EPISODE 999

[INTRODUCTION]

[00:00:00] JM: Nubank is a popular bank that is based in Brazil. Nubank has more than 20 million customers and has accumulated a high-volume of data over the 6 years since it has started. Mobile computing and cloud computing have given rise to challenger banks, such as Nubank, that operate more like software companies. When a software company reaches the size that Nubank is at today, it needs a data platform.

A data platform is a collection of different technologies that move data into different storage formats and applications so that different members of an organization can access that data. New data often enters an organization through an OLTP database. The transactional database which supports user transactions, and that transactional data is copied into a data lake which provides cheap bulk storage.

From the data lake, the data is moved into a data warehouse system for fast access, and along the way tools like Kafka, Spark and S3 are used to implement the needs of the data platform, and that's kind of a reductive description, because data platform architecture is not an exact science and different companies build their data platforms based on their own unique requirements.

Previous shows have covered the data infrastructure companies like Lyft, and Uber, and Facebook, and today's show is another case study in data infrastructure with a modern bank. In a previous episode a few years ago, we covered the core engineering of Nubank. In today's show we cover the data infrastructure of Nubank with Sujith Nair. He joins the show to talk about the data infrastructure of the company and what he's working on.

If you enjoy this show, you can find all of our past episodes about data infrastructure by going to softwaredaily.com. You can search for the technologies of the companies mentioned. Also, if there's a subject that you want to hear covered that you have not heard in a previous episode, you can feel free to leave a comment on this episode or send us a tweet @software_daily with your recommendations.

[SPONSOR MESSAGE]

[00:02:10] JM: If you don't love your job, it's hard to wake up in the morning with a feeling of excitement. Of course, finding a new job is a painful process. There's a lot of friction, and that's where Seen by Indeed comes in. Seen puts tech candidates in front of thousands of companies, like Grubhub, Capital One and PayPal across more than 90 cities. Just create your profile from your resume and they'll match you to the right roles based on your needs.

Every Seen candidate also gets free access to technical career coaching, resume reviews, mock interviews and even salary negotiation tips to seal the deal. Join today and get a free resume review when you go to beseen.com/daily. That's B-E-S-E-E-N.com/daily. Seen by Indeed is a tech-focused matching platform and the access to resume reviews, mock interviews and salary negotiation tips can be really helpful to somebody who has not looked for a job for a while or even somebody who has been looking for a job but just hasn't really considered these things like practicing before an interview or learning to negotiate a salary. You might be worth lot more than you realize.

If you're ready for a new job, you're ready for Seen by Indeed. Join today and get a free resume review when you go to beseen.com/daily, B-E-S-E-E-N.com/D-A-I-L-Y. Thank you to Seen by Indeed.

[INTERVIEW]

[00:03:54] JM: Sujith Nair, welcome to Software Engineering Daily.

[00:03:56] SN: Thank you, Jeff. Glad to be here.

[00:03:59] JM: We're going to talk today about data engineering in the company that you work at, which is Nubank, and Nubank is a thriving bank. It's commonly referred to as a challenger bank. So it's a newer bank. I want to just start by just talking about what Nubank is so we can get an understanding of the high-level application and then we can get into the data engineering side of things. Could you just give an overview for what the Nubank business is?

[00:04:26] SN: Sure. When Nubank started, it started out as a credit card issuer, a mobile app-based credit card issuer. That was the initial business model to provide credit cards to people in Brazil. Overtime, the products that we offer to customers has – We have moved on from credit cards to savings accounts and other financial products. Now we provide these services to customers in the countries of Brazil and Mexico.

[00:05:02] JM: We're going to need to talk in terms of both OLTP and OLAP transactions in the episode today, because when you're talking about data engineering, there is usually analytic processing that you're doing and you're doing these large scale analytic processes over large datasets. But when these datasets are created, they're typically created from individual transactions.

In the case of Nubank, maybe a customer makes a purchase or a customer transfers money from one bank account to another or a customer gets a credit card. Can you give an overview of some prototypical transactions, the user transactions that are going on across the Nubank platform?

[00:05:52] SN: Like you said, every transaction, every purchase that a customer makes is registered as a transaction within the Nubank system, in addition, money transfers or bill payments. These are different users which a customer makes of the Nubank card in general and each of these makes its way into the Nubank system as, even, I would say, as a record. Yeah.

[00:06:18] JM: The data after it's turned into transactional records, eventually it's going to be processed in other ways over analytic systems. When we talk about the analytic processing, that's going to be a very complicated system of things like Kafka and Spark. So before we get there, let's talk a bit about the data infrastructure of the transactional processing.

