EPISODE 693

[INTRODUCTION]

[00:00:00] JM: A data warehouse provides fast access to large datasets for analytics, data science and dashboards. A data warehouse differs from a transactional database because you often do not need to update specific records. Because of the read only nature of these access patterns and the high-volumes of data being queried from a data warehouse, the design of a data warehouse is very different than a transactional database.

With a transactional database, such as MySQL or MongoDB, it's important to have consistency guarantees. For example, consider a transactional database that serves as the backend for banking applications. If multiple frontend servers are hitting the transactional database to withdraw money, you need the records to be quickly updated. You need to avoid race conditions so that two servers cannot withdraw the entire back account balance simultaneously from different locations. In contrast to transactional databases, a data warehouse is often used to process a query that emphases a big dataset.

For example, Netflix might want to answer the question; how many users that watched House of Cards also watched Black Mirror in the last 5 days? Netflix has a lot of users, so they will want to be accessing those user records in a way that lets them scan through all the records quickly.

Christian Kleinerman is the VP of product at Snowflake Computing. Snowflake's main product is a cloud data warehouse. In today's show we talk about the difference between a data warehouse, a data lake and a transactional database, and the process of moving datasets between them, which is often known as ETL.

This show continuous our series on data engineering and data platforms. As companies accumulate more and more data, the complexity of managing that data and taking full advantage of it is escalating. Christian gives his perspective on these changing trends of data warehousing and data management and describes the plans for Snowflake to evolve as a business.

Before we get started, I want to mention that Software Engineering Daily is looking for sponsors. If you're interested in reaching over 50,000 developers, you can go to softwareengineeringdaily.com/sponsor to find out more, and you can send us a message there. We'd love to hear from you. If you are an engineer working at a company that is marketing to developers or hiring developers, if you tell your marketing department or your recruiting department about us, that's one way to help us out. You can just tell them about softwareengineeringdaily.com/sponsor.

With that, let's get on with the show.

[SPONSOR MESSAGE]

[00:02:48] JM: I have learned a ton from QCon. QCon San Francisco takes place this year, November 5th through 9th, 2018, and I will be going for my fourth year in a row. I always love going and seeing the talks, and in between the talks I hang out and eat some mixed nuts, chat with other engineering leadership about the latest talks and stuff that they're seeing, the 50 different stream processing systems that we're seeing, the different databases we're seeing, and QCon is a place where senior software developers and team leaders and managers and senior leaders, they all gather together, and you have fantastic conversations. You have fantastic presentations, and it's extremely high quality. Its technical. A lot of it is also cultural and about management, and you can get $100 by using the code SED100.

QCon is a great learning experience. The presentations this year include 18 editorial tracks with more than 140 speakers from places like Uber, Google, Dropbox, Slack, Twitter. They are curated high-quality talks, and you can get $100 off if you use code SED100 at checkout for your ticket. Again, QCon San Francisco takes place November 5th through 9th, 2018, and the conference is the 4th through 7th, November 8th through 9th are the workshops. You can go to qconsf.com to find out more.

Thank you to QCon. If you are going to buy a ticket, please use code SED100, and we really appreciate the sponsorship of QCon.

[INTERVIEW]

[00:04:46] JM: Christian Kleinerman, you are VP of product at Snowflake Computing. Welcome to Software Engineering Daily.

[00:04:51] CK: Thank you for having me.

[00:04:52] JM: The flagship product of Snowflake is a data warehouse, and I'd like us to talk through what a data warehouse is and how it fits into a data platform to kick off our conversation. How do you define that term that data warehouse?

[00:05:10] CK: Yes. I think a data warehouse is a system that serves as a repository for data. Usually the data comes from multiple sources. It doesn't have to come from multiple sources. It can be one or multiple. The primary use, primary purpose is to enable reporting and analytics basically to get value to their insights to get understanding from that data. That is like the shorter description on what data warehousing is.

[00:05:39] JM: has the number of use cases expanded from reporting and analytics to things like machine learning?

[00:05:47] CK: For sure. For sure, I think it's a continuum. I don't know that it went from – There's no machine learning to all of a sudden we have a new use case. I think data warehouses are used for analytics in general, and the types of analytics evolve overtime. We used to have simple aggregates where you could just count, say, the total sales with total number of events that happened some period of time. Then you ended up with more interesting analytical questions where you could say, "Show me a moving average or show me a rolling sum based on some window." Then as you get into linear regressions and more sophisticated types of analytics, it blends into the predictive capability of machine learning, which his obviously one of the hardest trends today in the analytics world for today, and it's a big complement to expansion to what traditionally data warehouses have been.

[00:06:42] JM: You're describing some use cases such as analytics and machine learning. These use cases often involve large datasets. How does the requirements of a data warehouse and the access patterns and the engineering of a data warehouse, how does it compare to what we commonly refer to as a database?

