in the same system and it supports the same SQL language on top. You don't have to first stitch

together two completely separate independent systems like that to do in the past if you had a requirement.

[00:29:49] SF: Mm-hmm. And then how does this technology sort of differ from something like Oracle's Converge database or MySQL's HeatWave database cloud service?

[00:29:58] TG: Yeah, I think to a large extent, we're just driven by our customers here. And some of the kind of core Snowflake advantages is probably what I would call out here. The cloud native architecture is probably something that differentiates us here in this space, and that should give us better elasticity and better scalability.

And also, I think the ease of use is another key design point for us that we're going to continue to pay great attention to even going forward as we're embracing more of the transactional workloads.

And then last but not least, I would maybe call out also the Snowflake consumption model, which is the billing model that we use. And it's utilization-based. So, you don't actually have to worry about licenses. It's essentially what compute are you using for your workload? And that's what you're paying in the end.

[00:30:48] SF: Okay. Yeah. I want to talk a little bit about sort of your past six years, almost six years, at Snowflake. And also, kind of looking towards the future in the next five to ten years in this space. You've been there. Obviously, Snowflake's probably changed a lot in the time that you've been there. When you look back on your journey there, what are some of the things that kind of stand out in terms of how the company and product has changed throughout your journey there?

[00:31:14] TG: I mean, first of all, it has been an amazing journey. When I joined the company, we were around 200 people. And right, we're somewhere around 5000 people, a couple thousand people, right? And the number of customers, I mean, has grown dramatically during that time. So, that has been awesome to experience firsthand.

What was also great to see is the different ways how customers have used our product for new problems, and we're able to creatively solve these problems that has been amazing to see again and again. And it keeps happening, right?

As we are wrapping our heads around new workloads, and we have just added UniStore and Security to the mix of workloads that they support, right? Those are just examples of the leg room that we have with some of the core fundamentals of Snowflake's architecture that are still playing out.

And I'm very excited to see us continue on that Journey and see what additional new workloads we're going to add over time. I'm very much looking forward to that. And I personally just think about some of the workloads that I work on. Think about some of the opportunities that we have to really change things dramatically. I think that's just amazing. That thinking about, for instance, the machine learning space, data science, one very common problem there is to actually help machine learning models getting productized and rolled out to the end user. That has been historically a tough problem, and it continues to be challenging.

Now one of the recent acquisitions that we make with the Streamlit and in the context of native applications, I believe, is actually going to help move the needle quite a bit on this front. If you think about a machine learning model getting prepared by a data scientist and then wrapped as a native application, Snowflake native application, that uses Streamlit as the way how that machine learning model interacts with the end user, that is much more meaningful, much more powerful than being stuck with, let's say, a Jupyter Notebook in a Docker container that the machine learning learning practitioner has to somehow hand off to the end user.

In many organizations, the end users will not know how to work with a Docker container in a Jupyter Notebook in order to leverage the machine learning model, which might be phenomenal, right? But it might be much more approachable to have an actual user interface with visualizations, which is exactly what Streamlit can provide, right? And just seeing that, bring machine learning and machine learning models to the marketplace and to the end users, is something that I'm really looking forward to. And that's one of the areas I'm personally very close to and also very, very excited about, as you can see.

[00:34:12] SF: Yeah. There's a lot of development, I think, in sort of the data engineering and data science like tooling world. But it seems like there's still probably a lot of work that needs to go into productionizing a lot of stuff to get it to the maturity level of, say, traditional application development. Do you think that we still have a long way to go in terms of supporting things in the data engineering space? Like, testing, and observability, and logging, and rollbacks and so on?

[00:34:41] TG: Absolutely. Yes, there's a whole wealth of uncharted territory. And we are aware of that as well. And that's also part of the excitement, because we're not going to run out of work anytime soon.

[00:34:51] SF: Yeah, that's great. Along the line, there's also been this tremendous growth, I think, in the fields of data science, data engineering, data analytics. You hear data is the new oil. There was data science is the job of the 21st century. And you see a lot of these roles start in sort of the big tech companies. And now you're seeing startups like hiring people in these roles much, much earlier. Why do you think there's been so much growth and investment here? And where do you see these roles sort of evolving and going in the next five to ten years?

[00:35:25] TG: I think that, actually, we're going to see just writing code and developing become a core skill across all of these roles. I mean, both data engineering and data science will require you to write Scala, Java or Python code. And that is not going to change anytime soon. I think that's actually going to get amplified.

For any of these roles, you need to start embracing software development and best practices for developers as part of your profession as well. Things like CI/CD, source code version, GIT, all of those are essential tools of your trade as well. And they're not just limited to the more traditional software developer or programmer.

[00:36:13] SF: Right. And then what do you think – looking ahead, it sounds like you're still really excited about what you're doing, which is great to see after spending five to six years at a startup that's really exploded. But what do you see is the future of sort of the data cloud in the next five to ten years? Are there specific things that you're really excited about?

[00:36:32] TG: I think it's really seeing customers and our partners build these collaborative end-to-end data value chains across the data cloud I think is it's kind of my mental model. And I think we're just at the very beginning of exploring the possibilities of that. Some of the basic infrastructure is what we've just started to work on with native applications, with the marketplace, with data sharing across cloud providers and regions. We're just a few years into that. And I think we haven't really tested out the waters and seen all of the possibilities, right? And that's really what I think is going to change how companies, how businesses, how organizations, and maybe even individuals at some point are going to interact. It's going to change substantially. And data is going to be in the middle of all of it together with the code that allows people to actually derive meaning and value from the data that they interact with. And providing the foundation for that, being the platform for that, is what we're all about here at Snowflake.

[00:37:41] SF: Yeah. This kind of goes back to what I was saying in the beginning, where I think it sounds like there's been incredible success, obviously, with Snowflake, and also just the whole space of data engineering, data science and so on. But a lot of the investments that you're now sort of in a privileged position to make, you're still early days. And you're just kind of scratching the surface of what can actually be done with this massive amount of information that people are sort of collecting and starting to analyze.

[00:38:08] TG: Exactly. Yeah.

[00:38:08] SF: Is there anything else you'd like our audience to know?

[00:38:11] TG: No. I think we covered lots of ground. I'm really thankful for the conversation. I really appreciate it. And yeah, looking forward to the next five to ten years. And I'm very curious to see what's going to happen and what customers are going to come up with and build.

[00:38:25] SF: Yeah, that's fantastic. Thank you, Torsten, for joining the show and spending so much time with us. It's great to hear about your journey going from, essentially, 200 people at Snowflake, to now thousands of people. And it sounds like you still have a lot of passion and excitement for what you're doing. I'm sure people will find a lot of value in this conversation. So, thank you again for joining us.

[00:38:44] TG: Thanks for having me on. I appreciate it.