When a user actually makes a purchase and that purchase is written to your database, I assume that the user is interacting with some client, like a mobile client or just a credit card and the transaction eventually hits Nubank's services architecture and then the service is going to write that data to a transactional database. What do you use for transactional database?

[00:07:16] SN: Nubank as a whole is structured as the services which we provide, the user-facing services or the internal systems. They are structured as microservices. Across Nubank engineering, we have a system of like writing microservices enclosure and each of these microservices are backed by a Datomic database. The transactional database that we use is Datomic. For people who are unaware of Datomic as such, Datomic is an immutable append-only database. You can think of it as Git for your database. In that sense, the records which are Datomic, the log is a first-class construct. You can imagine Datomic, the transaction stored in Datomic has been appended to a log, and then when users query it, you can query against a history of it. You can do time travel, and that is how the transactional records are stored.

[00:08:23] JM: The usage of Datomic is not something that's common. It's not necessarily uncommon. I mean, there are plenty of people who love Datomic. They love this idea of having an appen-only database that allows you to go back in time and see the history of a single database entry.

But in most databases, you have something called a write-ahead log. In a SQL database, you have a transaction history. You can implicitly go back in time because of the write-ahead log in a transactional database. Why do you need the log as a first-class citizen in your transactional database?

[00:09:08] SN: I suppose historically the reason – I mean, as a financial institution, audit trails and being able to reason about things in the past is something that the Nubank engineering cared from the start. Although you do mention that other relational databases do provide the construct of a write-ahead log, these are usually I would say a secondary constructs in which you have to work out mechanisms to extract value from it. Within Datomic, you do not – Because it is structured as a log, you do not have to come up with extensive change data capture mechanisms to extract value from the log itself. Yeah, I think that would be the reason why we prefer Datomic.

[00:10:00] JM: If I'm keeping all of the data over the entire history of my entire application, does that get to be so much data that it's going to cost me a lot of money or be hard to maintain?

[00:10:14] SN: Well, our experience suggests that it is not the case. Particularly in the modern era of cheap cloud storage, that has not been the problem at all for us. Datomic, as such, there are other limitations to Datomic in the sense that because it is an append-only database, there certain limitations to the scale that it can handle. Those are the limitations that we have faced, but [inaudible 00:10:44] storage or having to deal with the cost of that storage has not been much of an issue.

[00:10:51] JM: We've touched on the transactional data store at this point, the fact that you use Datomic to store things like user data and transaction data and financial data. Now we can start to talk about the data platform. What are the requirements of your data platform?

[00:11:12] SN: The need or the requirements of the data platform, I would say – Well, it comes from the fact that Nubank as such is structured – It started with microservices, each of the user-facing features or the internal accounting system. All of these are catered to by microservices. The need to stitch together data from these different data sources, these different databases, if you will, was always there. In addition, as a financial institution, we also have access to third-party data sources, data vendors who provide us datasets. The need to stitch together data from these disparate sources and to provide a platform for analysis, that is what I would say is the need or the requirement of the Nubank data platform.

[00:12:08] JM: When I think about the users of a data platform, I think broadly of two kinds of people. One kind of person is the interactive data analyst. This person is often times referred to as a data analyst or a data scientist. This is a person who might be interacting with a BI tool, like Tableau, or Looker, or maybe they're putting stuff into an Excel spreadsheet and they're building models.

Then you have the maybe machine learning data scientist, or the machine learning engineer, or the data engineer, and these are the kinds of people that are doing more engineering work. They're transforming data from an OLTP system like Datomic. Maybe they're moving it into a data warehousing system so that they can build aggregations. Maybe they want to create nightly reports and they want to do more technical work that they automate so they want to have these reports be automated or they want to create aggregations of transactions and send them throughout the company.

Tell me about the breakdown of the different kinds of users of your data platform.

[00:13:24] SN: Sure. As you alluded to, there are different user personas which make use of the analytical environment, the data platform. To name a few, I would say there are data analysts. There are business analysts, financial analysts, data scientists and machine leaning engineers. These are all the different internal users that that data platform caters to. To answer your question as to how do we actually cater to this wide variety of user personas, well, I would say from the onset, we were clear that to cater to such a wide variety of users and their needs and to scale the platform to the levels that we envisioned, the data transformation had to be devolved and democratized to the user. In a sense, what I'm saying is they would define the data transformation they require.

To give you a brief view of the architecture of how we do it, I would say that for a data analyst, business analyst kind of users, we have a data warehouse hosted on Google BigQuery. But to the more – I would say there is the transformation of transactional data and there's the process of loading that into a data warehouse. The approach that we have taken as a platform is to let the users decide what flows through our data lake and what eventually gets stored into the data warehouse.