[00:07:03] CK: I think database is the more collective term for a system that helps you store and manage data. I think over many years, there was no distinction on types of databases and storing data was storing data and that was it. Over time, I think this goes back 15, 20 years back, the databases started to get specialized based on the purpose and use case that was needed. I think, today, the market is split primarily on the operational database on one side and the data warehouse on the other side. The operational one being a database that is used for transactional activity operational activity, say, if you are building an application and you want to record every single interaction that the user is doing and show quick back and forth on edit a record, modify a record, display a record, that's more what a traditional operational database does. A data warehouse is a database in the context of let's get larger volumes of data. Let's get analytics. Let's get insights out of the data. So they have some commonalities and you'll find use cases that blend between those two worlds, but primarily the data warehouse is the use case we have a data repository for analytics.

[00:08:25] JM: There was this notorious paper that came out, it might have been 10 years ago, something like that, from Michael Stonebraker about the age of one-size-fits-all being over. I think you described some of the characteristics of what we have seen. One-size-fits-all meaning you need more than one data storage system at this point, because you have different applications that have different access requirements. You might have a search application that needs to access a database that's based around search. You might have a transactional data storage system, like if I'm requesting an Uber ride, then I need certain requirements. I need the database to be really consistent for my session of the Uber ride. But then if you have machine learning applications or batch or analytical applications that you're going to run large queries over collections of Uber rides, the access patterns for the analytical jobs are going to be very different than the user who's standing by the side of the street trying to get an Uber ride.

[00:09:42] CK: That's right. I think about my background. I was in Microsoft in the SQL server team for many years, and back then it was true, as you said, a single database system, and it was meant to support all types of database activity. The big realization over the years is that if you knew the use case, if you knew that you were going to record the Uber ride request and find the driver, etc., and separate that use case from the, “Show me a total number of trips across all Uber riders.” Once you understand those use case, you realize that you will build the underlying platform in different ways, and that is effectively what Stonebraker’s paper alludes to and is effectively what the industry is, where you do have a differentiation of use cases. As a platform builder, when you understand the use cases better and you can design for each individual, you can provide a better answer and a better solution for each of these cases.

[00:10:41] JM: There’s a term called ETL, extract, transfer and load, and this term confused me for a long time, but I think I understand it now. Is ETL a way of describing how you are getting data from one place to another? For example, you have this operational database, the database that’s serving my queries of requesting of an Uber ride and managing my session while I’m riding around in Uber. That’s one database. That’s often times a source of a lot of data. Then you have to run an ETL job to get that data into a data warehouse, or a data lake, or some other kind of data system.

[00:11:23] CK: That’s right. I think it is a traditional form of preparing data to go from whatever its sourced system is to a system for analysis. The ETL, the acronym helps think through the sequence of steps. The first one is extract. Usually, technologies that provide ETL have a number of connectors that can go through a variety of systems and connect to those systems and query and get data. That’s the extract piece.

The T goes for transform, which most often all of these data sources do not have a common representation of a data, say, the name that I use for you one system may not be exactly the same name that I have in another system, or a physical address may be different. So a transformation usually alludes to data preparation, data cleansing, making sure that the data will make sense once I analyze it. The last piece, the loading is how do you get into the data warehouse or into the analytics system, and the operation is common referred as, “I am going to load the data,” and then you can start asking questions and getting value and getting insights out of the data.

The traditional flow, I don’t know – Have you heard about the ELT, which is a slightly reversal on the operations?

[00:12:47] JM: No. Tell me about that.

[00:12:48] CK: Traditionally, the flow was ETL because the cycles, the CPU cycles and the resources on the data warehouse on the analytic system were limited and were often more expensive. But with the event of parallel computing and more commodity hardware, etc., a common pattern, and we see this a lot from our customers today, is to do ELT, which is you extract the data, then you load it into a raw form into the analytics system, the data warehouse. Within the data warehouse, then you transform it and do the preparation and the cleansing that I described. It has been a significant shift into how data gets prepared for analytics in the last few years. Even though there is still obviously many customers doing ETL and there's room for it, we see a big trend towards ELT type of preparation of data.

[00:13:50] JM: So the advantage of ELT versus ETL is what? Can you calcify what exactly is the advantage if you switch from ETL to ELT?

[00:14:01] CK: Yeah. It's a good question. Most often it's just the need to deal with fewer systems. It could have been that you had one system to do the transformations. When I say one system, it could be a platform and a specific programming language and a specific form of source control. You had to know how to control or manage these transformations.

Then once you have it into the data warehouse, you end up having, again, some set of scripts and governance, etc. In the ELT world, you end up having all of the operations consolidated in your data warehouse or your analytic system. From a manageability and governance, respectively in maybe simpler workflow, and I think that's one of the main advantages.

[00:14:43] JM: As we continue to fill out the different components of a modern data platform, I'd like us to touch on the data lake as well. Eventually, we will get to talking more specifically about Snowflake, where you work. But I want to give a lay of the land for people who are less familiar with this area. What is a data lake and how does that compare to these other components of the data platform, like the data warehouse and the transactional database which we've talked about?

[00:15:13] CK: Okay. Yes, I'll talk about them, and for sure when we get to the Snowflake part of the conversation, it's going to be interesting, because some of the concepts start to blur based on data and technology capability. But a high level, the notion of a data lake is sort of like born from a set of unmet needs from traditional analytic systems. Some of the promise of data lake – Data lake at a high level is similar, it's a repository where you can store data and prepare the data for analysis. The reason why it's different is on the data lake side, there's this promise that it will be able to accommodate all your data, and that's a big departure from how traditionally data warehouses were seen where you had to highly select and curate and subset the data. Where if the data lake says, "It doesn't matter what data you have. You can put all of your data in the data lake and have it in a single place," which is a compelling capability."

When it says all data, data lakes also refer to both structured relational data as well as semi-structure data or unstructured data. Again, traditionally, relational data warehouses didn't have support for anything but relational data. So data lake again was a compelling answer, or is a compelling concept, because you don't have to now have different answers and different systems depending on the type of data.

The other notion on a data lake is that you store the data in its original form, similar to what I described in the ELT workflow, where you just load it and then later on you decide to transform it, the data lake says something very similar, you just give me the data in its raw form, later on you can decide what you want to transform or adapt the data to facilitate analysis. That was the last benefit of the data lake, which is if you compare it with how data warehouses traditionally used to work where you have to know what questions you want to ask, what SQL queries you want to write, and that's what inform what data you got into your data warehouse. The value proposition of the data lake as a model or a system is get all the data in its raw form. You do not need to know all of the queries and questions you want to ask. You can define those after the fact and go and then modify or transform the data as needed to answer those questions.

The summary of the data lake is a similar repository or a pattern for design that will accommodate all data regardless of structure and structural value and will let you keep it in this original form and transform it only later on as the questions or the SQL queries or the queries you need to run are better understood. I'll pause there, but obviously all of these is an artifact of what traditional data warehouse technology enabled, which a system like Snowflake would change.

[SPONSOR MESSAGE]

[00:18:30] JM: OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CI/CD built-in monitoring and health management, and scalability. With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure as well as your own on-prim hardware.

OpenShift from Red Hat gives you Kubernetes without the complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

OpenShift is open source technology built to enable everyone to launch their big ideas. Whether you're an engineer at a large enterprise, or a developer getting your startup off the ground, you can check out OpenShift from Red Hat by going to softwareengineeringdaily.com/redhat. That's softwareengineeringdaily.com/redhat.

I remember the earliest shows I did about Kubernetes and trying to understand its potential and what it was for, and I remember people saying that this is a platform for building platforms. So Kubernetes was not meant to be used from raw Kubernetes to have a platform as a service. It was meant as a lower level infrastructure piece to build platforms as a service on top of, which is why OpenShift came into manifestation.

So you could check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW CONTINUED]

[00:20:38] JM: We're in the midst of a lot of changes to what the data platform looks like, and I've stumbled into doing a bunch of shows about the world of the data platform. There was an episode we did recently that really calcified some of these questions that I guess a lot of different enterprises are asking, because Uber has a great engineering team and I've realized that even Uber is struggling with a lot of these issues, but the Uber data platform episode that we did, it was discussing this two-year project that it took over two years to build this system that took transactional data, the ride sharing data and did ETL jobs from the transactional data into the data lake, which I think was on S3, and they had to version the data in S3, I believe. Well, I guess S3 takes care of versioning for you to some degree, but they had their own schema list system because there's a difference between parts of the Uber data that is schematized, versus parts that are unstructured. Then they have this entire set of systems that sit on top of the data lake and pull data from the data lake into different kinds of data warehousing, or search systems, or machine learning systems, or stream processing systems, and it's a fast-moving space and it doesn't seem like anything is set in stone.

Then I look at Uber as an example, I'm sure an insurance company with as much data as Uber has the same problems that Uber has, and yet they're an insurance company. They don't have as many engineers as Uber has. It will be interesting to see – I mean, this is where a company like Snowflake comes in, of course, because this is the kind of customer that you might be selling to, is perhaps an insurance company that is having these problems and you can come to them and say, "Hey, we can help you figure this out, because we have a data platform for you. Can you tell me about some of the modern problems of the "data platform" that you see at a large enterprise, like an insurance company for example?

[00:23:02] CK: Yeah, sure. I think your premises is completely spot on. We do business with companies on every single industry and vertical, small companies, medium companies, large companies, and I think the problems are very, very common across all of them. Most interesting to me is that a good number of the problems that companies deal with are all about working around limitations on existing technology. I'll go to some of the examples and you'll see how in reality it's technology that is sort of getting in the way of having a smoother experience and focusing on getting value out of the data as supposed to wrangling infrastructure.