Maybe I should also talk a bit about the architecture of the data platform in general. Like I said, we were clear from the very beginning that data transformations have to be devolved and democratized to the users. To achieve this, we separated the concerns of behavior, state and performance. Behavior is the transformation logic or how data is transformed from one form to another. Is user contributed? But incorrect behavior, or buggy transformation, or any such thing will – Any buggy code will not lead to an inconsistent state of the data lake per se.

In that sense, I would like to also talk briefly about the architecture of the data lake maybe?

[00:16:02] JM: Please do.

[00:16:03] SN: Yeah. The way we have structured the data lake I would say is the core primitive of the data lake is a bag of functions called ops, as in directed acyclic graph of ops. Ops stands for operations. An op is a decorated Scala function that consumes as Spark data frames and produces Spark data frames. These ops are contributed to the platform by the users of the system. I mentioned, data analysts, business analysts, machine learning engineers and others. These ops are deterministic and item-potent.

For a given input of data frames, if you will, they always output the same data frames. What other benefits of structuring the data transformations within the data lake as item-potent ops? The biggest benefit, the biggest advantage of this is that such item-potent ops are very conducive to unit testing. The users who contribute ops to the system can be very sure of the correctness of their ops.

Also, to tie back to the initial mention of separation between state behavior and performance, the users can focus on contributing behavior to the data lake while the data infrastructure team focuses exclusively on the problems of performance and essential state.

[00:17:41] JM: You've given us a lot of context there, and I'd like to explore a lot of the things that you talked about and zoom in on them. I want to start with this term data frame. Can you give an explanation for what a data frame is?

[00:17:57] SN: Sure. I mean, in general – I mean, not to talk about Spark, particularly. But in general, a data frame can be viewed in a very relational sense a table. You can think of a data frame as being structured as multiple columns and rows. That's how you would think or visualize a data frame as a relational table.

[00:18:21] JM: The idea of a data frame, we create these because we're taking a subset of the data from our data lake, right? The data in the data lake, it could be unstructured data. It could be – Maybe some of it is structured. Maybe we're taking different sources from our data lake and pulling it together into relational format and that is the data frame. Am I understanding it correctly?

[00:18:54] SN: Right. Assuming that your initial source of data is unstructured or semi-structured, by the time it ends up in the data lake in the form of an op or a data frame, it is already in the form of – Has already assume some structure, some relational structure yeah.

[00:19:14] JM: Let's go back a little bit. First of all, I guess we kind of skipped the data lake idea. This data gets written to transactions, or this transactional data gets written to the OLTP database, the Datomic database in your case. What kind of data is getting put into your data lake? How does data make its way into your data lake?

[00:19:35] SN: Sure. Data ingestion into the data platform, there are many sources, multiple types of sources. Let's talk about how data from the production databases makes its way into the data lake. Like I said, each of our production microservice is backed by a Datomic database. As I alluded to before, since the log is a first-class construct and that we do not need extensive change data capture mechanism to capture logs form it.

A data platform service connects to every production data instance, extracts logs from it at regular intervals and stores it as Avro files on S3. These Avro files are read into Spark during the batch execution of the ops, I would say.

[00:20:29] JM: Avro files, that's a format that's like similar to Parquet, right? It's like a file format that's useful for storing large volumes of data.

[00:20:39] SN: Yeah. I wouldn't call it similar to Parquet, because Parquet is a columnar file format in general. It's more suitable for workloads where you are performing full table scans or reading entire files, that sense. In Parquet, the more suitable workload is to – The more analytically kind of workloads wherein you're only scanning through a certain subset of the columns.

[00:21:08] JM: Tell me more about the choice of the Avro file format for your data lake.

[00:21:15] SN: I wouldn't say Avro is the file format of source for the entire data lake, but the initial ingestion. At the ingestion stage, that is where we use Avro. The preference for Avro at that stage is because when the batch execution of Spark happens over this ingested data, it usually happens via full table scans. For such workloads, Avro has a better performance and that's why we have Avro at that stage.

But if you were to think of – Well, one point that I missed out on when describing the data lake, these ops, they're structured as a directed acyclic graph. At the very beginning of it, the data is in the format of Avro. It's read into Spark and processed. But subsequent outputs as stored as Parquet on S3.

[00:22:11] JM: Okay. The data that's actually getting put into the data lake, the first dump of transactional data, is it just the copies of the Datomic transactional database or we talking about some different data sources other than the transactional database?