A common problem that I see across the spectrum is the notion of data silos, which is companies have different systems and different repositories for different types of data. They may have – I think we talked a little about structured relational data on one side, versus semi-structured and structured on the other. But they may also have the HR data in one system and the marketing data in another system and the sales data in another system.

Pretty much every company that I talk to and even being part of, they're trying to get comprehensive views of their customers, their products, their services, their activity, etc., and all of these silos get in the way. Then if you start the dialogue with any of these companies on, "So why do we have all these silos?" The conversation often shifts to, "Oh, because X, Y, Z system didn't have enough bandwidth, so we created a shard of the data, or because of latency reasons we bought the company and had [inaudible 00:24:55] technology and we don't have an easy way to integrate it, and the list goes on and on.

But most of the reasons are the technology is at fault, the technology is getting in the way and is leading to silos. That's one of the challenges that I see across the board. Another one is around access to data to a broad set of users. So data is seen in many companies as, "Oh! That's what the analyst do," and if your title doesn't say analyst, then you're not part of it.

Obviously, we're shirting into an era where data is a fundamental component into informing decisions, informing products and services, designs and development, and it should be accessible and available to everyone. Then you start asking, "Well, why does everyone not have access to the data or why can't everyone just go slice – I don't know, sales by region or by category or product, and then you run into, again, technology, "Oh, this system cannot deal with all of these users." We cannot support that many concurrent users, or they don't have the expertise to write proper queries without hurting other people on the system. It's a little bit surprising to me, which is, "Okay, some companies are ready to go and embrace data for everyone." But, again, the technology makes it hard such that either they're worried that it might affect the performance of executive dashboards or regular systems, or they simply don't have the technology that has the compute cycles and the ability to scale to the [inaudible 00:26:39].

The last piece that I hear pretty much every company wrestling with is around tuning the system in order to get fast insights. A concept that I spent a lot of time thinking through is a lot of time to insight, which is how quickly will you be able to get value out of some data? I mentioned [inaudible 00:27:03] entirely from when an event is produced to when you got an answer, when you a got dashboard that reflects the impact of that event.

Again, from where the industry is going or where the data as a strategic differentiator is going, lower time to insight is desired by every single organization, and today there's, again, technology getting in the way and tuning up technology and turning up like crazy that's sort of prevent focus on what is the right metric, what is the right dimension, what is the right question to ask as supposed to do I have the right settings to get my query to perform?

At a high level, data silos, ability to get data to everyone, which has a concurrency element and also the time to insight and performance and simplicity are general policies across every single industry. Of course, I am eager to get to the Snowflake part of the conversation, because that is what is compelling and what's motivating to me about Snowflake, but in reality, stat of the art technology has had people jump in to all these hoops.

[00:28:11] JM: Yeah. Well, just to give a little bit more credence to what you're saying, we've used the example on this show a number of times of the VP of marketing at a company. So the VP of marketing probably doesn't know how to do a complex Map Reduce query, and probably may not even know what the right semantics are for SQL, and yet there's tons of money at the disposal of a VP of marketing. The way that things should work is the VP of marketing should be able to say, "Computer, how much money did we spend on billboards this year and how impactful was it? What kind of sales did those billboards drive?"

In order to get an answer to questions like that, the VP of marketing has to ask the data scientist, who has to ask the data engineer, who has to ask the infrastructure engineer. The infrastructure engineer finds out where a dataset is. The data engineer figures out how to configure it. The data scientist figures out how to query it. You just have this big stream of people that's really troublesome, and this exists at much lower levels of the economy too. There are data-driven questions that an Uber driver wishes they could ask of the data around Uber. They would help Uber.

If an Uber driver could say, “When are the times when I should be driving around?” That simple question, or where should I go? I know that Uber to some extent recommends that kind of behavior to the drivers, but I’m sure it could be made better. I’m sure it could be made more interactive. So just to motivate the conversation a little bit further and the degree of problems, the degree of opportunity that could be solved.

So as we make our way to talking about Snowflake, can you give a brief history of the data warehouse market and maybe the founding story of Snowflake? What drove Snowflake to be created and why there was an opportunity for a new kind of data warehouse?

[00:30:26] CK: History of data warehousing or background. I think we’ve touched a little bit already in the beginning there was a database workload without much differentiation on analytics versus operational overtime, and this is like roughly maybe 20, 20+ years ago, the notion of a database for analytics is born.

Also, the data model starts too to evolve, where instead of arranging my tables and the data in my tables in an optimal way for my transactional application, then there’s models to how do you optimize it for faster analytics, and you’ll hear about the dimensional data model versus the normalized data model and it’s almost a religious topic for some of the people in the industry. What you see is an evolution towards specialization towards systems that will help get insights out of data and do analytics on a data.

The most interesting trend or historic trend, and I was a big part of it over at Microsoft, or I was riding the trend, was the notion you had. Primarily, data warehouses were a single server or a single machine software solution, and that’s what most of us in the industry differ for many years when I say, “I was at Microsoft as part of the SQL server team, but that’s what Oracle was doing.” That’s what most systems were doing.

Then you see the growth of parallel systems, which is more than one server, more than one machine operating as a single system. The complexity of configuring and managing and tuning the multi-machine systems gave way for the proliferation of appliances. I think what year, in 2008, 10 years ago, I think the data warehousing market was incredibly hot for appliances, and appliances were nothing but a pre-configure hardware and software solution that were tuned so that it would get the maximum benefit out of – The software could get a maximum benefit out of the hardware. That was the state of the art, again, 10 years ago, and systems were preconfiguring. You can say, “I want to buy the 10 terabyte model, or the 20 terabyte model, or the 40 terabyte, or whatever it was,” and purchase decisions were done and then, “What is the throughput of the IOPS that I would get in a system? What is the dollar price per terabyte?” Back then it was in the thousands of dollars, and it was still very compelling, because it was enabling companies to do analytics and answer questions that were un-thought off before or deal with volumes that were un-thought off before.

Then you enter the latest trend, which is the cloud. Let me backtrack. Before the cloud, there’s another important trend, which is the emergency of the Hadoop-based systems. If you look back at what Hadoop systems do, or HDFS-based systems, and you start to see Spark in there as well and you re-alluded to Map Reduce, all of those systems in my mind were just dealing with shortcomings of the traditional data warehouses. If a traditional data warehouse could have dealt with many petabytes of data and the traditional data warehouse could have dealt with structured and semi-structured data and it could have dealt with very, very large clusters to process queries and every single amount of time, I think it would have just gone from data warehouse systems to larger data warehouse systems and would have probably skipped the whole Hadoop wave, which is a strong statement.

Then that is when you start to see the emergence of the cloud, and cloud starts to be viable. I remember maybe 10,12 years ago, executive briefings – The executives would set the rules, “Please do not bring up the C-cloud, because I’m not C-word, because I am not interested in hearing about this cloud.” The momentum was not there. There were lots of privacy concerns, security concerns, etc.

That is the context in which Snowflake emerges. Now, Snowflake was founded in 2012, and that’s at the hype of Hadoop systems as the scalable way to get answers on analytics. That is a big beginning of companies being more open to the cloud, but it was not yet as mainstream accepted as it may be today. Then we have our founders. Two of them had been at Oracle for 20+ years each, and our third founder, Marcin, he was the CEO at VectorWise, or founder, CTO at VectorWise. He invented Vector-based execution.

The insight that our founders had was the cloud provides a new set of building blocks with a different set of primitives that would enable you to build a data warehouse. It is first and foremost a data warehouse without any of the limitations of the past with some of the benefits of a Hadoop type systems but still with the familiarity of it is SQL, it connects with BI tool, etc., etc., etc.

I'd say it's remarkable to think that at the high of the hype on Hadoop-based systems, these guys went for something that was frankly bold, very, very bold and said, "We're going to go build up traditional data warehouse, because that's what people want. Now, if you fast-forward, I think it was [inaudible 00:36:36], the ability to foresee in reality what customers in the industry would have wanted was the scalable data warehouse and not jump in through all these hoops in between.

I'm going to pause, because [inaudible 00:36:48] in case you have any questions or clarifications.

[00:36:51] JM: It's all really interesting. I would love to go deeper on this presentation of the HDFS, the Map Reduce era as being kind of existing in the gap between when cloud was coming out and when cloud was becoming acceptable. What are the shortcomings of Map Reduce systems, of Hadoop-based systems, and how did Snowflake figure out how to alleviate some of those issues with a new data warehouse system?

[00:37:31] CK: I think that the Hadoop-based systems basically enabled one thing, which was very large volumes of data, and to enable dealing and getting value out of very large volumes of data, and we're talking petabytes, [inaudible 00:37:47] petabytes, Map Reduce comes out of Google and they were doing I believe hundreds of petabytes at some point, and in order to be able to deal with those large volumes of data, there was a tremendous willingness to make tradeoffs. Tradeoffs in what sense? For example, the programming model. Even though from a small data, I use a SQL-based language to query it. I am okay learning a different thing and even writing some Java code if that is the only way that I can deal with large data.

I'm also willing to give up on some of the consistency guarantees. Think of transactional consistency that databases and data warehouses have had for 30+ years, but who cares about the transactions [inaudible 00:38:36] consistency guarantees if I cannot deal with a large data?

Basically, Hadoop and HDFS emerged as an alternative. It was, I think, the only viable alternative to deal with large volumes of data that required large compute clusters and in return there were all these tradeoffs. But the question stands is if there had been a system that could enable all of the traditional concepts, the SQL programming model, the transaction consistency, etc., and dealt with high volumes of data, would that have been the better answer? I think the market right now is telling us, at least based on the momentum that we're seeing on Snowflake, is that the answer is, "Yeah, we didn't want to give up all of those tradeoffs. It was necessary, because there was no other alternative. But given the choice now, I think the traditional SQL model, the traditional database model is powerful and well-received."