[00:22:32] SN: It is not a direct copy. There is a transformation needed in the sense that Datomic, like I said, it stores things in a log form, in a long-form format which needs to be kind of pivoted into materialized view to make a sense in the data lake. Of course, there is that transformation. I wouldn't call it a simple dump of data.

But having said that, this is not the only source of data into the data lake. Other sources would be – There is a form of streaming ingestion into the data lake. Microservices can stream data into the analytical environment by publishing them as messages via Kafka. A service, a data platform service again consumes these messages from Kafka. Accumulates the data from messages in batches and stores them again as Avro files on S3.

Also, other data sources that I can think of are, as I mentioned before, as a financial institution we have access to third-party datasets, which are hosted on S3 and which are again accessible in this DAG of Spark jobs.

[SPONSOR MESSAGE]

[00:23:56] JM: Today's show is sponsored by StrongDM. StrongDM is a system for managing and monitoring access to servers, databases and Kubernetes clusters. You already treat infrastructure as code. StrongDM lets you do the same with access. With StrongDM, easily extend your identity provider to manage infrastructure access.

It’s one click to onboard and one click to terminate. Instantly pull people in and out of roles. Admins get full auditability into anything that anyone does. When they connect? What queries they run? What commands are typed? It’s full visibility into everything. For SSH, RDP and Kubernetes, that means video replays. For databases, it’s a single unified query log across all database management systems. StrongDM is used by admins at Greenhouse, Hurst, Peloton, Betterment and SoFi Control Access.

Start your free 14-day trial of StrongDM by going to softwareengineeringdaily.com/strongdm. That’s softwareengineeringdaily.com/strongdm.

Thank you to StrongDM for sponsoring Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:25:18] JM: You’re talking about the different data sources that are getting put into the data lake. Just to give a little bit more color there to make sure I understand this correctly. You have one kind of data source, which is just your transactional database and you have to do some kinds of transformations on that data to put it into the Avro format so that it’s more easily consumable by Spark jobs in the future. You also have data that is being consumed from Kafka. This is like exhaust data from your microservices.

Maybe your microservices are producing – I don’t know. Logging data or just maybe certain exception kind of data, but your microservices are generated to some kind of data that’s getting pushed into Kafka. Then off of Kafka, it’s getting transformed into other files that can be stored in the data lake. Then you also have third-party data sources. Maybe you have a daily dump of tax information or credit card rates or just different kinds of third-party data sources that might need to be written into the data lake because you have a variety of consumers that might want to do interesting data engineering work based off of those third-party data sources.

[00:26:31] SN: Correct.

[00:26:33] JM: Okay. Cool. Now, as we talk about working off of that data lake, coming back to this data frame idea, what is the process of taking data from a data lake, in Nubank's case, and making practical use of it?

[00:26:53] SN: Sure. Again, it depends on the user. I mean, what kind of use to the data the user wants to do. What kind of transformations they want to do. Let's assume that you want to define certain joins on certain datasets and then certain transformations on some datasets. You would define, let's say, an op. Like I said, these ops are just decorated Scala functions and you know. These Scala functions, you are more or less writing simple Spark, SQL transformations of how these different data frames or output of ops should be transformed into something else.

Then you can annotate them to say that, "Okay. I need this data to flow into the data warehouse for further analysis," or you can annotate it to say that this needs to be loaded into a DynamoDB table so that it can be fed back into the production environment where microservices exist, or you want it to be, well, sent as Kafka messages into certain topics so that downstream consumers of it, which could be other services, they can process it. These are the different ways in which data can flow out of this analytical environment.

[00:28:16] JM: That term op that you've described a few times. So when you're talking about taking the data out of its raw data lake format and then transforming it into something more useful, I think you're using the data frame and op somewhat interchangeably. I guess you're saying that the idea of an op is a structured, a table-like format that is easier to work with for a data analyst or a data scientist because it is in a relational format. Is that accurate? That's why you're saving it back to the data lake.

[00:28:55] SN: Yeah. To clarify the difference between an op and a data frame, I would say an op is basically a function, function which defines a transformation. You can assume that it takes in some inputs, which could be data frames. The op is the function itself and you can visualize the data frame as being the output of that op.

[00:29:17] JM: I see. Okay. It's useful to save these kinds of ops because if you have these kinds of ops to find, you could chain them together and create meaning calculations that you could reproduce on a daily basis, on an ad hoc basis. You could reproduce them in all kinds of applications.

[00:29:38] SN: Yes. That's exactly the idea.

[00:29:41] JM: Interesting. From our correspondence, I think that what you're describing here is inspired some by Maxime Beauchemin's idea of functional data engineering. Can you elaborate on that term?