Again, much of this would not have been possible had it not been for the cloud, where the cloud gives you, I think, two basic building blocks. One is virtually unlimited amount of storage and the other one is virtually unlimited amount of compute, and as long as the system is architected to deal with those two independently and leverage the scalability of those systems, then you can reimagine data warehouses. Does that make sense?

[00:40:06] JM: Well, what I don't quite understand – So let's go back to, whatever, 2009, 2008, around the time when Hadoop was becoming more established, cloud was starting to become more popular. You're seeing the war between Cloudera, and MapR, and Hortonworks to win over the enterprise Hadoop customers. There's the Map Reduce way of querying your dataset. You're over HDFS and maybe you want to run a hive query. I think hive is the SQL like one. You can issue a SQL query over your HDFS cluster and it gets translated into a Map Reduce query and you get your summary of how your advertising did in a nightly batched Map Reduce job and you get a report the next day for it.

In contrast, there is the query model of hitting your data warehouse. Let's say the same volume of data that's in a Snowflake cluster, how does the query semantics and the query planning and the query distribution across the nodes and the setup for the data on those nodes, how does it differ between the HDFS Map Reduce style way of doing things versus the data warehouse in the Snowflake world?

[00:41:34] CK: I think the single biggest proof that Hadoop emerged in a world where customers and companies actually wanted databases is the fact that a lot of the work on Hadoop for the last many years has been around adding database capabilities, like richer SQL semantics and richer query processing, etc. The question that you ask is a big part of it, which is traditional databases, traditional data warehouses, plays a big premium on being smart in the query compilation and the query plan strategy.

So if you would say, "Give me all the sales for last month." Traditional data warehouses would go way out of their way to make sure that they only look to the data on disk for the last month. Had you done something like this with the early implementations of Hadoop? The mental model was it's not about selecting the last month. It's about selecting everything as fast as possible, and then you can decide to only display or return the result for last month. It was a BS fast as possible a high volume scan engine.

Then when you start comparing the performance of an operation like that with a traditional data warehouse for substance of a data, then the traditional data warehouse is significantly or [inaudible 00:43:02] faster than brute force approach, and then you start to see the iteration where you say, "Okay. Hadoop wants to do a little bit smarter query execution," where it knows which files to touch as supposed to scanning everything. The key thing that Snowflake does is Snowflake breaks up the data into small files, very small files automatically. We'd call them micro partitions, and we keep meta data information about every column in every one of those files.

For example, we could easily tell, "Does this file had information from last month or not?" So when you do the question on, "Give me the sales total for last month," we can very quickly look on the metadata and decide, "Okay, which files do we need to touch?" For a vast majority of those files, probably we don't even need to look at them." Then optimize a query to just look at the relevant data. That is where you see the behavior of the older style data warehousing technology operating on very large volumes of data. That's a key difference and that's why performance-wise, obviously performance varies based on the query, based on the data. So I need to be careful with making strong statements. But in a large number of use cases, or the vast majority of use cases, we see Snowflake delivering a performance advantage.

[00:44:29] JM: So it's almost like you are – Whereas as in the naïve HDFS Map Reduce model, they're just ingesting data, and then whenever you do a query, you scan everything. Because it's Map Reduced, you get to parallelize everything in a dumb parallelizable way. In contrast, the Snowflake way of doing things, at least the early days of doing things was more like you have these files that give you metadata sort of like an index in a database so that you can speed up the access time and be smarter about what parts of your dataset you're scanning.

[00:45:10] CK: Yeah, that's right. I want to acknowledge that, of course, Hadoop-based systems have continued to innovate a lot of the last few years trying to get to that smarter selection of files, but that was built into Snowflake on day one as a core capability. Again, because it's a system built by traditional database architects that just knew how to leverage the power of the cloud.

[SPONSOR MESSAGE]

[00:45:43] JM: Your audience is most likely global. Your customers are everywhere. They're in different countries speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly. Transifex is a SaaS based localization and translation platform that easily integrates with your Agile development process.

Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowd source your translations. If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial.

With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time. Translators work on live content within the development cycle, eliminating the need for freezes or batched translations. Whether you are translating a website, a game, a mobile app or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15 day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[INTERVIEW CONTINUED]

[00:47:32] JM: What kinds of penalties do you pay for implementing that micro-partitioning, the indexing over that volume of data?

[00:47:41] CK: Yeah, I think that if you compare it against a system where a database administrator is absolutely competent, like the strongest person perfectly understands the workload of the users and does very detailed tuning of which file and which indexes, etc., you can envision use cases where you suffer a little bit on the performance side. But in reality, such BDA doesn't quite exist. They're always behind trying to keep up with an ever growing demand of queries and analytics it needs from users and they are actually end up being very busy just keeping the system running, tuning knobs and creating statistics and doing vacuuming of files and partitions and garbage collecting and all those stuff. In Snowflake, the fact that we do all of these automatically freeze up the database owners to spend time in thinking about how do you give more capability and more analytics power to your users as supposed to tuning the system?