[00:29:56] SN: Yeah. The idea of – Well, I would give you a premise of why the ideas of functional programming paradigm is needed in the first place. In general, data engineering elsewhere in many places I would say, the data engineering workflows or data engineering jobs as such, it is very difficult to test. It is not a norm, I would say.

As this domain of software engineering has matured, many people have question or like they have spent some thought on how to introduce testability and reproducibility into data engineering workflows. Maxime Beauchemin's post alluding to that fact, how we can structure data engineering workflows in a form which makes it easier for users to make it testable and reproducible?

[00:30:54] JM: Well said. I mean, the idea of testable and reproducible data engineering is something that's come up in some of my conversations with other data engineering people. There's a show we're doing in the near future about great expectations, which is I think a tool for essentially doing unit testing on datasets and data engineering workflows. I understand that this is pressing issue. But I think it's worth applying this functional data engineering concept to what we have discussed in practice so far. Can you tell me how functional data engineering applies to this infrastructure of ops and data frames that we've been discussing?

[00:31:36] SN: Sure. You can think of the output of these ops, like I said, the data frames as being each op can be visualized as mapping to a partitioned table, if you will. Each execution of an op can be visualized as pointing to a single partition in a table. Let's say over multiple executions, what we are doing is essentially adding new partitions to the table, each of this op. To maintain the – To kind of borrow the idea of functional programming into the data world, what function programming essentially works with is the idea of pure functions so that given the same set of inputs, it would always give the same output and there is no mutable state. In a sense, there's no global mutable state, which makes things easier to reason about, easier to test, etc.

Now if you were to borrow that idea into data engineering, you would say that when you execute an op, you're basically writing, adding a new partition into a table. If you are rerunning an op within the same execution, it would require you to kind of overwrite the same partition so that is never an inconsistent state as such.

[00:33:06] JM: Yeah. That makes sense. Now, I want to come back to what you've said about the idea of an op or the idea of what you've said, is basically a functional operation that will produce a data frame. How are you saving those operations and sharing them throughout the organization so that you different people across the organization can make use of those different operations?

[00:33:33] SN: Sure. The output of each op is stored into S3. But however, a central catch log service, if you will, keeps track of the output path of like where is the output of this particular execution, often op, stored in S3? Let's say that each batch execution of this DAG is tracked within a single transaction, let's say, a single transaction of transformation.

Whenever an op execution is done, it stores that information in the catalog service that the output of this op is stored in particular S3 pad and this is a schema, etc. This concept of the central catalog service is also important to understand how we model the four mentioned overwriting of partitions, because when people talk about overwriting, we do not actually need to perform a physical overwrite, because overwrites can be modeled as a free metadata operations in this central catalog service.

Once this is stored and the central catalog service has access to the output of a particular op, other services can talk to this catalog service and say, "I need the output of a certain op for a certain day," and then access the records. To understand, also, this output is further loaded into – Some of them are loaded into a data warehouse and that is also another way that people can access this data.

[00:35:23] JM: This is coming into a clearer picture now, because if you have this system where there are a multitude of different phases in this data pipeline, you're going to sign different phases in this data pipeline to different kinds of programmers. At this point, you've defined the ingest system, the process by which data that is generated from the transactions are ingested into the data lake.

They're transformed into a file format that's easier for data engineers to consume. You've defined a process by which data engineers can create these functions that can run over your data lake and create data frames, consumable sets of data that could be written to S3 on a period basis. Then now you've basically led us up to a point in which the data has been muncged into a format that is in a table that could be consumed by a semi-technical or a less technical user, a data analyst, or a data scientist or somebody machine learning. If I understand correctly, you've defined so far the data pipeline that leads to basically the set of tables that are more easily consumable by a wide range of data users.

[00:36:59] SN: Yes. Correct.

[00:37:00] JM: Great. Once the data is in S3 in a tabular format, in a data frame format, and I think you mentioned that you have a cataloging service that is going to enumerate those different datasets, take me into the life of a data scientist or a technical analyst that is consuming this tabular format at the end.

[00:37:23] SN: Sure. Assuming that they have different workflow, let's start with a data analyst. Let's say a data analyst requires to – He or she works on certain dashboards which rely on certain datasets to be computed every day. These dashboard rely on data, which is stored in the data warehouse. Well, we have a BI tool called Looker, which sits on top of the warehouse. Of course, those dashboards are refreshed every time newly data is loaded into the data warehouse.