In short, there may be a performance hit related to a perfectly tuned system, but a perfectly tuned system is a hypothetical thing that is not the reality of most customers, which is what why at that point then we talk about Snowflake where it uses the performance benefit and probably most important, it gives the simplicity benefit, where the administration of a Snowflake instance is almost zero.

[00:49:16] JM: Yeah. So some issues that come to mind, we just had a show about database performance and optimization, and one of the areas that we focused on was actually indexes, and you have a similar problem here, but it's not as acute, because consistency is not as important in a data warehousing operation. So if you have an influx of a bunch of new data from your transactional data store into the Snowflake data system, you're going to need to index that data into these micro-partitions. You're going to need to create – You're going to need to update the existing indexes. Maybe you need to create new indexes. If you have a query to the data warehouse that comes in before that indexing takes place, you may miss the most recent data that's been ETL'd into the system. Is that correct? Would you count that as a possible disadvantage of the indexed-based model?

[00:50:19] CK: Maybe. Let me clarify a few things. So Snowflake doesn't have indexes. The only concept that you can specify for a table in Snowflake, it's called the clustering key, and the clustering key is just a column or columns that specify what is the physical ordering of a tables on disk. That's the only knob, and if the system doesn't perform, the only other choice for you to have is to increase the size of your cluster, which you can do it dynamically and we can grow [inaudible 00:50:48] clusters with a few seconds of latency.

But other than that, you set a clustering key and that's all you can do. There's no more tuning and indexing and anything like that. To the consistency comment, I think it is true that data warehousing users or data warehousing or analytical workloads may have a tolerance for lower consistency. Again, I think that is just a tradeoff imposed by limitations of traditional technology. Snowflake is a fully transactional system [inaudible 00:51:24]. So you can decide which data gets committed and seen by the rest of the system when you commit the transaction or if something goes wrong when you roller back.

So in reality, with Snowflake, you get the best of both worlds. You get the fully consistent system. You get the very large volumes of data that were not possible before and you get the simplicity of not having to be creating indexes and tuning things. Again, the only thing that you get specify as a cluster and key and that does have a very big impact on query performance, but you specify it and that's the end of it.

[00:51:59] JM: What parts of this data are going to be on disk versus in-memory and how fast does the access time of a data warehouse query need to be?

[00:52:13] CK: Great question. So all of our data gets stored in S3, S3 in Amazon and since we also support Azure, it would be in Azure storage. So all the data goes to persistent disk or a remote storage. As queries in Snowflake get data from storage, we do cache that into the local compute nodes that's cached in SSDs. I don't know if you would come back closer or not to memory, but it's SSDs. So subsequent queries may leverage and benefit existing data in the caches.

To the question on query latency, my utopia would be to say any analytics query should be answered in a second or a few seconds. Obviously, there are other factors that are in play, like data volumes and cluster size. In Snowflake you can have one to as many clusters as you want and each one of them specify the size. Larger clusters are 128 nodes. So you have some control in the performance of the query, but at the end of the day, there's physics involved. If you have – I don't know, 10 petabytes, or 10 petabytes of data, there are various really physics in the way of how quickly you can move that much data over a network [inaudible 00:53:38] that is a gigabyte of networking.

But I would say for properly constructed queries that are looking at subsets of data where you don't have to move a petabyte of data in the query, it is not uncommon to just take a couple of seconds for regular query performance.

[00:53:56] JM: We've seen the rise of these streaming systems, like Spark and Dataflow from Google. What access patterns and architectural patterns are you seeing of people using these systems or systems like Tensorflow together with a data warehouse like Snowflake?

[00:54:19] CK: Yeah. For sure it's an important trend and it speaks to what I mentioned around lower time to insight. I think the notion of something happened today and I need to wait until tomorrow for my dashboard to report, that is becoming increasingly rare. It still happens a lot and many companies are still doing like daily ETLs at the end of the day, but the need and the notion is I want data sooner. I want lower latency. I want lower time to insight. The way to achieve this is by having streaming interfaces that are just moving data continuously or in very small batches from wherever the data is getting created to this system where you analyze it. That's where you see the emergence of Dataflow, Kafka, Kinesis. All of these systems are great to move data with low latency in a streaming fashion from one place to another and a large number of customers are landing continuous data into S3 or Azure Storage or GCS over in Google Cloud.

Eventually what customers want is to analyze that data. So we see all sorts of patterns. I was just doing a week-long customer visits, and I think most customer I talk to had some form of – This process that was getting data from a Kafka queue into S3. Then from S3, in Snowflake we have a feature called Snow Pipe, that basically you can configure that as files land on S3, we will identify and pick them up and loading them into Snowflake making them available for query.