Now, assuming that the user is not a passive consumer of data, but wants to define certain transformations, they would just go ahead and define these as simple ops and contribute it to the repository. From that point onwards, they would have access to these data loaded in a repeated fashion into the data warehouse. Now to talk about, let's say, a data scientist. Like I said, this platform is about solving data engineering workflows. As such, I would not say that it solves all the problems which a data scientist would face in the course of their normal work.

Of course, a large part of the work of a data scientist involves data engineering work such as training some features or scoring their model with new data, etc. Each of these functions can again be modeled as an op. Let's assume you are trying to train some features and you're trying to kind of course data in a certain form so that it can be fed into your machine learning models. That would be defined as an op in the data lake and that data would then be available for your machine learning models. Then once your models area defined, the scoring aspect of it can again work within the same DAG of operations.

[00:39:45] JM: Now, I'd like to go a little deeper into the different roles that you have that are consuming these end datasets, like the Looker analyst, for example, or the machine learning engineer who's building models. Let's first talk specifically about that data analyst, the kind of person that is using Looker and building dashboards or consuming dashboards. Do you have an example use case? What might this person be doing on a day-to-day basis?

[00:40:18] SN: Yeah. I mean, depending on the particular team this analyst belongs to, they could be tracking different things very simplistically as to the number of credit cards issued in a particular month or day or the number of accounts opened or other metrics such as – I mean, I would be at a loss to list out the entire set of use cases, because there are far too many of them.

[00:40:48] JM: Right. But I think it's worth pointing out that this conversation is going to give people context for just how much data engineering work needs to go on behind-the-scenes in order to generate these kinds of dashboards. I mean, they're maybe data scientists out there who they come into work every day and they look at these dashboards and they have no idea how these dashboards are making their way to their desks, but it is in fact a complex series of steps that need to take place in order to actually have this data hit their dashboard.

Then, certainly, if they're doing some things that are more dynamic, like if they want to use Looker to generate customer ports, custom ad hoc reports, then they need to have these datasets that are easily consumable by Looker, and that's what we described in the data frame generation process.

[00:41:40] SN: Right. I general, I would say it is for the better that such complexity is masked from the users, the end users. Typically, we would very much like for them to never having to know about – It is only when our – Well, daily execution back jobs fail that they would be like, "Oh! This data did not arrive, or this did not refresh." A general sense, we would like for them to not be even aware of that such thing is happening under the hood. That would be the ideal case.

[00:42:17] JM: We should actually discuss some more of the infrastructure usage. You mentioned the use of BigQuery near the beginning of this show, and we haven't really talked about how you use BigQuery and what BigQuery is. Could you explain what BigQuery is and describe how it fits into your workflows at Nubank?

[00:42:38] SN: Sure. BigQuery is a managed warehousing solution by Google. How it fits into this infrastructure we have at Nubank is that, like I said, some of these ops definitions can be annotated that it flow downstream into the data warehouse. You can think of the data that we have in the data warehouse as a set of curated data which flows out of the data lake, which is more or less in the final form that data analysts or financial analysts would like to consume, and that data is loaded – That is the kind of data which is loaded into BigQuery and then accessible via Looker or even the BigQuery console to the user.

[00:43:26] JM: Now, how does the usage of that data compare to what we were describing with the process of munging through this data and generating data frames that get written to S3?

[00:43:41] SN: There are a set of power users within systems who are actively defining ops, kind of defining the data transformation which they need and final form they need their data to be in. If you leave out these set of power users, a lot of our users tend to be people who are code hours and they still need access to data, because to take an example, a financial analyst needs to compute some metrics which tell them about the health of the company, the health of certain businesses, etc.

I mean, they have the entire complexity of financial regulations and the bureaucracy to deal with. For them, dealing with the complexity of code or dealing with the complexity of defining these transformations may not be the best user of their time. These are the users who would rely more on a BigQuery or Looker to kind of access their data, analyze it or even visualize them. Yes.

[SPONSOR MESSAGE]

[00:45:01] JM: Today's episode is sponsored by Datadog, a modern, full-stack monitoring platform for cloud infrastructure, applications, logs and metrics all in one place. From their recent report on serverless adaption and trends, Datadog found half of their customer base using EC2s have now adapted AWS Lambda. They've examined real-world serverless usage by thousands of companies running millions of distinct serverless functions and found half of Lambda implications run for less than 800 milliseconds.

You can easily monitor all your serverless functions in one place and generate serverless metrics straight from Datadog. Check it our yourself by signing up for a free 14-day trial and get a free t-shirt at softwareengineeringdaily.com/datadogtshirt. That's softwareengineeringdaily.com/datadogtshirt for your free t-shirt and 14-day trial from Datadog. Go to softwareengineeringdaily.com/datadogtshirt.

Thank you, Datadog.

[INTERVIEW CONTINUED]

[00:46:14] JM: Let's take a step back and talk some about the choices of technology that you've made. Have you considered using Snowfl ake for data warehousing? What was the tradeoff decision between using Snowfl ake and other technologies like BitQuery?

[00:46:31] SN: Sure. We did not start out with BigQuery. The initial data warehouse was Red Shift that we were using. Then at some point, although I wasn't directly involved in the decision, there did come a stage where we as a team thought that Red Shift was not well-scaling as well as we would like it to. At that point, we were on the lookout for new alternatives. I may not be the best person to talk about this, but I'm sure that a lot of alternatives were considered, BigQuery, Presto, Snowflake might have been among them. But in the end, for a variety of reasons such as the ease of management and pricing and certain other aspects, we decided to go with BigQuery.

[00:47:24] JM: Yeah. That sounds like a subject for an entire show, the choices of modern data warehousing. You've written about the history of data engineering on Twitter. I know you have a pretty broad historical perspective. When you consider data engineering since the creation of Hadoop, where are the key inflection points in the history of data engineering?

[00:47:47] SN: Yeah. Again, this is a very opinionated, I would say I might be very well wrong on this. But I think during – The way I like to think about the history or the evolution of modern data engineering is that it has been thought of a cyclical in the sense that we come back, we keep going back to the same solutions or the same abstractions. With respect to data engineering, I was particularly alluding to the fact that how the abstraction of a table or the construct of a table is something which was lost at some point when Hadoop was introduced. Well, how that has slowly surfaced back again.

When I say the abstraction of table, what I'm talking about here is, in general, the declarative language of SQL, but also the low cost of execution in itself. Hadoop, when it was originally designed, and still, has its low cost of execution to be a file. Then users were supposed to ride these MapReduce jobs which was pretty technical and not really accessible for somebody higher up or in the value chain.

These two aspects somehow, like at some point, people realized that we need to go back to SQL. That is an interface which many users would find more accessible. In that context, I have written about how [inaudible 00:49:28] step forward in that direction the reintroduction of SQL in the hadoop ecosystem and the reintroduction of table as such. All the way through the present day era within Spark, Spark SQL is also of course a step in that direction. Then you have the current stage where like there are new table constructs coming up on top of such processing frameworks such as Apache Iceberg or Databricks Delta and others.

[00:50:02] JM: Yeah. I'm just preparing for a show about Presto right now, and from what I understand, Presto is basically a system of like give it SQL and it sorts through your legacy technologies and figures out how to satisfy your SQL query on top of files.

[00:50:21] SN: Right. I mean, yeah, that seemed to be the direction in which – I mean, it's not a bad direction to be in, but yeah, that seems to be the direction in which the domain is headed. Yeah.

[00:50:32] JM: I'm sure there's a lot of other stuff we could discuss. I know we're nearing the end of our time. I guess just a few random contextual questions, because it seems like you follow this broad area of data engineering pretty closely, the separation of ideas between a data warehouse and a data lake.

From what I can tell, there are industry attempts to unify those two things. Maybe not actually unifying them, but at least providing abstractions that make it seem like your data warehouse and your data lake are the same thing. When I did a show about Snowflake, it seemed like that's what they were trying to do. Do you have a sense of what it takes to make the data warehouse and the data lake feel like two seamless abstractions?

[00:51:21] SN: Sure. My feeling about this is the entire reason for the existence of two separate notions of data lake and data warehouse, that stems from the fact that the current way in which data engineering systems are built is this federated way of stretching together, processing technologies or processing frameworks and storage frameworks and you need a specialized team to kind of stich them together or kind of course them together, because it does not come easily. It takes some degree of expertise to do that.

I think of Snowflake, somehow – I mean, I do not know much about the Snowflake architecture, but I would say the Snowflake is kind of trying to address that need of not having to stitch together these disparate systems or course them together into a system which works for the user.

Having said that, in the current scenario the distinction between data lake and data warehouse is also moving towards a more – Well, I would say the primary difference between them would be to what degree of modeling or what degree of curation does the data have. Data lake is more of a staging area, if you will, a docking area for all the data of the organization to flow into. Then data warehouse is mostly a curation or selection of data of clean data into something which is more useful for the end user.

[00:53:13] JM: Well said. I know that you are concerned right now with the idea of data quality. Ensuring data quality throughout an organization is really hard. One example of why it's hard is you have different teams throughout an organization that are writing data to perhaps the same data lake with different idea of what different terms might mean.