I think, today, latency is on the order of a minute or two end to end from when the data lands in S3 and the moment that it's available in Snowflake. What we're seeing as a big trend is, yes, customers want a lot of streaming, and we at Snowflake integrate with all of those systems, make it easy to ingest data from those streaming systems and make it available for query as low as possible.

Aspirationally, I would like to continue pushing latency lower, and there are some tradeoffs there. But it's pretty remarkable that many customers today can tell you that their dashboards are 5, or 10, or 15 minutes out of date as supposed to daily or even longer.

[00:56:50] JM: All right. Just a few more questions and we'll wrap up. You alluded earlier to the fact that you don't really think that the notion of a data lake is maybe necessary or is the right abstraction to be thinking about. I think I agree with you. You're the second or third or fourth person who's come on the show and kind of described the data lake being replaced by some system that involves Kafka and S3, and you could say S3 is the data lake here, or you could say Kafka is the data lake here. Maybe could you give a little bit more color on what you mean by when you say that this data lake idea might not be necessary and how Kafka fits into this equation.

[00:57:34] CK: I think of a data lake as a set of requirements. I need to be able to fit all my data. I need to be able to fit structure and a structured data. I need to be able to keep it in its original form and I need to be able to model it for query and analytics. Once you turn it into requirements, then there are many systems that do that. S3 with, say, a Kafka queue or a Kinesis queue helping move data from operational system into storage and some query ability that would provide a data lake solution. The query availability, you may decide where you want – I don't know, a [inaudible 00:58:13] system or [inaudible 00:58:14] spectrum, etc. There are multiple ways to query the data, which as a whole, satisfy the needs of a data lake.

Obviously, I'm at Snowflake and I'm convinced that Snowflake is a perfectly suitable solutions to meet the data lake requirements. You can ingest as much data as you want. You can ingest structured data and semi-structured data and leave it in an original form and convert it later on. We have a world class query engine to analyze the data.

So, yeah, I think that data lake is less so a system and more a pattern. As we discussed Kafka or Kinesis can feed Snow in the same way that it can feed S3 or Azure storage. As a set of requirements, there are multiple systems that meet those needs. Of course, Snowflake can do those very well.

[00:59:08] JM: All right. Just to close this off, we've been talking about a lot of the problems and the opportunities in the data platform space and it's a pretty amazing time, because we're starting to see so much amazing technology be accessible to enterprises that don't have as much engineering capability. They have some engineering capability, but the combination of cloud and really interesting startups that have been created on top of cloud, basic cloud infrastructure, like S3, is just really amazing to see.

So what are the things that we're going to see from the data platform industry as a whole and Snowflake in particular over the next 5 to 10 years?

[00:59:53] CK: I think the move to managed services is a big trend and this enable by the cloud. Managed services that there are companies like Snowflake that are running the infrastructure and removing from the customers to do list everything that was traditionally in there to the list. Are procuring hardward? Are you getting new servers? Are you renewing them? Are you patching them? Etc. We take it a step further on administration of tables and [inaudible 01:00:23] and all those things. I think managed service as a whole will continue to rise. I think the cloud is here to stay. I have not spoken to one company that has told me we're not doing the cloud. Even the most conservative want to have plan B on-prem, they're still thinking cloud fits a big portion of our strategic long-term architecture.

I think multi-cloud is emerging very strong across most of the customer [inaudible 01:00:53]. Customers don't want to lock in into a single cloud where my managed service lives. So you'll see many services, again, like Snowflake providing ability to run on one or multiple clouds. Today Snowflake is on AWS and azure. Sooner we'll do others. Basically, how do we abstract some of the, let's say, cloud-specific capabilities so that higher value services can be presented to customers and, again, simplifying the experience. I think you're going to see a big shift away from managing infrastructure and from just using services.

[01:01:33] JM: I completely agree with you. I'm sure we could have talked about things like the compute moving to things like lambda as well or other higher-level compute layers, but I think we've talked enough. Christian, this has been really interesting. I appreciate you coming on to give the background of the data platform world and your time at Snowflake.

[01:01:54] CK: Thank you so much for inviting me.

[END OF INTERVIEW]

[01:01:58] JM: Thank you to our sponsor, Datadog, a monitoring platform that unites metrics, distributed request traces and logs in one platform. Datadog has released new log management features and trace search and analytics to help you explore and analyze your APM data on the fly. Plus, Watchdog is a new auto detection engine that uses machine learning to alert you to performance anomalies without any setup.

Datadog includes out of the box supports for more than 250 technologies, including PostgreS, AWS, Kubernetes and more. With a rich dashboards, algorithmic alerts and collaboration tools, Datadog can help your team troubleshoot and optimize modern applications. Start monitoring with a 14-day trial and Datadog will send you a free t-shirt. You can go to softwareengineeringdaily.com/datadog to get that free t-shirt and to try out Datadog with all its new features. That's softwareengineeringdaily.com/datadog.

[END]