For example, if I write data to my data lake and I'm using the word account or the word currency, I might have different ideas for what I mean by those terms than you do. If there's some data scientist that's trying to do a join between datasets that are created by different teams, they can run into problems with data quality. Now, that's just one example of a data quality issue. There are many others. But I see this as a huge problem. Tell me about your process of improving data quality at Nubank.

[00:54:21] SN: Sure. The one problem that you mentioned now that is certainly a big issue that how we maintain or how do we going to make sure that a certain column has the certain meaning and that meaning is known to every downstream user who relies on that.

We as an organization, we have taken multi-pronged approach to addressing this, I would say. I mean, this problem also ties closely to data discovery in the sense that how do – There is this given dataset or like you need a particular dataset. You do not know how to find it or you already have access to the dataset but you are not clear of the meaning of it.

Like I said, we have a multi-pronged approach to it. One of the approach is to – For a core team, a central team, which works really close to the data infrastructure team to kind of curate certain core canonical datasets and mark them as like this is what – To document them as like this is what this dataset pertains to. This is what the columns mean, etc. Another approach is of course to build an automated service on top which aids data discovery, which helps with users understand their data.

On top of it, another concern that when I said I work with data quality is also about pushing the responsibility of monitoring or like being responsible for the datasets to the owners of the dataset. What I mean by that is as a data infrastructure team, there are like tens of thousands of datasets which are computed as ops in our data platform every day. It is really hard for a data infrastructure team to be cognizant of what kind of anomalies are flowing through them. Is today's run anomalous in some sense?

What we're trying to do is also kind of push or give more power to the owners of the dataset and saying that, "Okay, these are my expectations of this data. It supposed to take this certain shape. This column is never supposed to be null, or this is supposed to be an account ID.

In that sense, I'm alluding to a lot of different things here. But in general, what we are working towards is also to kind of give more power to the user to monitor their dataset expectations for them so that downstream, users of that dataset can rely on these datasets.

In a sense, I would tie it to how a DevOps team would work in the sense – Platform infrastructure team would work that a platform team provides abstractions through other engineers in the organization to spin up their services and to reliably monitor them so that they take to their SLAs. In that same sense, if you were to think of a dataset as a service, the dataset owners should be able to guarantee certain SLAs on their dataset so that downstream, consumers of that dataset can reliably use them.

[00:58:04] JM: I think you're really touching on something that is only going to grow in importance in the near future. I think people are afraid to build data applications when they don't have a strong belief in the integrity of the datasets and it's hard to have faith in the integrity of the datasets when you know how many different people are writing to the data lake and how many different heterogeneous datasets are being combined and are they all being updated at the same time? Is something out of date?

If we're just building machine learning models that sort of paper over the inconsistencies in our data, maybe that's okay in some cases. But in other cases, like you really need data integrity. If you don't have somebody that's looking up and down the data pipeline to ensure that data integrity, it's very unlikely you will have it.

[00:59:09] SN: Correct. In general, the best way to kind of ensure data integrity is to assign certain ownership of the data to a certain team and then make sure the team or individual really sticks – Like I said earlier, it's important that we treat a dataset as something more than just a lump of data and more as a service, wherein you are guaranteeing certain service level agreements to downstream users. If you're not keeping up with that kind of SLAs or integrity, then downstream consumers know that they should not rely on this data and maybe should rely on some other sources of data within the organization.

[00:59:59] JM: Okay. Well, I now we're over time. I just want to ask you one last question. I did a series of shows on these different challenges banks, including Nubank. I talked to the CTO I think a couple of years ago, and it's really amazing how much more value there is in financial data when you're building a bank from the ground-up with new technologies, with cloud technologies and better open source tools compared to the old school banks. As somebody who is inside one of these newer banks, I just like to get your perspective, like what do you think banking is going to look like in 5 or 10 years? What can we expect as consumers in the future of banking?

[01:00:46] SN: As somebody who's not much in touch with the business side of things, I would still have a very outsider view on things I would say. But still, in the end, I would say it is about giving more power to the customer. I mean, the current archaic traditional banks work is that there is no transparency to the way your data is used, to how it is processed, how it is being used for you or against you.

I mean, in future, as the banking industry matures, I would hope to, and this could be wishful thinking that we have more transparent, more open systems where at least with regards to data, we know for a particular customer how it is being used for you and how does it affect the service that the bank provides you. That is my hope for the future. But like I said, you should take this with a grain of salt.

[01:01:51] JM: Sujith, thanks for coming on the show. It's been really fun talking to you.

[01:01:54] SN: Thank you. Thank you, Jeff.

[END OF INTERVIEW]

[01:02:04] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[END]